Ontology-driven Education: Teaching Anatomy with Intelligent 3D Games on the Web

Trond Nilsen

A dissertation
submitted in partial fulfillment of the
requirements for the degree
of
Doctor of Philosophy

University of Washington
2015

Reading Committee:

Thomas Furness, Chair
James Brinkley
Christina Mastrangelo
Steve Tanimoto
John Clark

Program Authorized to Offer Degree:

Industrial & Systems Engineering

University of Washington

Abstract

Ontology-driven Education: Teaching Anatomy with Intelligent 3D Games on the Web

Trond Nilsen

Chair of the Supervisory Committee:

Thomas Furness, Professor

Industrial & Systems Engineering

Human anatomy is a challenging and intimidating subject whose understanding is essential to good medical practice, taught primarily using a combination of lectures and the dissection of human cadavers. Lectures are cheap and scalable, but do a poor job of teaching spatial understanding, whereas dissection lets students experience the body's interior first-hand, but is expensive, cannot be repeated, and is often imperfect. Educational games and online learning activities have the potential to supplement these teaching methods in a cheap and relatively effective way, but they are difficult for educators to customize for particular curricula and lack the tutoring support that human instructors provide. I present an approach to the creation of learning activities for anatomy called ontology-driven education, in which the Foundational Model of Anatomy, an ontological representation of knowledge about anatomy, is leveraged to generate educational content, model student knowledge, and support learning activities and games in a configurable web-based educational framework for anatomy.

To Svanaug and Alan, my parents

Acknowledgements

I am grateful for the help and support of many people during the preparation of this work. I would particularly like to thank the members of my committee Tom Furness, Jim Brinkley, Steve Tanimoto, Christina Mastrangelo, and John Clark for their advice and support, with special thanks to Tom and Jim for their assistance with funding throughout my time at the University of Washington. I would also like to thank Onard Mejino, Todd Detwiler, Nolan Nichols, Melissa Clarkson, and the other members of the Structural Informatics Group for their assistance with the Foundational Model of Anatomy and related technologies; the students in the CSE 590D seminar and the many friends and colleagues with whom I have had productive game design discussions over the years; my sister, Maja, for her help with proof-reading; and, most importantly, my lovely wife, Bridget, whose support and unswerving belief in me helped me keep moving forward when it seemed I was going nowhere.

Table of Contents

Table of Figures

Chapter 1. Introduction

The human body is one of the most complex structures known to mankind. It is no surprise, then, that students find the prospect of its study both challenging and intimidating [1], [2]. Mastery of anatomy requires that students learn a dizzying array of facts, including the name, shape, and placement of thousands of body parts; the way these parts relate to one another in larger structures, networks, and systems; and the way these parts interact with the vascular, nervous, and lymphatic systems that supply nutrients, control, and drainage. In addition to this factual knowledge, students must learn the skills of speaking in anatomical terms, reasoning through anatomy to interpret symptoms and make diagnoses, and thinking spatially to identify structures within an extremely complex context. Finally, they must apply this knowledge to understand the developmental process through which adult anatomy arises from the embryo and the many ways in which genetic, injury, and disease processes might affect anatomy and lead to pathology. As a domain, anatomy is composed of a mix of declarative, conceptual, and procedural knowledge with symbolic, spatial, visual, tactile, and kinematic dimensions.

Though difficult to obtain, a solid grounding in anatomy is essential for those seeking to practice medicine, be they doctors, nurses, paramedics, or radiologists [3]. Outside medicine, anatomy is highly relevant in a variety of other fields that work with the human body: yoga instructors, sports coaches, dancers, and massage therapists, to name but a few, all benefit from an understanding of anatomy. Even in daily life, we rely on some level of anatomy to interpret injuries, physical performance, and our general aches and pains.

Anatomy is taught from kindergarten all the way through medical school. We begin as infants by learning the most basic parts of the body as we learn language. Then, in school,

we learn about the body's major organs, systems, bones, and muscles, usually as part of a general science curriculum and often linked with a study of physiology. Advanced study at university begins with pre-med classes that often incorporate animal dissection to support understanding of the human body, and culminates in medical gross anatomy classes incorporating human dissection.

The methods used to teach anatomy vary widely as we pass through these stages. Children learn much basic anatomy through immersion as they talk with others and learn about themselves. Classes in school employ a wide range of techniques, including text books, quizzes and other exercises, and, in some cases, basic animal dissection. University education employs lectures, textbooks, and some dissection, while a class in medical school likely employs a combination of human dissection, prosection, lectures, text books, and living anatomy activities. Alongside these formal pedagogical tools, online resources might be used by some students to rehearse and reinforce what they learn in the classroom.

Online resources for teaching anatomy exist in a wide range of forms, from test-yourself quizzes and virtual atlases, to interactive text books and more. Unfortunately, most resources do a poor job of engaging students, being static presentations of information with often limited structure and guidance. Interactive learning activities are largely absent, while quizzes, being the notable exception, promote only the most basic form of learning by drilling students in rote memorization. In other fields, games and other interactive learning activities have been found to offer a useful adjunct to traditional teaching methods [4], and it is both unfortunate and surprising that they have not been widely leveraged to teach anatomy, particularly given the field's spatial and visual nature.

My dissertation research addresses some of the challenges implicit in teaching anatomy through games and interactive learning activities by developing methods based on the Foundational Model of Anatomy (FMA), a detailed representation of knowledge about human anatomy. I develop, discuss, and evaluate several prototype systems that support the creation and automation of learning resources, and present uses of these systems in

communicating and teaching knowledge about anatomy. By combining these methods, I demonstrate an approach to creating educational games and learning activities that I refer to as game templates, where a set of rules about user interface and game logic combine with separately authored curriculum and content information to create educational games on demand to satisfy particular educational needs. Finally, recognizing a lack of design knowledge in this space, I present a series of game designs and concepts along with an account of my use of a method known as the Morphological Box to explore the design space and create these ideas.

The remainder of this chapter further establishes the context for this research and outlines my vision for the use of pre-existing domain-specific knowledge representations to automate and supplement the creation of games and learning activities.

- Chapter 2 provides detailed coverage of background material, including a review of anatomical education, an overview of the FMA and several necessary technologies, and a detailed survey of existing learning resources for anatomy.
- Chapter 3 presents my research goals and strategy in detail, and includes caveats regarding my assumptions, notes on the scope of my research, and discussion of the various limits and implications of my approach.
- Chapter 4 presents the Anatomy Engine, a framework and engine that uses the FMA to support the development of content and applications for teaching anatomy, with discussion of the technical aspects of the system, and the results of an expert review by a group of students, educators, ontologists, and developers.
- Chapter 5 documents my foray into the realm of intelligent tutoring, including efforts to leverage the FMA to provide pedagogical support through automated assessment, curriculum specification, and student modeling.
- Chapter 6 offers the game concepts and designs I have created using the Morphological Box and provides an account of my experience with that method alongside suggestions for its use in game design in general.

- Chapter 7 integrates the ideas and software presented in previous chapters to create the Anatomy Learning System, a platform for the construction of educational games that demonstrates my overall vision for ontology-driven games, along with an example quiz game built using it.
- Finally, Chapter 8 concludes with a summary of my contributions and findings followed by discussion of future research directions stemming from my work.

1.1 Teaching Anatomy

Classically, anatomy is taught in medical school, through a combination of lectures, dissections, and living anatomy labs. Of these, lectures are often derided as old-fashioned learning environments where a "sage on the stage" verbally transmits knowledge to passive learners [5]. This judgment may be overly harsh, as lectures presented by a skilled teacher may also be highly engaging and interactive experiences. Dissection supports highly active learning and allows students to view and interact with anatomical structures in context, but is limited by the fact that once cut apart, a cadaver cannot be put back together in order to repeat the activity [6]. Dissection labs are also extremely profound and emotional experiences with wide ranging effects both positive and negative; while some students may gain a greater sense of respect for life and the human body, others may be scarred or deterred by such a dramatic confrontation with mortality [7]. Finally, dissection relies on a supply of human cadavers which may be precarious or ethically problematic [8]. Living anatomy classes provide an opportunity for students to get hands-on experience with living human bodies, but are limited to external features and internal features that can be felt from the surface or located using landmarks [9]. Self-directed learning is normally limited to textbooks, which act as detailed repositories of knowledge but support little in the way of active learning and present anatomy in an idealized way that is detached from the messiness of reality.

Just as learning anatomy is challenging, teaching anatomy in today's environment is equally so. First and foremost, dissection labs, the gold standard of teaching anatomy, are

expensive, time consuming, and resource intensive. Lab space is required, a donor program must be maintained, equipment and materials for embalming and maintaining cadavers must be available, and tutors must be employed in a relatively high ratio to students. Despite these challenges, dissection continues to play a large role in anatomy education, with most schools that have abandoned dissection later reinstating it [10][11]. Living anatomy and lectures support dissection labs, but are no substitute, as are textbooks.

Aside from practical challenges, anatomy is difficult to teach because it requires that students learn so many facts in a single course covering a broad range of knowledge types, including visual characteristics (shape, color, and texture), three-dimensional structure, networks, partitions, connectivity, and tactile experience, not to mention spatial, reasoning, and linguistic skills. No one technique supports teaching all of this material – even dissection must be supplemented by atlases, schematic diagrams, textbooks, discussion, and problem solving.

In short, anatomy is both difficult to teach and difficult to learn, and though dissection is and must remain at its center, there is a need for tools and innovation that support dissection in the laboratory, present material in more accessible ways during study, offer interactive activities to aid in meaning construction and memory formation, and allow students to repeat learning experiences once dissection is over.

So far, discussion has focused on prospective physicians in a medical school context. This presents an incomplete picture, however – almost everyone has a need to learn at least some anatomy, and many non-physicians must possess advanced knowledge to be effective and safe in their work. Nurses, physical therapists, and non-physician specialty clinical workers such as sonographers all receive some instruction in anatomy as part of their training, though most stop short of getting the full dissection experience. Those working in physical performance such as athletes, trainers, dance instructors and yogis all benefit from knowledge of anatomy, though few receive any direct instruction in it. Even members of the general public require anatomy on occasion to perform first aid and

interpret illness and injury in themselves and others. Non-physicians, then, stand to benefit significantly from tools and innovations that recreate the dissection experience to some extent and offer quick and accessible means to learn anatomy. Even fully-trained physicians may benefit from tools that assist them in maintaining their knowledge and communicating with patients.

1.2 Computer-based Learning – Resources, Activities, and Games

In the last decade and a half, there has been a flowering of computerized tools that teach and support those wishing to learn anatomy. These take the form of computerized resources available either as installable computer software, online via the web, or, more recently, on mobile and tablet devices via the various app stores. Throughout the remainder of this dissertation, I will refer to these simply as learning resources, except where it is necessary to distinguish them from resources available in other forms, such as physical books and atlases.

These learning resources come in a number of forms. They may be standalone teaching resources such as tutorials and text books, testing and test preparation tools such as quizzes and flashcards, reference resources such as glossaries and atlases, or learning activities such as games and problem solving exercises. There are a great number of these resources available – my survey counts approximately 500 distinct resources currently focusing on adult human anatomy. I provide a detailed review of these in section 2.8.

These resources are no substitute for dissection labs or human tutors, no matter what the marketing departments of some of the companies producing them say, nor is it productive to critique them as such. Rather, they are best considered and employed as supplements that help students learn in existing classes and review material once classes are over. Only in the case of those who cannot or will not receive formal instruction should they be considered the primary source of knowledge.

I shall focus primarily on games and related interactive learning activities such as quizzes and interactive tutors. Aside from describing them to provide context, I will spend little

time discussing static textbook-like resources and atlases. As discussed in section 2.1, I will use the terms "educational game" and "learning activity" interchangeably to describe the general category of resource that I am interested in, despite theoretical differences between the two. Where brevity is a concern, I will rely on the words "game" and "activity" if it is clear from context that I am not using these words to refer more generally to other types of game or activity.

1.2.1 Strengths

While the computerized imagery in interactive learning resources is no substitute for the physical experience of dissection, they have several strengths that other means of teaching anatomy do not.

1.2.1.1 Frequent updates

In today's internet culture it is common, even expected, for resources to be regularly updated with new technology, content, and design. Even downloadable resources are regularly updated, usually every year or so. Changes and updates are not inherently beneficial, but in practice, the regular update cycle of most software acts iteratively to correct errors and refine and improve the user's experience.

1.2.1.2 Global audience

Material released online is, by default, available to a massive global audience. Coupled with good feedback mechanisms this results, in the best examples, in extremely polished products. Furthermore, when the audience is massive, the price is distributed broadly and resources are often free or cheap, making them more available to people in disadvantaged circumstances. Similarly, investment in building and improving resources is much easier to justify when the audience is large.

1.2.1.3 Support for student communication

Online resources often tie into systems for distributed communication between students, such as forums, chat boards, and help sites. Not all students will use these supports, but those who do report greater satisfaction and obtain better grades [12]. Similar support

networks exist in the traditional classroom, but online systems have advantages in that they are anonymous, thus reducing the social threat of asking questions; shared, in that students can read answers given to other students; and persistent, in that students can revisit answers given in the past.

1.2.1.4 Repeatability

Unlike dissection labs and lectures, interactive learning resources can be employed again and again in review sessions later in a course or as refresher material well after a course has ended.

1.2.1.5 Diverse interactivity

The flexibility afforded by general purpose computing devices allows the creation of learning resources that exploit a virtually limitless range of interactivity. Active learning is known to be highly effective at stimulating the meaningful construction of knowledge [13], and while dissections also provide an opportunity for active processing of information, they are not repeatable.

1.2.1.6 Support for game-like motivational structures

Games designed for entertainment utilize well-constrained goal and decision spaces with regular feedback and clear information to create intrinsically motivating activities that border on addictiveness [14]. Though the superficial fictive and frivolous aspects of such games may be less apparently useful in education, it is clear that by utilizing design ideas and motivational techniques derived from games, one can develop compelling learning experiences that engage and enthrall students.

1.2.1.7 Visual representation

In dissection, structures of interest are presented in the whole body context, and may be obscured, miniscule, or even completely absent. Learning in this context requires a substantial amount of what Mayer calls extraneous processing [15], being cognitive processing that is required but that does not support the learning objective. Good visual design using, for example, schematic and simplified diagrams, greatly alleviates this

problem in text books and atlases, but neither are interactive, whereas interactive learning resources can be both.

1.2.1.8 Multi-dimensional organization

Most real world experiences and physical resources follow some kind of linear organization. Text books read from front to back, lectures go from start to finish, and though dissection can proceed in a variety of orders, the student's experience nonetheless goes from an intact body to one that is not. Computerized resources support diverse organizational structures, allowing material to be found via search (as an index in a textbook might support); via hypertext; via systemic, regional, and other schemes; and via interactive navigational methods such as interaction with a 3D model. This allows students to approach material from a variety of directions, reduces the time taken to find material, and improves the visibility of the linkages between different pieces of material.

1.2.2 Challenges

Despite the several strengths of online learning resources, those seeking to develop and adopt them face several challenges:

1.2.2.1 Content creation and modification is technical and time consuming

Interactive learning resources require a great deal of time, effort, and specialized skill to construct and modify or extend [16]. Unlike paper resources, which can be created or modified by anyone with a word processor or pen, interactive online resources are constructed using a variety of technologies that require specialist technical skills that few educators possess or have time to learn.

In principle, this problem could be partly resolved by a system that generates and customizes activities based on high level instructions from a student or educator. Such a system would rely on libraries of assets, such as art, 3D models, characters, and questions that can be combined in different ways according to rules for constructing a particular type of activity. This approach is hardly new – early learning systems such as Uhr's from 1969 [17] implemented it in a simple form by selecting and grouping quiz questions into learner-

specific activities. More recently, game designers have used this approach in conjunction with procedural generation to create games that are never the same twice; examples include rogue-like games [18], where game worlds are created on demand according to player configuration, and randomized play games such as Strange Adventures in Infinite Space [19], where each 20 minute play session includes a world, plot, characters, and goals freshly generated from a set of components.

1.2.2.2 Diversity vs quality control

Over the last six years, several ecosystems of mobile and tablet devices have emerged and become widespread, with 58% of Americans owning a smart phone as of 2014 [20]. App stores that have evolved alongside these devices now provide a distribution mechanism that makes it far easier to publish new software and resources. Similarly, in the preceding decade, the emergence of the world-wide web greatly simplified the publication of information, enabling the entire class of online learning resources that have now largely replaced software purchased on CD and floppy disk.

With a lowered barrier to entry comes reduced quality control. Previously, most resources were published by universities and academic publishers with strong editorial and review practices. Resources published on the web and through app stores, however, often go through no such process, and are frequently replete with errors and abnormal usages of anatomical vocabulary. Though improvements in quality are available to all content producers simply by employing proper experts and engaging in better testing, even experts differ on the use of many terms, and so the employment of standardized vocabularies and ontologies will, over time, likely come to be considered best practice.

1.2.2.3 Lack of tutoring support

Interactive learning resources may help students learn, but only rarely do they have any ability to measure that learning or support metacognition. Of currently available resources, quiz software comes closest, in that it often reports how many questions a student got correct, information which, if questions are constructed well, a student or tutor can interpret to determine progress. Games often award points, but there is little or no

assurance that scoring criteria bears any resemblance to balanced and judicious grading. Furthermore, while well-constructed games for entertainment regularly support mastery learning [21] in game tutorials, there is substantial evidence to suggest that more detailed individual tutoring helps students perform substantially better than those who learn in a shared classroom setting [22], with the difference in average performance found to be as large as two standard deviations [23].

Individual human tutors offer several key advantages that may explain this benefit [24]. Good tutors can assist students by:

- Customizing learning activities to balance difficulty with ability [25];
- Scaffolding learning and building confidence by gently confirming correct actions and supporting the risk-taking necessary to attempt new solutions [26];
- Motivating students to learn from their failures rather than being discouraged by them [27];
- Adapting their teaching style to the student's learning style [28][29];
- Assisting students in uncovering their own misconceptions through questions and counter-examples [30][31];
- And supporting students' metacognition, allowing them to assess their own progress.

All of these benefits can be traced to the tutor's ability to closely observe and respond to the individual learner. Unfortunately, however, individual human tutors are neither practical nor cost effective in most contemporary educational settings, and so have long sought to develop techniques for replicating these effects in software. These techniques have not yet been applied to resources for teaching anatomy, but may offer significant improvement in learning outcomes.

1.2.2.4 Design

The possibility space for designing interactive learning resources to teach anatomy is diverse, and the potential of the medium can only be fully understood by broadly exploring that space. Unfortunately, research efforts involving games and interactive learning activities tend to be narrowly focused on particular design ideas and do little to illuminate the design space, with results that are often heavily dependent on a particular implementation. Similarly, although professional game designers have a great deal of knowledge of how to create games for entertainment, there is yet little consensus on how best to design games for education [32], let alone games specifically for teaching human anatomy.

1.3 Anatomical Knowledge Representation

It should be no surprise that anatomical entities and terms are ubiquitous in virtually all texts that discuss the human body in any depth, including not only medical texts, but texts about sports, dance, singing, massage, and even self-improvement. Rosse & Mejino [33] explain this ubiquity elegantly and formally as being due to the fact that:

> *Anatomical entities are the independent continuants of biomedical reality on which physiological and disease processes depend, and which, in response to etiological agents, can transform themselves into pathological entities.*

Also obviously true, but not so obviously important is the fact that anatomical entities have been an important part of the vocabulary and discourse of every human culture throughout time.

Such diversity of usage presents a challenge to those seeking to educate and to integrate knowledge, as with it comes diversity of terminology and, more subtly, diversity of meaning. In today's globalized world, where medical information must be meaningful across borders and cultures (for example, to enable the communication of clinical and research information), there is need for formal methods for determining equivalency between terms, labels, and references in both written and computable resources.

For this reason, there have long been efforts to standardize anatomical terminology, beginning in 1895 with the Basle Nomina Anatomica [34]. It has evolved into the Terminologia Anatomica (TA) [35], developed by the Federative Committee on Anatomical Terminology and the International Federation of Associations of Anatomists, and released in 1998 [36]. The TA presents standard names for 7500 structures in English and Latin alongside a unique numerical code and detailed notes justifying naming decisions.

Though a landmark effort of high importance, the TA is insufficient in several ways. Firstly, the human body contains many more than 7500 unique structures; secondly, the TA does not account for laterality (for example, the left temporal bone and right temporal bone are the same within it); thirdly, it provides no guidance for the meaning of its terms; fourthly, it does not tie into other ontologies used in specialist domains (for example, brain parcellation schemes) and fourthly it includes little information about the way in which structures relate to one another (though some partitive information can be inferred from the numeric coding scheme). As a result, it offers no support for interoperability in modern clinical and research environments.

The Foundational Model of Anatomy (FMA) moves beyond the TA by shifting from a vocabulary model to that of a reference ontology. This entails representing relationships between structures, adding additional layers of classification and taxonomy, and, perhaps most importantly, supporting the application of inference algorithms that use the model to answer questions about anatomy such as "which vertebra gives rise to the nerve fibers that connect to the muscles of the little finger?"

The FMA, then, is a detailed representation of much of what is known about the structure of the human body. It is discussed in greater depth in section 2.5.

Chapter 2. Background

In this chapter, I provide an overview of the various technologies and fields of research on which my work rests. A full discussion of each of these fields would take up far more space than is available – I will thus limit myself to presenting only enough detail to understand the remainder of my dissertation and assess where it fits into prior work, and no more.

Throughout this chapter, I will address the following background questions:

- What do I mean by "educational game" or "learning activity"? How are they different from each other and from quizzes, tutors, or any other form of learning resources?
- What is anatomy? What sorts of anatomy have I included in my investigation?
- What constitutes anatomical knowledge? That is, what sort of material am I developing software to teach?
- How is anatomical knowledge represented digitally? What operations can be performed on this representation? How can it support educational tools?
- How can software tools support education? Which approaches to doing so are relevant or useful in my work?
- What technologies will I need to produce digital educational games? Which pre-existing frameworks can I leverage?
- What digital resources for teaching anatomy exist already? What can I learn from them?

2.1 Vision - Ontology Driven Education

My work is driven by the conviction that the Foundational Model of Anatomy, a detailed and computable representation of knowledge about the structure of the human body, ought to provide invaluable support for the construction of interactive learning resources. This conviction is more general, and would apply to any well-developed knowledge representation in any field that is complex to learn and teach. Human anatomy provides an excellent test bed for this thesis due to the availability and high quality of the FMA and the well-structured and factual nature of much of the knowledge it is comprised of.

To understand this thesis, it is important to conceptualize the FMA not merely as a source of terminology or a tool for standardizing the way information about anatomy is represented, but as a repository of knowledge that allows a computer to intelligently reason about anatomy. In this way, it is similar to the knowledge representation approach to artificial intelligence pioneered by projects such as CYC [37], which since 1984 has attempted to codify all of the knowledge necessary for general inference about day-to-day human affairs, and which, despite immense challenges, has found applications in clinical reasoning, counter-terrorism and defense analysis, and encyclopedia building.

By allowing computers to reason about anatomy, I contend that the FMA can support the creation of interactive resources by:

- Interpreting high-level instructions to generate content for use in interactive resources; for example, allowing an educator to specify that a game should use "all the muscles of the face" rather than requiring that they specify structures one by one.
- Asserting relationships that can be leveraged to tailor the visual display of structures; for example, coloring structures according to what class of entity they are, what group of structures they belong to, what they are connected to, or even what data values they are associated with in some data set through the FMA's use as a source of standardized data annotations.

- Acting as a source of knowledge that can be used to automatically assess student actions and responses; for example, grading quizzes or generating knowledge-based events that a game can use in scoring.
- Defining the domain model required to create an overlay-based student model to manage and compute estimates of a student's knowledge as they play.
- Providing a common language for educators to specify the material that should be taught in a game and ensuring that structures are consistently named throughout.

My overarching goal has been to explore whether such applications of the FMA are feasible, to construct prototypes where possible, and to assess the viability, challenges, and potential of these applications. I have furthermore sought to integrate these tools into complete game generators by combining them with game and interaction rules, called game templates, to dramatically accelerate the authoring of intelligent learning activities and games and support authoring by educators with little to no technical skills.

Finally, I have worked through a structured design process that has enabled me to explore broadly a possibility space within which many interactive learning resources for anatomy fall. This process has allowed me to characterize the broad possibilities of this medium for teaching about the human body.

In the following sections, I will present a variety of background material that supports the work presented later in this dissertation and provides the context into which my contribution will fit. Chapter 3 will revisit my research goals and articulate more precisely the hypotheses I have sought to test in this work.

2.1.1 Ontology - definition

The term "ontology", as used here, differs significantly from the more common definition used in philosophy. Where philosophical ontology is the study of existence and of how entities in the world can be conceived and classified, ontology in computing and information science is a formal representation of the concepts and entities used in some

domain of knowledge, and can be thought of as a concrete application of that philosophical field of study.

Ontologies are closely related to taxonomies, but go beyond term specification and classification to define relationships, hierarchies, types, and other properties of entities. Ontologies can be interrogated to extract the information explicitly defined within them, but may also be the subject of inference algorithms that generate new knowledge according to logical rules.

This work is largely based on applications of a detailed ontology called the Foundational Model of Anatomy, discussed in depth in section 2.5.

2.2 Games and Learning activities

The concept of a game, like that of art, has historically proved very difficult to define, partly because the concept is somewhat amorphous and contextual, and partly because different groups have at different times exerted cultural control over it; witness the change in the meaning of the word "gaming" from once primarily referring to gambling to now referring to computer and tabletop games played solely for fun [38]. Some notable areas of dispute that apply to educational games and learning activities include whether productive activities can be properly considered games (early theorists, notably Caillois [39] and Huizinga [40] would argue that they cannot) and whether open-ended game-like activities such as toys (e.g. Lego [41]), simulations (e.g. SimCity [42]), or role-playing activities (such as tutoring simulations; see examples in section 2.8) are in fact games.

In general, the best way to define both games and art is polythetically; that is, using a list of criteria that are individually neither necessary nor sufficient, but that taken together allow candidates to be considered more or less game-like. For detailed discussions on both subjects, see [14] and [43]. For my purposes, I shall build on the working definition provided in [14]:

> *A game is a system in which players engage in an artificial conflict defined by rules, that results in a quantifiable outcome.*

The key points in this definition are that games have players; these players engage in some conflict with each other or with the game system itself; the activity is governed by rules; and there is some kind of quantifiable outcome such as a score or victory state.

The term "learning activity" is easier to define – I consider it to encompass all structured activities in which learning occurs or is designed for. In the context of my work, I focus on computer-based learning activities that are online, ideally running within a learner's web browser, and interactive, in that they require action on the part of the learner beyond passive consumption, such as reading text or watching videos.

Learning activities and educational games are closely related but not identical. The term "educational game" implies an activity that is self-contained, enjoyable for its own sake, and more complex than learning activities such as quizzes. The term "learning activity" is more neutral than "game" in that it does not have any baggage associated with it, such as prejudices about games being distracting wastes of time. It is also broader, and includes quizzes, tutors, and simulations, which are not necessarily games. For brevity's sake, however, I will use the terms "educational game" and "learning activity" interchangeably. Both refer to resources that support learning through interaction, and both can be generated and integrated with tutoring tools using the methods I describe. The one exception to this is in Chapter 6, where I will focus primarily on games rather than activities, the reason being that existing design knowledge already amply covers the creation of learning activities.

2.2.1 Mayer's multimedia theory and games

An opposing view to the use of games in education is based on Richard Mayer's cognitive theory of multimedia learning [13], which argues that the cognitive processing required of students should be minimized by removing distractions and extraneous tasks from learning activities. Games, on the other hand, introduce large amounts of material in order to support play, including narrative, additional tasks, and content in the form of graphics, sound, and text.

These two competing views impose a trade-off between communicating knowledge in a direct manner that maximizes the efficiency of learning, and framing it in a context that makes learning more appealing and more accessible to students so that they begin more easily and persist for longer. This implies different designs that support the needs of different students: clear, direct learning materials for highly motivated students, and games and other motivational scaffolding for students who have trouble engaging.

This is supported empirically. In a study conducted with colleagues at the University of Canterbury, we found significant differences between students learning undergraduate educational psychology through lectures or a game [44]. Students found game-based learning to be more difficult and reported higher levels of confusion during class, but reported higher levels of engagement and motivation. Many other questions were raised by this research, however, particularly around differences between high- and low-performing students.

2.3 Anatomy

Human anatomy is a comparatively broad term, with 11 definitions in the Oxford English Dictionary [45]. To set the scope of my work, it is necessary to spend a moment defining exactly what form of anatomy I will address.

Broadly, the science of anatomy is defined as consisting of:

> *The body of facts and deductions as to the structure of organized beings, animal or vegetable, ascertained by dissection; [or] the doctrine or science of the structure of organized bodies [45]*

My interest is in anatomy applied to the human body. The techniques and overall pedagogical approach I discuss in this dissertation could be applied to the anatomy of other organisms or even non-living structures such as machines, but I do not discuss this here.

More specifically, I focus on the gross anatomy of the normal adult human. Gross anatomy is distinguished from microscopic anatomy by focusing on structures visible to the human eye, and so the internal anatomy of cells and the organizational patterns through which

cells become tissues is outside the scope of my work. Similarly, I focus on adult anatomy, not developmental anatomy. Though there are obviously important correspondences between the two, I focus on adult anatomy as it is relatively static, extremely well understood, and the dominant subject of the Foundational Model of Anatomy, a chief resource for my work. I also limit my scope by focusing on normal anatomy rather than pathological or abnormal anatomy.

In addition to visual inspection by looking at the body either whole or dissected, anatomy can be understood through the various medical imaging modalities that have been developed over the last century [46]. I focus primarily on an understanding of anatomy through the inspection of visible three-dimensional structures, as contrasted with an understanding through two-dimensional slice representations such as radiographs, ultrasound images, and CT slices. 3D volumes are constructed using imaging bridge these two categories, but I largely do not focus on them, except inasmuch as they are a means of obtaining 3D models.

2.4 Anatomical Knowledge

As mentioned in the introduction, the science of human anatomy is based on a body of knowledge comprised of an array of facts, a series of skills, and groups of concepts defining several processes. Since different types of knowledge require different teaching strategies, it is necessary to understand the individual components of anatomical knowledge in some detail.

Unfortunately, there is no single well agreed upon curriculum or list of teaching goals for the study of anatomy [47]. An examination of course curricula published by various universities shows general consensus about the core elements, but the descriptions contained within these are too broad to be useful in this context, and so in the following discussion I will attempt to break down the anatomical body of knowledge into specific components.

2.4.1 Facts

The Terminologia Anatomica [35] and the Foundational Model of Anatomy [48] and its precursors [49] offer formal descriptions that define some subset of the anatomical body of knowledge, focusing specifically on facts that can be specified symbolically, such the names and many of the properties of individual structures as well as the relationships between them. Properties such as color, shape, texture, and elasticity are not captured by these representations, however, nor do they provide any insight into the skills students learn as a result of studying anatomy.

Following the FMA, I define anatomical facts as being the properties and relationships of anatomical entities, including immaterial entities such as anatomical spaces and foramina, and constructs such as anatomical lines, landmarks, and planes. From these facts, I have identified the following categories:

- **Visual characteristics** – the shape, color, and visual texture of an entity
- **Nomenclature** – the names of an entity as well as the meaning of the patterns and word parts that make up these names
- **Spatial relationships** – the layering, placement, and spatial arrangement of an entity with respect to other entities and to the body as a whole
- **Partition –** the partition of an entity into smaller entities, landmarks, and features
- **Connectivity** – the continuities between entities that allow them to form networks and other structures
- **Supply** – an entity's supply of blood, innervation, and drainage
- **Development –** the precursor entities from which an entity derives or is formed
- **Physical characteristics** – an entity's physical texture, elasticity, and feel

I do not claim that this categorization accounts for everything that can possibly be known about anatomy; I claim only that it encompasses most of the facts taught in a human gross anatomy class.

2.4.2 Skills

The skills portion of the anatomical body of knowledge is less distinct and well defined. Drawing on various critiques and discussions of the role of anatomy [47][50][51] in the medical curriculum as well as guidelines from the Human Anatomy and Physiology Society [52], I suggest that the key skills within the anatomical body of knowledge are:

1. **An ability to recognize structures both individually and within context:** Anatomical entities seen in the body rarely have "textbook presentation" and are often partially concealed or damaged. Students must be able to confidently and correctly identify entities based on their visual and structural properties and context. Furthermore, they must be alert to variations in structure that may account for problems or indicate new risks.
2. **An ability to reason and make inferences using anatomical facts:** Knowledge about the structure of the body is fundamental to physiological and diagnostic thinking. In particular, it supports prediction of the physiological effects of changes in anatomy due to disease, aging, or injury.
3. **An ability to speak clearly in anatomical terms:** Effective communication in a clinical setting relies on rapid and precise encoding of descriptions and instructions concerning the body. To become proficient at this, students must learn to deploy and interpret anatomical vocabulary within a variety of standard idioms.
4. **A sensitivity to working with the human body:** During medical consultations and procedures, patients are vulnerable and in a position of weakness, even if they are otherwise well. The study of anatomy is usually the first opportunity students have to work directly with mortality, morbidity, and the living human body, and arguably many of the personal, social, and emotional skills necessary to work in medicine can best be learned as part of this study [53][54].
5. **Laboratory skills:** Many of the physical skills necessary in medical practice are derived from skills learned in the laboratory, including dissection itself, histological

skills, and laboratory procedure. Furthermore, skills such as palpation and auscultation can be seen as deriving from techniques learned in living anatomy sessions [51][54].

6. **An ability to synthesize anatomical ideas with those from other fields:** Anatomy is a fundamental subject in that it provides the basis for a variety of other subjects including physiology, pathology, the many medical specialties, and even other more general areas of knowledge such as diet, exercise, and basic injury. Students must develop anatomical fluency so that they can integrate and employ new knowledge from these fields.

Skills 1, 2 and 3 concern recognition, discussion, and reasoning purely within anatomy itself – they are thus integral to its mastery. Skills 4 and 5 are auxiliary to anatomy in that one could claim to fully know anatomy without being proficient in them. Nonetheless, they are skills that are important and often learned within the anatomy classroom. Skill 6 is, to some extent, an external bridging skill applied to anatomy, in that skill at synthesizing knowledge is usually learned outside the anatomy classroom, but must be applied to anatomy and adapted to its peculiarities.

As my work is focused on applying an anatomical ontology to the teaching of anatomy, I am limited somewhat by what knowledge can be encoded within or linked to by such an ontology. As a result, skills 4 and 5, and, to some extent, 6 are outside the scope of my research.

2.4.3 Concepts

Though anatomy is sometimes thought of colloquially as consisting solely of thousands of names and facts, it also contains several important groups of concepts. Many of these concepts are somewhat physiological or developmental in nature, but they offer important reasons for patterns of anatomical structure. For example, the function of sympathetic and parasympathetic nerve fibers offers a useful guide to understanding which structures they connect to, and the embryological origin of many structures offers insight into their

apparently arbitrary layout within the body. Similarly, much anatomical reasoning relies on interpreting the physiological consequences of variations in anatomy.

Unfortunately, the Foundational Model of Anatomy does not encode conceptual knowledge except in that the relationships and entity classes it relies on have conceptual underpinnings that can be interpreted by users. A query on the FMA can determine which blood vessels connect to one another and eventually supply a particular structure, but understanding the implications of this requires additional conceptual knowledge. In my educational applications, this conceptual knowledge is embedded within game templates – for example, a game with mechanics based on blood supply might use the results of the aforementioned query to create in-game consequences when a particular vessel is severed.

For the purposes of understanding the variety of conceptual knowledge implicit in the anatomical body of knowledge, I will rely on my previous classification of facts based on the type of anatomical facts a concept is based on. So, the example above of blood flow is an example of a concept related to facts about connectivity.

2.5 Foundational Model of Anatomy

My approach relies on the application of the Foundational Model of Anatomy (FMA), an ontology that defines and describes the entities necessary to understand the structural organization of the human body. It was created by the Structural Informatics Group at the University of Washington, and is under active development and expansion. I introduce it here to the extent necessary to understand its use in my work; for a more detailed discussion of its structure and contents, see [48] and [33].

The FMA represents anatomy symbolically, specifying both anatomical entities and the relationships between them. At a high level, it can be thought of as a semantic network, a graph whose nodes are entities and whose edges represent relationships.

The FMA was constructed as a frame-based ontology using the Protégé knowledge engineering framework, and is available as a database that can be imported into that tool. Recently, it has been converted into OWL2, the Web Ontology Language [55].

2.5.1 Entities

The FMA incorporates diverse anatomical entities, including:

- **Concrete body parts:** Specific structural elements of the body such as the mandible, the heart, the left lung, and the right foot. Note that laterality is important; the foot is a class of body part whereas the right foot is a specific structure.
- **Classes of body part:** serous membrane, muscle, nucleated cell, limb.
- **Body substances:** Fluids and other non-structural physical entities.
- **Spatial concepts:** anatomical planes, directions, and patterns, as well as immaterial spatial entities such as surfaces and spaces.
- **Systems and sets:** high-level conceptual groupings based on physiological function such as the cardiovascular system and the respiratory system, as well as spatial or organizational groupings such as the set of ribs or the set of olfactory nerves.

Each entity is represented using a data structure called a frame comprised of a number of attributes called "slots", each representing some property or relationship defining the entity. Each slot is constrained in the values it can take; some take only primitive types, while others take strings or relationship instances. The set of slots available to each frame varies depending on the type of entity; this is handled with a dual hierarchy of classes where each frame is both a class and an instance of a meta-class, as described in [48].

Note that individual, specific pieces of anatomy such as Galileo's finger, St Bonaventure's arm, or the brain of Paul Broca have no place in the FMA – the FMA represents the abstract entities we use to talk about human anatomy, not the physical objects themselves. All in

all, the FMA defines some 150,000 terms referring to 75,000 entities, linked together by over 2 million instances of around 200 types of relationship [33].

2.5.2 Relationships

Many types of relationship are represented in the FMA, including partitions, spatial relationships, vascular, supply, skeletal attachments, neural supply, connectivity, typing, homonymy, synonymy, and the class relationship. Relationships are unidirectional and typed; thus, each can be conceived of as a simple statement with a subject (the source entity), a predicate (the relationship type), and an object (the sink entity).

Take, for example, the liver. The FMA expresses an entity called Liver, which is related to another entity called Abdominal Cavity by a relationship of type contained-in, allowing us to simply state that the liver is contained within the abdominal cavity. In turn, Abdominal Cavity is related to Liver by a relationship of type contains. Note that two types of relationship are required to describe this situation – one for the entity containing, the other for the entity being contained.

Certain relationships within the FMA are hierarchical in nature, such as is-a (the class relationship), part-of, regional-part-of, and systemic-part-of. Each of these relationships spans the bulk of the FMA graph, allowing it to be conceived of not as a graph, but as a tree made up from some primary relationship, whose nodes are related to each other in various ways. This simplification is pertinent, as it allows hierarchical decomposition of anatomy; see section 4.7.2.3 for an application of this property.

2.5.3 Inference

The FMA asserts over 2 million facts about anatomy. From this set of facts, many more can be extracted. For example, one part of the entity Left lung is the entity Lower lobe of left lung, which in turn has the entity Left superior bronchopulmonary segment as one its parts. This implies that, at a finer level granularity, we can consider the Left superior bronchopulmonary segment to be a part of the Left lung. This process of extracting new facts from existing ones is known as inference.

Though the example given above is fairly simple, more complex questions can be asked by stringing together a series of inferences and selection criteria in the form of a query. For example, we might write a query to determine the full path of venous drainage from the lungs, the full list of anatomical structures contained within the mediastinum, or a list of the muscles in the right leg. It is this power of inference that makes the FMA such a powerful framework for generating anatomical scenes for us in learning activities.

2.5.4 Querying Engines

The FMA itself is merely a data structure, albeit an extremely large and complex one. As such, queries are performed by external software which interrogates the FMA in order to produce results. The FMA's authors have developed a tool called Query Integrator [56] which supports queries in a number of languages, including XQuery [57] and SPARQL [58]. Both languages are W3C specifications, with XQuery targeted at generic XML data, and SPARQL targeted at RDF, the Resource Description Framework [59], a specialized form of XML for representing ontologies, of which OWL is a further specialization [60].

SPARQL is the preferred language for querying the FMA. At a high level, SPARQL queries can be thought of as patterns to be matched by FMA entities paired with output specifications for listing those entities.

```
PREFIX fma:<http://sig.uw.edu/fma#>
CONSTRUCT
{
    ?con_part fma:name ?values .
    ?con_part fma:FMAID ?fmaid .
}
FROM <http://sig.uw.edu/fma>
WHERE
{
    fma:Lung  fma:constitutional_part ?con_part .
    ?con_part fma:FMAID ?fmaid .
    ?con_part fma:Preferred_name ?name .
    ?name fma:name ?values .
}
```

Figure 1 – Sample SPARQL Query

The query shown in Figure 1 – Sample SPARQL Query offers a simple example which returns a list of the preferred name and FMAID of all entities that are constitutional parts of the lung. The queries can be interpreted as follows:

- PREFIX fma:<http://sig.uw.edu/fma#>

  Specifies an abbreviated prefix that can be used to uniquely identify all entities referenced in the query

  SPARQL queries operate on the FMA in its OWL form, which, as mentioned previously, is a specialization of RDF. In RDF, all entities must have globally unique identifiers so that they can be used in queries that span multiple ontologies. To achieve this, each entity is referred to with a URI, implicitly following the rules for domain names set by ICANN. Since full URI names are cumbersome to work with, SPARQL allows the use of abbreviated prefixes. In this case, the prefix allows http://sig.uw.edu/fma#Lung to be abbreviated as fma:Lung.

- CONSTRUCT { … }

Specifies a format for returning results of the query

SPARQL queries return RDF triples as output, where a triple is a subject, predicate, object statement, as discussed above. In this case, output is generated for all entities ?con_part, determined by matches against the pattern in the WHERE clause, with one triple stating the name of the entity, the other its FMA ID number.

- FROM <http://sig.uw.edu/fma>

  Specifies the ontology the query is to be applied to.

  The Query Integrator allows integration of multiple ontologies in a single query. This clause states the ontology to be queried, in this case the FMA.

- WHERE { … }

  Specifies the pattern to be matched by this query

  The query searches for entities that match all of the triples listed in this clause, with variables beginning with '?'. The pattern can be interpreted as "find all entities that are a constitutional part of the lung, that have an FMAID, and that have a preferred name". Two triples are needed to assess the preferred name since it is not a simple field value, but a reference to a set of metadata about the entity's preferred name, including information about the authority asserting that that name is preferred.

Though the above query is comparatively simple, more sophisticated, semantic queries can be used that employ the FMA's power of inference to produce much more specific lists of entities; for example, all of the muscles in the face, the vascular supply of the spleen, or all parts of the endocrine system.

In addition to facilitating individual queries, the Query Integrator allows developers to specify queries that can be re-used as views by later developers. It also allows queries across multiple ontologies, as well as documents in other formats, such as XML. For example, the Query Integrator could be used to integrate entities from the FMA with entities or data from other sources, such as an ontology of physiological processes, a

database of clinical outcomes, or, as I propose to do, a database of 3D models, thus allowing semantic queries which return model geometry which can be used to construct a 3D scene that can be embedded within a game.

2.6 Applying Technology to Tutoring and Education

In the early days of computing, optimistic researchers in artificial intelligence were quick to proclaim the potential of computers in education, claiming that they could be programmed to understand the student, the subject matter, and the learning process. In the decades since, it has become apparent that the problem of teaching with computers is much more difficult that originally imagined. Nonetheless, a wide variety of techniques and approaches have been explored and found to be effective. I will briefly overview the field, with particular attention to approaches that I believe can be used to leverage the knowledge baked into ontologies such as the FMA to create good learning activities.

Early educational systems went under the moniker CAI, for "computer assisted instruction", and were little more than tools for delivering questions and tabulating a student's answers [61]. As time went on, systems became capable of generating questions according to sets of rules, eventually taking into account the student's past performance and thereby creating a rudimentary personal learning system [17].

By the late 1970s and early 1980s, CAI had evolved into the field known as Intelligent Tutoring Systems (ITS) [62]. ITS research was distinct from earlier CAI work in that it sought to move beyond simply delivering questions and tabulating results into explicitly modeling domain knowledge, the student, and the learning process itself. By conceptualizing their work in this way, ITS researchers were able to break the problem apart and focus on each part individually. Today, the field has divided further, with researchers in ITS focusing more on complex tutoring agents in narrow domains with complex tasks, and those in the broader field of Artificial Intelligence in Education incorporating artificial intelligence techniques in more generalizable computerized

educational tools. Despite this split, the fundamental techniques employed tend to be similar, with the difference lying primarily in complexity and scope.

2.6.1 The Trinity Model of Tutor Design

Human tutors employ knowledge of three aspects of the tutoring process: the subject matter, the student, and the pedagogical actions that may be taken to facilitate learning. Recognition of this fact has led to the classic "trinity model", where educational systems are built around three interdependent components, each corresponding to one of these aspects [63][64]. Existing systems and common approaches to each component are discussed below.

2.6.1.1 Modeling the Domain

By incorporating computable representations of domain knowledge, systems are able to generate fresh learning activities from domain knowledge and assess students through both their direct actions and any finished artifacts they might create. In addition, a good domain model provides a strong foundation on which to model the student's progress.

Knowledge varies in character from domain to domain; it might be a library of concepts and definitions, a collection of facts and statements, a series of skills and techniques, or a combination of all three [65]. While various sophisticated schemes for classifying knowledge exist [66][67][68], in building domain models the key distinction is between assertions of fact or conceptual structure, called declarative knowledge, and sequences of operations, techniques, and decision-making heuristics, called procedural knowledge. This distinction is key, as the strategies used to model declarative knowledge are poorly suited to modeling procedural knowledge, and vice versa [69]. Consequently, designers normally focus on one or the other, with those focusing on procedural knowledge often relying on a basis of declarative statements that define the entities used within the procedures taught.

The depth of knowledge modeling varies significantly. Models fall into the following three classes [43]:

- **Black box models** model the domain implicitly within canned questions and responses. Since they are closed and non-computable, black box models cannot be used in open-ended activities where all correct responses are not known in advance, nor can they be used to support activity generation. Black box models were common in many early ITS and CAI systems, such as SOPHIE, a tutoring system for electronic circuit design [70].

- **Glass box models** represent domain concepts and relationships explicitly in terms that would be familiar to human experts, but rely on algorithms that humans would find unnatural or impractical. While they allow designers to focus on computational efficiency and inferential power, they provide limited support for problem solving and are rarely suited for teaching procedural knowledge. Glass box models are common in systems built around existing knowledge bases and expert systems, such as GUIDON, an educational system built around the MYCIN expert system for bacterial infections [71].

- **Cognitive models** include explicit representations of both the concepts and reasoning mechanisms used within a domain, as understood within some cognitive theory. This allows them to facilitate in-depth feedback that supports a student's reasoning, and makes them particularly appropriate in teaching materials that require significant independent thought and problem solving. As such, cognitive models have been applied to teach problem solving in algebra [72], geometry [73], and LISP [74].

A range of different formalisms exist for knowledge representation. Selection between them involves many considerations, including the following:

- **Expressiveness** – the level of complexity representable in formal expressions

- **Efficiency** – how simply a complex expression can be represented in a formalism

- **Tractability** – the computational difficulty of determining the validity of statements and making inferences
- **Cognitive fidelity** – how closely the formalism maps to human thought processes, as understood in some cognitive theory
- **Vagueness** – much human knowledge is vague, subjective, or probabilistic, and special formalisms are required to cope with this

Common formalisms used in domain models include production rules, semantic networks, conceptual graphs, ontologies, description logics, and constraint-based models; see [69] for an in-depth overview.

Advanced description logics such as those used by web ontology languages such as OWL are among the most complete and expressive formalisms available that remain, with a few notable exceptions, computationally tractable in polynomial time. Since anatomy is primarily a domain of declarative knowledge and since the FMA is already expressible and queryable in OWL 2, my domain model will be constructed using description logic.

2.6.1.2 Modeling the Student

Just as human tutors observe their students in order to improve and customize lessons, tutoring systems collect information through direct prompts, observations of student behavior, and performance tracking, using it to form a student model. This model allows the system to:

- Adapt pedagogy to suit the student's learning style;
- Select learning activities in order to focus on knowledge the student lacks;
- Identify and address any misconceptions the student holds;
- And offer the student insight into their learning progress over time.

Student models include a variety of information – typically anything potentially useful that can be practically collected. Early systems tended to focus on summative assessment, tracking the student's progress by counting correct responses in order to determine

whether they had reached certain learning objectives, while more advanced systems incorporate formative assessment by collecting information about the student that allows them to adapt to the student's needs and wants [75]. Many types of information can be collected, including:

- Demographics such as age and gender can both be relevant in selecting pedagogical strategies, and grade level is useful for determining the complexity and level of detail of learning activities.

- Student preferences may be used to customize the learning experience and select appropriate learning activities, depending on the student's preferred learning style. Preferences may be obtained directly from student feedback, or they may be inferred from student performance during an activity.

- Learning history, that is, the activities attempted and results achieved, is easily collected and provides a context for the student to assess their own work over time. It may also be used by instructors to evaluate whether the student is meeting particular goals or expectations.

- Affective estimates allow the model to infer frustration, boredom, enjoyment, and other emotional states, which may guide future pedagogical choices. Affective states are challenging to determine, and systems incorporating them are an area of active research.

- Knowledge estimates are obviously useful as they allow the system to gauge a student's progress and choose activities that covers material they do not yet know. Two aspects of student knowledge are commonly modeled:

    - Correct knowledge consists of facts, concepts, and procedures the student is believed to know, often represented as a subset of the domain knowledge understood by the tutor in what is called an overlay model.

- Misconceptions are incorrect facts or practices that the student appears to possess. By explicitly recognizing misconceptions, the system can assist students in correcting them.

Much information collected about a student is declarative and definite, allowing comparatively simple representation; the student's name and gender, for example, may be represented as individual variables, while the student's history of activity may be represented in a list structure. Other information is probabilistic and partial [64], and requires more sophisticated representation; in particular, student knowledge and affect are effectively impossible to model perfectly, as all information concerning them is based on imperfect observations and inference. Relatively few pedagogical choices require perfect knowledge, however, and it is generally accepted that intelligent tutoring systems should emphasize pedagogical effectiveness over perfect modeling – if a student scores 20% and 30% in topics A and B respectively, it is no great harm if the system attempts to teach them topic B first, even though this choice may not be optimal [76].

Given that information collected about a student can have several different types, it is often stored using a mixture of data structures. Furthermore, representations must be flexible to enable easy and fast updates so new information can be incorporated as it becomes available. Finally, since the exact information inputs are rarely important, aggregate representations are often used.

Though representations of student knowledge and the domain utilize the same terms and concepts, the former are probabilistic and the latter deterministic, so different formalisms are required. Furthermore, estimates of student knowledge about individual facts may be inter-dependent, with knowledge of one concept implying an increased probability of knowledge of another. Numerous authors have argued for Bayesian Networks as the preferred means of representing such collections of inter-dependent probabilities [51][77][78][75][79].

A Bayesian Network is a graphical representation of the joint probabilities of a series of inter-related variables. Each variable is represented as a node in a graph, connected by

arcs to all variables on which it is dependent. The directionality of these arcs is normally read predictively; that is, a simple network with variables A and B connected by an arc from A to B indicates that A implies B. However, since the relationship between two variables can be interpreted diagnostically as well as predictively there is no mathematical reason that arcs cannot be reversed. Normally, arc direction is chosen both for ease of understanding and simplicity in network design.

A number of issues must be considered when developing a Bayesian student model:

- The network must be computationally tractable so that useful results can be produced in real time. Fully acyclic Bayesian networks are computable in linear time [51][80], and can be sufficient if focus is limited to assessment and adaptive activity selection.

- Variables that represent the student's internal states cannot be used as inputs, as they are hidden. Furthermore, networks containing long chains of hidden variables tend to be error-prone and difficult to validate [77]. Typically, a Bayesian student model should proceed from student actions as observable input variables to estimates of internal states as output.

- Dependencies may exist between related topics, and must be represented if the model is to perform well. Unfortunately, dependencies are difficult to model in an acyclic network. This problem can be addressed by exploiting hierarchies within the domain; that is, by assuming that the probabilities that a student understands subtopics A and B are independent given the probability that they understand the parent topic to which both belong [104][105].

- Estimates of student knowledge should incorporate both current and past performance to capture changes over time. Since a student will cumulatively perform many thousands of actions, the student model must either compress these actions or expand to a very large size. Dynamic Bayesian models are one solution,

where nodes representing past actions are discarded once an estimate has been computed, and this estimate is used in their place in future calculations [75][81].

There are three general schools of Bayesian student model construction [77][75]: expert-centric approaches where the model is created by a domain expert with educational expertise such that the network is thought to closely represent the student's internal states and their dependencies [82]; efficiency-centric approaches that use general rules to create large numbers of computationally tractable models across a whole domain [83][105], and data-centric models that use machine learning to create the model from observations of student behavior over time [77][104]. Each approach has its trade-offs: expert centric models offer high fidelity to cognitive processes but are time-consuming to create, vulnerable to mistakes or misconceptions on the part of the expert, and are less tractable; efficiency-centric models can quickly be created from the domain model and are always tractable, but are often simplistic and lack sensitivity; finally, data-centric models can be difficult to interpret (as the results of machine learning algorithms often are) and require a large body of data in advance to learn from. Given these alternatives, the declarative (and thus comparatively simple) nature of the anatomy domain, and its large size, I will focus my work on an efficiency-centric approach to student modeling, as described in section 5.2.

Two remaining research issues are salient to my work:

- Though student models are intended as tools to enable assessment and adaptive activity selection, the information they contain is useful to the student as a gauge of their progress and of the effectiveness of the effort they put into learning. Such feedback can act as a strong motivator, and many researchers have explored its effectiveness in what are called inspectable student models [84].

- Almost all student models are closely built around the tasks the student performs. This limits their applicability, and some researchers have proposed that student

models should be built independently of learning activities so that they can be used across multiple activities [75].

2.6.1.3 Modeling the Tutor

Human tutors encourage learning in a variety of ways. They may react in response to events and student actions; for example, by providing corrections and hints in response to student errors, encouragement following signs of frustration or lack of motivation, ways out of impossible situations caused by previous errors, or answers to direct questions and requests for help. Alternatively, they may proactively provide scaffolding to ease familiarization with new techniques or concepts, offer hints that prime students to pay attention to key elements of a task, make unprompted statements of support to reinforce learning, or engage in Socratic dialogue to lead a student into reaching their own conclusions [85]. Finally, when the student finishes an activity, the tutor is responsible for assessing their work and helping them decide what they should do next.

In doing so, tutors rely on a combination of general tutoring expertise, experience with individual students, subject matter understanding, and keen observation. In an intelligent tutoring system, observation and student knowledge are captured by the student model and subject matter by the domain model, while tutoring knowledge is handled by a third subsystem that maps from the learning context to particular pedagogical actions.

Tutoring systems vary widely in the level of support they provide – some systems seek to model the student's thought process in depth so that they can provide hints and guidance at every step of an activity, while others are satisfied with modeling the expected outcomes of different learning activities in order to select between them based on what the student most needs to focus on at any given time [86]. Step-by-step tutoring is powerful, but is often closely tied to specific problems or activities and requires significantly more advanced design and engineering.

Frequent and interactive interventions are often justified with the Interaction Hypothesis, which claims that tutoring effectiveness increases with interactivity [64]. Further work by

Van Lehn, however, suggests that tutors that intervene beyond providing step-by-step assistance by, for example, attempting to intervene in the student's decision making process, yield little to no increased learning, and may in some cases dissuade students [87]. Moreover, duBoulay and Luckin question the underlying assumption that the strategies known to be effective for human tutors will be effective for computerized tutors given their radically different affordances [85]. They suggest that since human tutors rely on social, vocal, and physical direction whereas computerized tutors tend to rely on text, graphics, and rapid detection and interactivity, different tutoring strategies are relevant, and researchers should embrace these rather than seeking to mimic human tutoring too closely.

Educational games also challenge the use of traditional tutoring techniques. In these, the goal of learning is properly shared with the goal of fun, the idea being that by blending the two, students are more motivated and open to learning. Consequently, learning is often in the background, concealed by gameplay and narrative, and so interventions that directly address learning can appear jarring. In general, tutoring interventions that unnecessarily interrupt play or otherwise detract from fun are likely to be viewed with hostility by players. There is little to no consensus on how tutoring should be applied within a game.

Activity selection, too, is likely to be challenging within a gaming context, as one of the defining characteristics of games is their voluntary nature – if there is no choice, players may quickly interpret them as work, greatly losing motivation. To preserve the illusion of choice, activity selection algorithms should offer learners with a number of options to choose from, perhaps including the reasoning behind each of them. This approach preserves the illusion of choice while still guiding learners towards productive activities.

In general, there are three approaches to activity selection [88]:

- Simple heuristics – select activities using arbitrary rules thought to approximate optimality. Rules might be based on previous activities (do X after Y), patterns of need (if subject A is troublesome, do activity Y), patterns of performance (if activity

X isn't improving performance, try activity Y), or general schemes (do X until performance exceeds some threshold then do Y).

- Decision theory with heuristics – choose activities based on the maximization of some utility function. Since the utility of each activity is not known, heuristics must be applied to generate utility values for each activity.

- Decision theory with data – gather information about changes in performance after each activity, use changes to estimate the effectiveness of each activity, and then use estimates to set utility values for utility maximization.

Activity selection is discussed in section 7.2, where an approach based on decision theory with data is applied.

2.7 Game Engines

Games vary widely in terms of their content and the things that players do as they play. Nonetheless, they are very similar in their construction. Most, if not all games utilize graphics to display content, physics to determine the interactions between in-game entities, audio to alert players and provide atmosphere, artificial intelligence to control opponents and the game environment, and scripting to create narrative and define in-game events. Other common areas of functionality include networking, memory management, social integration, and localization.

These similarities suggest that games can be built more efficiently if components developed for previous games can be re-used. This strategy is indeed used widely in modern game development, with re-usable components often bundled into full game development platforms known as game engines.

In addition to reducing effort through re-usability, game engines simplify development in two ways: by supporting cross-platform development (a game written once using a cross-platform engine is itself cross-platform) and by reducing the level of expert knowledge

required (game developers no longer need to understand advanced algorithms that are now embedded within the game engine).

2.7.1 Choosing a Game Engine

Since my research involves creating games and learning activities, I required an engine on which to build them. My needs are somewhat different to most game developers, in that I do not seek to produce marketable products, but rather to explore game generation methods along with a variety of designs.

In my research proposal, I defined selection criteria for choosing a game engine on which to base my work. I later refined these criteria, as follows:

- **Graphics**: The engine must support 3D graphics with a scene graph and support for basic picking, collision detection, transparency, and on-the-fly model loading. The engine should also support 2D GUI elements and other overlays, particle systems, and vertex and fragment shaders.
- **Physics**: Physics support is desirable, but not essential.
- **AI**: Artificial Intelligence support is not required.
- **Networking**: The engine must support asynchronous network calls to web services initiated by the game and engine; server driven network events are not required. Synchronous game networking is not required.
- **Audio**: Audio support is desirable, but not required.
- **Scripting**: The engine must support general purpose scripting, on-the-fly generation and insertion of audio and graphical elements, on-the-fly loading and invocation of scripts, and normal user interaction with both mouse and keyboard. The engine should support external third party libraries, allowing me to import missing functionality or write my own.
- **Web-based**: The engine must be web-based – games and activities must be embeddable within regular web pages. The engine should support events passed

both into and out of the game environment to the surrounding JavaScript environment.

The criteria that the engine be web-based immediately excludes many popular game engines – most game engines are designed for games that operate in native mode on a specific hardware platform; web-based games must run on the software platform of the web. I considered the following game engines:

- **X3D** is an ISO standard managed by Web3D consortium for defining interactive 3D scenes for the web [89]. Scenes written using X3D can, in principle, be viewed in any X3D browser, including some that embed X3D scenes within webpages viewed in a traditional web browser. X3D supports scripting with ECMAScript, and was successfully used in my preliminary work building interactive anatomy browsers using the FMA. Unfortunately, no X3D browser supports the full standard, and all browsers have their own quirks, meaning that complex applications must be targeted at a specific browser. X3D's scripting support is also fairly lightweight, and does not allow communication with external libraries or web services. Finally, X3D browsers are generally not well optimized, meaning that complex graphical scenes run very slowly.
- **Unity** is a popular game engine for PC, consoles, mobile devices, and the web. It incorporates a wide range of features including advanced graphics, physics, and full scripting using JavaScript and C#. Unity is a proprietary environment, and many advanced features require the professional edition, which costs around $1500. In addition to its role as a game engine, Unity provides a detailed authoring environment and asset management workflow. Unity is the most fully featured of the engines considered, but is expensive and proprietary, and while it operates on the web, is not primarily intended for it.
- **Flash** is a development environment for animations, games, and interactive activities on the web. Though its heyday has passed, Flash remains very popular and is used in a great many legacy applications. It provides support for 2D graphics and

animation, audio, and scripting in a proprietary language called ActionScript, but does not support 3D graphics.

- **WebGL** is a JavaScript API based on the Open GL ES standard that adds support to HTML5 for 3D graphics. It is defined as an open standard, but relies on browser manufacturers to implement it, much as they implement rendering for HTML itself. It runs natively within a normal browser environment and does not require the use of a plugin – this allows it to co-exist with HTML elements and events in a way that the other candidate technologies cannot: DIV elements can overlay the WebGL canvas and interaction with both HTML and WebGL objects is cleanly integrated. Furthermore, multiple interacting viewports can co-exist within the same browser window, allowing for sophisticated web-based applications. WebGL, offers no support for audio, networking, or scripting; this functionality can instead be taken up by the regular features of HTML 5. Finally, as a native component of the browser, WebGL can be built on by third party libraries to provide more advanced functionality.

From the above technologies, I selected WebGL as my preferred development platform. Though it is not a full game engine in itself, in conjunction with third party libraries such as three.js [90] and jQuery [91], and the features already included in HTML 5, it provided all of the functionality necessary for my work. Furthermore, it is as cleanly integrated with the web as it is possible to be, requiring no plugin or other proprietary component, and it is free.

2.8 Existing Games and Activities

Before proposing new games, it is important to understand what games and learning activities have come before and determine what can be learned from these. In addition to examining games, it is worthwhile to survey other online educational resources, as these will likely incorporate ideas for interactivity, visual communication style, and content structure that can be employed within games.

In this section, I present the method and results of a survey of the literature and public internet of resources that teach anatomy. Rather than attempt to catalogue these resources exhaustively, I will characterize them through a series of examples that illustrate particular design approaches. I will begin by addressing games, then move on to more general resources, including several that fall outside my area of consideration but that contain interesting ideas or suggest design approaches worth exploring. Finally, I will provide a brief discussion of some general patterns that can be seen in the collection.

2.8.1 Method - Survey

In 2012, I conducted a brief survey of games for teaching human adult gross anatomy with the assistance of an undergraduate research assistant, Thuy Duong. This survey did not follow a well-defined methodology, but resulted in the collection of 81 game-like learning activities that related to anatomy. In 2014, I conducted a more exhaustive survey with advice and support from Melissa Clarkson, a PhD student in the Department of Biomedical Informatics and Medical Education, using the following methods:

- **Search Engines:** Searches were run on the Google and Bing web search tools with terms "learning anatomy", "anatomy atlas", "anatomy game", and "anatomy education". URLs for resources were collected by manually selecting from search results until 5 pages had passed with no new resources being mentioned; on average, search results were explored to around 30 pages deep.
- **App Store search engines:** Searches were run on the Google Play and iTunes mobile application stores using the terms "learning anatomy", "anatomy atlas", "anatomy game" and "anatomy education". Apps were manually selected from the full list returned by the search engine. In addition, lateral searches were performed using the "related application" search results shown on individual app pages.
- **Literature review:** A manual search was performed through the abstracts of papers published in the journals Clinical Anatomy, The Anatomical Record, Medical Education, Medical Teacher, and Anatomical Sciences Education. Publicly available

resources described in the papers found were collected, while those that referred to resources not released to the public were examined for interest, but not collected for the survey.

- **Existing lists:** During the preceding methods of search, various lists of resources were uncovered. URLs and apps on these lists were collected. Lists that were consulted that are published in peer-reviewed journals include [92][93][94].

The 2014 survey yielded 2045 distinct URLs for resources and 909 URLs for mobile applications. Once collected, resource URLs were filtered to remove dead resources, duplicate resources, links to different parts of the same resource, obsolete and non-functional resources, resources that incorporated only trivial content, and irrelevant URLs such as resources that turned out not to concern adult human gross anatomy or those to vendors of animations and physical models. After filtering, 462 web resources, 309 Android resources, and 425 iOS resources remained. Notes on these resources were collected, and certain resources were selected for discussion here. A paper reporting on the full survey is planned for publication in the near future.

2.8.2 Games

Games set goals for players, then pose decisions or tasks that must be completed in order to achieve that goal. They are directed activities – the player does not need to determine what they should do, merely do the things the game asks them to. Decision making is constrained, attention is focused, and, typically, player skill is matched with challenge, creating conditions particularly well suited to invoking Flow, a highly engaging state of experience in learning is increased and motivation is particularly high [95]. As a result, games have long been of interest as a means of stimulating and encouraging learning [96][97]. Though today's educational game research focuses primarily on experiences between one student and one computer (or tablet), many interactive learning activities that have been employed to inspire and engage students throughout the ages, such as role-playing, problem solving, and even Socratic dialogue, can be thought of as games or at least possessing game-like qualities.

2.8.2.1 Early games

The earliest game mentioned in the literature is the Anatomy Ball Game, dating from 1951. In it, students play a close analogue of baseball where, instead of pitching and batting, players take turn answering questions from a list pre-approved by their teacher. The Anatomy Ball Game is thus an example of an existing game used as a scaffold on which to build a learning activity. Such games rely on the assumption that since students (presumably) enjoy the base game, they are likely to enjoy a learning activity built around it. For this reasoning to work, it is essential that the learning activity not interfere too greatly with the base game – as a result, such educational games are limited to relatively simple quiz or drill activities. The key benefit offered by games of this type is that they incorporate learning into social, narrative, and physical contexts that are more memorable and meaningful to students than regular classroom learning.

2.8.2.2 Quizzes

Quiz mechanics such as those demonstrated in the Anatomy Ball Game are very common in educational games, but may be structured in significantly more sophisticated ways. Rather than simply tallying the number of correct answers to determine victory, games may incorporate quiz mechanics into narrative or organize them such that different types of questions perform different roles in terms of game mechanics; for example, requiring one question in each category to achieve victory. This style of play is sufficiently attractive that it forms the basis of Trivial Pursuit [98], a successful commercial game designed primarily for entertainment. Much of the enjoyment players experience in playing it comes from the feeling of testing oneself in the context of a social experience; pub quizzes are another example. It is perhaps no surprise, then, that a similar game, Corpus Morphus [99], has been created using anatomical questions. In it, players receive victory points based on the difficulty of the questions they answer, and win once they exceed some threshold and have answered questions from each of the question groups corresponding to different areas of anatomy. Though Corpus Morphus was not evaluated in any formal laboratory studies, informal comparisons showed that average performance on tests

increased by around 5 percentage points once students got access to the game, with the largest performance increase accruing to those students who reported playing it most.

Computer-based quiz games are also common, and though they lack the social facet of tabletop games, they support more rapid play and tend to incorporate more visual questions. These games are diverse, and may use many types of question, including multi-choice [100], label placement on diagrams or 3D models [101], structure selection [102], structure placement [103], pin and structure identification [104], short answers, sentence completion [105], and flash cards [106]. Several libraries of quizzes exist online, including a large collection at the University of Minnesota targeted at college-level anatomy students [107][108] and several collections at general purpose quiz sites created by members of the general public [109][104]. These games are diverse, but tend to focus on the identification and location of anatomical structures in histological images, prosections, and anatomical sketches. Ben Crossett's Anatomy Arcade [102] (Figure 2) is a smaller collection, but is notable in that it incorporates a practice phase alongside a quiz with time-based scoring. Many online quizzes are presented directly as educational activities, but some, particularly those for younger learners, are wrapped in some narrative or game context. For example, Akl et al present a quiz game that teaches clinical practice guidelines based on the TV game show Jeopardy [110], while another uses the premise of a Frankenstein monster in need of repair [103].

As discussed in section 2.1, many game design theorists would not consider most basic quizzes to be games, in that they are not interactive systems. In a basic quiz, a player answers questions, but the quiz system does not change in response to their answer. By comparison, in a true game, a player's decisions evoke changes in the system of the game which in turn condition the environment in which the player makes their next decision.

2.8.2.3 Toys

Toys are defined as interactive systems that do not impose specific goals on the user, and are thus considered by many to not be games, despite the fact that they facilitate play. Toys such as SimCity [42] may be as sophisticated and complex as games, but they do

not create a sense of progress to motivate play, instead relying on free-form curiosity, exploration, and self-imposed goals. Toys may also feature in personal fantasy and narrative, such as the stories a child tells using their soft toys. Educational activities for young learners are often toys; Duckiedeck, for example, provide several toys for pre-school children in which body parts can be put together onto dolls to trigger funny animations and sounds [111] (Figure 3). More sophisticated interactive systems for adults are often referred to not as "toys" but as "simulations", "atlases", and "interactive environments", and are discussed in section 2.8.4. Nonetheless, their free-form interactive nature puts them in the same category.

2.8.2.4 Puzzles

Puzzles are essentially problem-solving activities wrapped in a game-like context. Classic word puzzles such as crosswords and word solve puzzles exist as somewhat more sophisticated knowledge tests than quizzes [112], but are relatively generic and generally limited to verbal questions. Other puzzles such as spot-the-difference, sliding tile [113], and jigsaw puzzles [102] (Figure 5) do not directly engage players with anatomical knowledge, but use it as background context for another activity.

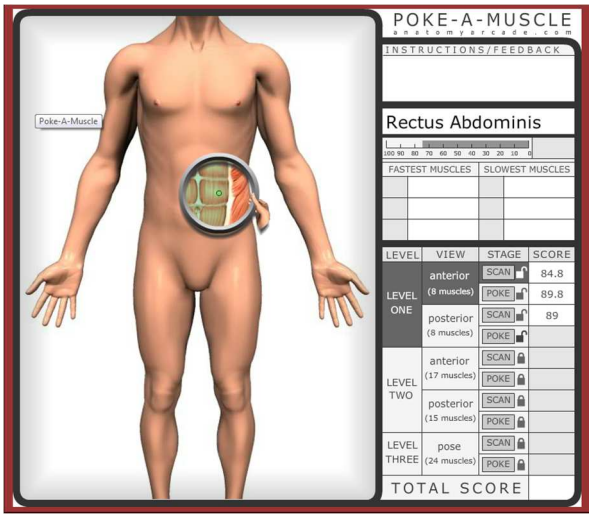


Figure 2 - Poke-a-Muscle; Anatomy Arcade [102]

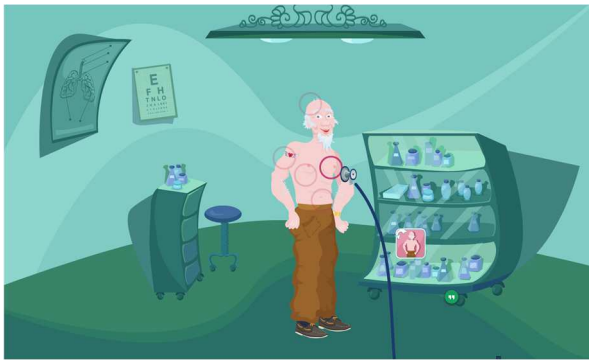


Figure 3 – Stethoscope; Duckiedeck [111]

Not all educational puzzles, however, have such a loose attachment to the material they purport to teach. Med School [114] (Figure 4), from the University College Cork School of Medicine, uses a narrative framework and a point-based skill purchase system to wrap a sequence of diagnostic puzzles. As the game progresses, the player learns about physiological processes and diagnostic techniques that are used in game to interpret symptoms and diagnose them against a pool of known diseases. As more and more disease possibilities and diagnostic techniques become available, players must shift from brute force application of known techniques to carefully following trails of evidence to diagnoses.

A similar game that employs diagnostic actions and narrative in play is the Oncology Game, a computerized board game in which players diagnose patients by "sending" them to specialists corresponding to different locations on the game board [115]. Though neither game has a strong emphasis on anatomy, an analogous game that relies on anatomical concepts seems possible; for example, players might be asked to deduce specific damage to nerves and blood vessels by particular symptoms.

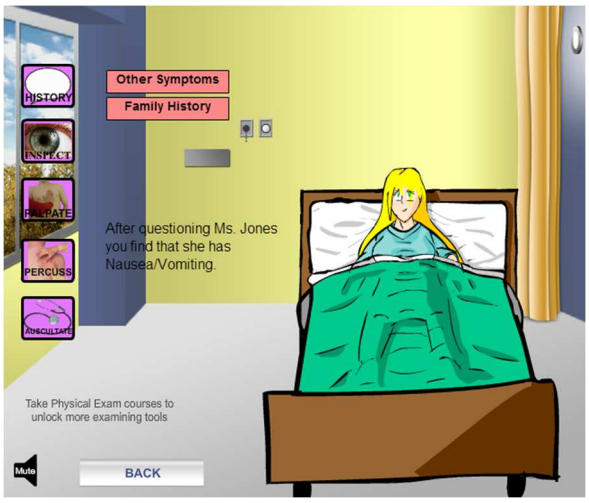


Figure 4 - Med School; University College Cork School of Medicine [114]

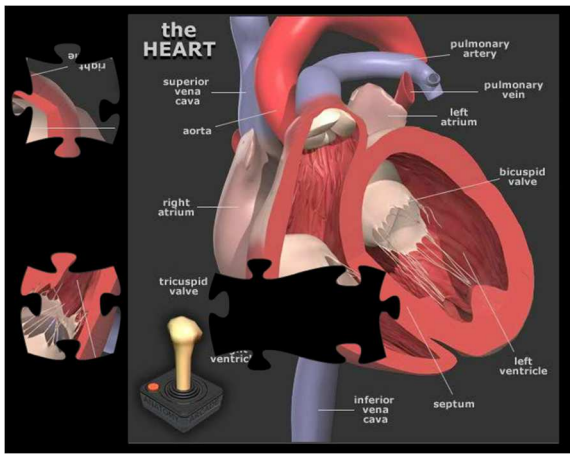


Figure 5 - Jigsaw puzzle; Anatomy Arcade [102]

2.8.2.5 Skill games

Related to puzzles but based on physical dexterity rather than memory or reasoning, skill games pose players with a straight-forward challenge that must be overcome through careful movement and physical skill. A classic example is the comedy board game Operation [116], in which players must extract oddments from a cardboard surgery patient using a pair of tweezers. Also intended as comedy, but with more potential for realism, Surgeon Simulator 2013 [117] asks players to successfully complete surgical procedures using a limited and intentionally difficult set of inputs. Finally, the Trauma Center [118] series of games, originating in Japan, poses diagnostic puzzles alongside skill puzzles in a narrative context involving emergency medicine, bioterrorism, and forensics. Unlike Surgeon Simulator and Operation, Trauma Center takes itself seriously, and, through its use of the Wii Remote, demonstrates that interaction through kinesthetic input devices can create a compelling medical simulation that could be leveraged as a platform for teaching anatomy, basic surgical concepts and procedure, and to some extent, clinical practice. Like puzzles, skill games engage players through their desire to successfully overcome challenges. Unfortunately, however, they tend not to require that students actively process the medical content they incorporate. A more serious game along the lines of Trauma Center could indeed do this, but for now, skill games are relatively limited.

2.8.2.6 Tutorial games

Tutorial games take the general idea of physical skill challenges and applies it to tutorial activities that approximate surgical procedures, asking students to perform various surgical actions on virtual patients. Several examples exist, including hip resurfacing [119], knee replacement [120] (Figure 6), and heart transplant surgery [121][122], as well as cancer diagnosis and treatment [123]. These activities do not attempt to recreate the minutiae and anatomical detail of real surgery, but instead aim to familiarize the player with the steps and anatomy involved. This style of play has been employed in games for entertainment, though with less commitment to realism. Games such as Dark Cut [124] (Figure 7), which focuses on wartime medicine during the American Civil War, and Amateur

Surgeon [125], a comedy game based around back-alley amateur surgery (complete with pizza cutters instead of scalpels) engage players by adding more game-like mechanics such as scoring, timed performance, bonuses, and random events, such as bleeds and sudden drops in vital signs. Though these morbidly humorous games do little to teach real anatomy or surgery, the design approach they employ could easily be adopted for more serious activities.

2.8.2.7 Time-Management games

Time-based mechanics show up frequently in skill and puzzle games, but also form the basis for a genre of their own: time-management games. In games of this nature, players are presented with problems that must be solved by completing a series of small tasks. These tasks are normally not individually challenging but are made so by forcing the player to work on tasks from multiple problems simultaneously and under time pressure. Time-management games often employ simplified real-world contexts such as hospital management [126][127], restaurant management [128], or even a chocolate factory [129][130]. Time-management games are very common, and at least ten have been published with a general medical theme. Their strength is in the way they reinforce knowledge by forcing players to act reflexively. Such knowledge can be task oriented, spatial, visual, or verbal. Various time-management games that focus on medical knowledge of this nature can easily be imagined; an example concept is described in section 6.2.3.

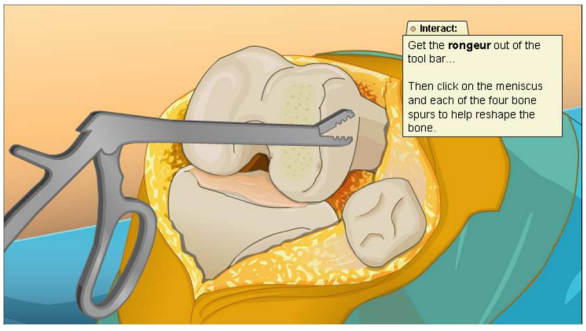


Figure 6 - Virtual Knee Surgery; Edheads [120]

Figure 7 - Dark Cut 2; Armor Games [124]

2.8.2.8 Egocentric games – role-playing and simulation.

Most of the games presented thus far are exocentric in nature; that is, they do not represent the player within the game world, instead allowing them to act from the outside by issuing commands that cause changes to the game system. Egocentric games put players inside the game world in some way, usually by representing with some in-game avatar or character. Egocentric games tend to include more personal narrative and action, and prominent genres include role-playing games, first and third person action games, and many adventure games. Simulation of the game world is also prominent in games of this nature. Egocentric games focused on anatomy might pose players as mobile body parts, such as white blood cells, or artificial constructs such as nano-robots or miniaturized capsules, as in the Magic School Bus books [131]. In the Nervous System game [132], a real-world example of a simulation game, children in elementary and middle school physically act out the role of nerves passing messages around a body drawn out on the floor. Although this activity is less a game than a playful re-enactment, it has been successfully used to teach neural anatomy, sensory and motor function, reflexes, and spinal cord injuries by encouraging students to actively process the knowledge received as part of classroom learning by using it to guide their actions in play. Similarly, in a Neverwinter Nights [133] module designed as part of a course on educational psychology for undergraduate students, players interact with and observe "memory imps" moving through a schematic environment based on Mayer's model of memory formation [4][15].

2.8.2.9 Virtual worlds

Despite great advances in the technology behind computer games over the last decade, most educational games for medicine rely on web-based imagery and other simple 2D environments such as Adobe Flash. While the technical expertise required to work with advanced game engines is surely an impediment, the advantages they possess in terms of simulation and graphical quality makes them worthy of further exploration [134]. Thus far, their primary application has been in surgical simulation, where realism is of primary concern [135][136], and while researchers have begun to explore their application for

anatomy education, no viable systems have yet been made public. One interesting example employs egocentric viewpoints in exploring the human body by allowing students to control an avatar climbing over and within various abdominal organs [137]. Unfortunately, however, this game does not appear to have moved beyond the prototype stage and included little anatomical detail either in the form of detailed models, labels, or linkages with other educational resources. Another example, the Cranial Nerve Skywalk [138], provides an explorable model of the skull and associated muscles and nerves in the shared virtual world Second Life [139] (Figure 8). Shared virtual worlds are of particular interest for education, as they support easy construction of educational content in a persistent 3D environment with support for collaborative play and learning and potential for integration with learning management systems. Technology changes fast, however - in the last two years, for example, virtual reality systems have vaulted back into prominence with cheap headsets such as the Oculus Rift [140] and interaction devices such as the Razer Hydra [141] becoming available. These offer the possibility of fully immersive educational environments which research suggests will greatly enhance meaningful experience and knowledge construction [142].

A special form of egocentric view, augmented reality, involves the layering of computer-generated imagery over images of the real world seen from the perspective of the user. In a typical augmented reality system, a user wears a head-mounted display with an attached camera and sensors that track the position and rotation of their head with respect to the environment. Input from these sensors is used to generate imagery of virtual objects which are then drawn over images from the camera so that the user experiences the world as containing both real and virtual objects. Augmented reality offers interesting potential as a closer analogue to dissection than images shown on a computer screen, as early researchers [143][144][145] quickly realized. Though these systems did not leave the laboratory, the general idea of allowing students and clinicians to visualize anatomy in situ over the human body remains compelling, and research into augmented reality systems for teaching anatomy continues [146], particularly as head-mounted display

technology becomes cheaper and more accessible. In one, more recent, project, researchers using the Virtual Human Dissector system have explored the projection of general anatomical models onto the body of students during Living Anatomy sections [147].

2.8.2.10 Engaging Educational Games – an Exemplar

Though the games presented thus far are diverse and, in some cases, have been proven to be effective, none of them approach the true complexity and engaging power of top tier games produced for entertainment. Unfortunately, this is all too commonly true, as educators often lack game design experience, technical skills, and the funds required to build truly compelling games. One outstanding exception to this rule is Cellcraft [148] (Figure 9), a game designed to teach high-school cell biology. In it, players control a cell, complete with organelles and cell machinery, as it gathers resources, fights off pathogens, and evolves in the context of the game's narrative. In addition to being fun and engaging, it treats the science being taught as a functioning, interactive system, then leverages that interactivity to build game mechanics. Educational content is placed in the foreground, and, where necessary, tie-ins to more pedagogical resources are provided. For a more detailed review, see my blog [149].

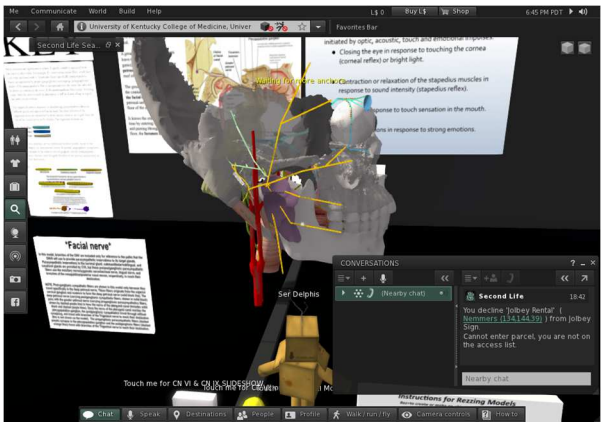


Figure 8 - Cranial Nerve Skywalk (Second Life)

Figure 9 - Cellcraft

2.8.3 Game classification and descriptive vocabulary

Cellcraft, with its attempt to integrate educational material into game mechanics, suggests a means of classifying educational games based on the level of integration between the things the player is intended to learn and the things they actually do when they play. I place educational games into three categories based on this criteria:

- Games where the knowledge to be learned is a necessary part of the decisions the player must take during play, are considered to employ **knowledge in the foreground**.
- Games where knowledge exists in the game world and informs the player's understanding of that world and of the meaning of game entities, but is not necessary to play, are considered to employ **knowledge in the background**. Such games may place educational material in box text that explains events in the game, but do not require that the player exhibit or act upon knowledge during play.
- Games where educational content does not exist in the game world except as "window-dressing" are considered to employ **knowledge as context**. Such games can serve to enthuse students about a subject, potentially encouraging them to learn in other ways, but they do nothing to teach themselves.

Many other methods exist to classify games. Rather than attempting to exhaustively list these here, I direct interested readers to the following excellent resources: Rules of Play [14] by Salen & Zimmerman, The Art of Game Design [150] by Jesse Schell, and Patterns in Game Design [151] by Björk and Holopainen. Classification schemes are primarily useful as they provide a vocabulary for discussing and critiquing different game designs. Most are at least somewhat subjective in the way they are defined and applied, though this generally does not detract from their usefulness. Common schemes include genre of play, the ludology-narrative continuum [152], and the facets of experience they employ to engage players [153].

2.8.4 Online Resources

Online anatomy resources come in a variety of forms other than games, including atlases, textbook companion sites, image libraries, simulators, and more. These resources do not offer the same motivational and learning-enhancing characteristics as educational games, but many are nonetheless extremely useful for education. In particular, these resources often contain far more detailed and thorough representations of anatomy than games.

2.8.4.1 Atlases and Image-Based Education

To most, the word anatomy brings to mind detailed diagrams of the human body, sans skin, with bones, musculature, and organs labeled, as in the fine engravings of classics such as Gray's Anatomy [154] (Figure 10). This is not inappropriate, as a significant portion of anatomical knowledge revolves around the visual characteristics, spatial relationships, and organization of bodily structures, and images and models have historically played a large role in its teaching [9]. While hands-on experience with cadavers plays an indispensable role in conveying the structure, texture, and sheer physicality of the human body, imagery plays a key supporting role by providing idealized and schematic views impossible to obtain perfectly in each cadaver, and furthermore, allowing students to take these images away with them for study [155].

2.8.4.1.1 Image Library Atlases

Image library atlases consist of collections of anatomical sketches, diagrams, and photos of cadavers, labelled and hyperlinked to create a navigable and useful whole. Atlases of this type have become less common in favor of interactive atlases, and the most notable example still available is the Anatomy Atlases [157] collection. In addition to presenting a whole body image atlas based on sketches, it includes a cross-sectional atlas, a bone-box atlas, and an extremely detailed reference guide to anatomical variation.

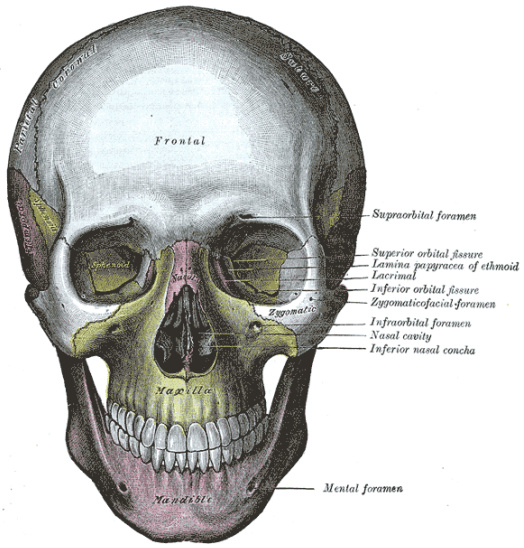


Figure 10 - Anatomy as depicted in Gray's Anatomy [154]

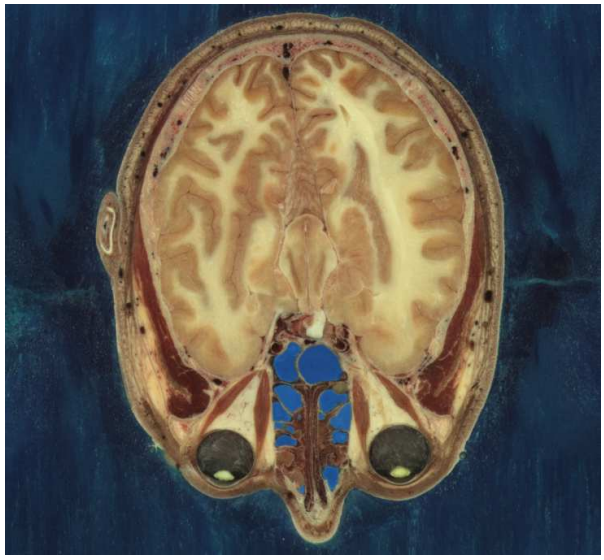


Figure 11 - Anatomy as depicted in Visible Human [156]

Cross sectional atlases are a special case of image library atlas where images are slices through the body, typically along one of the three anatomical planes. Such images may be obtained using volumetric imaging methodologies such as MRI or CT, or they may be obtained from finely slice cadavers, as in the Visible Human [156] (Figure 11). Cross-sectional atlases may present slices sequentially or images may be presented using more sophisticated interfaces showing multiple slices on different planes such as in the Harvard Medical School's Whole Brain Atlas [158].

Image library atlases occasionally, but rarely, incorporate schematic images of the human body where extraneous detail is removed so that only the features of interest are shown. More abstract images even eliminate concern for exact shape and spatial proportion, much as the London Underground Map does for that city. No good examples exist online using images of this type, but significant examples can be found in physical atlases, particularly those for neuroanatomy, such as the Thieme Atlas of the Head and NeuroAnatomy [159].

Though simple in concept, image library atlases can be made more sophisticated by including interactive layers. For example, O'Byrne et al [6] presented a web-based system that combined photos of dissected cadavers and schematic images with simple interaction in the form of shaded toggle-able layers, mouse-over events, and fade-through effects in

an open-ended exploratory environment which was well-adopted and used by students, though no significant improvement in learning was seen. Results of this nature, where a resource is found to be useful but does not measurably improve learning, are common, and their implications are discussed in section 2.8.5

2.8.4.1.2 Computer-generated Atlases

Modern computer generated imagery have several advantages over traditional sketches, from ease of production and customization, to interactivity and support for the third dimension. The potential value of such imagery has long been acknowledged by medical educators [160] and, over the years, a great deal of research and development has been applied to the creation of computer assisted learning tools.

In recent years, a variety of vendors have produced interactive 3D atlases of the human body, including Netter's 3D Interactive Anatomy [161], the BioDigital Human [162], the Zygote Body Browser [115], and several more. These atlases employ a variety of technologies to create 3D images of the human body on both the web and mobile platforms (both Android and iOS), with scene construction based on selecting structures of interest from a list. As well as providing a relatively straight forward interface to explore the structure and layout of the human body, several of these resources include content authoring and publication tools that allow educators to create custom scenes for incorporation in their own web resources. 3D navigation and scene construction is overkill for some users, and studies have shown that including too much flexibility in presenting 3D structures can inhibit the learning of some users, particularly those with low spatial ability, by distracting or confusing them [163][164]. Systems such as the Inner Body [116] remove navigational complexity by limiting users to key views where certain anatomical features are presented optimally.

2.8.4.1.3 Full reference atlases

Some atlas resources go further, attempting to become visually indexed reference guides to the whole of human anatomy. For example, both Inner Body and Netter's 3D Interactive Anatomy incorporate textual descriptions accessible from the atlas that contain further

information about detailed features, connectivity, and the physiological implications of anatomical structure. Healthline [114], though now technically obsolete, went further, providing links from its atlas to additional reading on the physiology and key pathologies of those structures.

2.8.4.2 Textbook Resources

Print textbooks are now commonly sold with access to an accompanying textbook support site. These sites contain auxiliary and extension material not included with the book itself. They are typically not exhaustive, in that much subject matter is included in the book but not on the site, and so are not standalone resources. Typical auxiliary material of interest includes animations and videos explaining dynamic or complex processes, such as developmental anatomy, and interactive models of key concepts.

Fully digital text books are not yet common, but efforts to create them are ongoing. Digital books afford searchability and interactivity, and can often be made free or cheap due to the virtually zero marginal cost of their distribution. The most thorough digital text book for anatomy presently available is the Anatomy & Physiology text distributed by Primal Online Learning [165]. It exemplifies many best practices in the presentation of pedagogical material by integrating images and text, presenting material in manageable small chunks, clearly indicating how students should proceed through the material, and including frequent testing and knowledge anchoring activities alongside the text. It is, unfortunately, not cheap. Free resources include the OpenStax Anatomy & Physiology text [166], which is presented as a digitally available print book, Wes Norman's Anatomy Lesson [167], an early anatomy hypertext, and Wikipedia, whose articles, while peer-contributed and peer-edited, are incorporated into a wide variety of other anatomy resources where descriptive text is required.

2.8.4.3 Videos and Interactive lectures

Related to online textbooks by their direct pedagogical intent are libraries of educational videos and animated tutorials. YouTube alone contains thousands of videos illustrating different anatomical regions or concepts, though the quality of these is mixed, to say the

least. Several collections are available, with the most complete probably being the Instant Anatomy lectures by Robert Whitaker at Cambridge University [168].

More sophisticated than video series alone are the Massively Open Online Courses offered through Coursera [169], MIT OpenCourseWare [170], FutureLearn [171], and the Khan Academy [172]. These revolve around video lectures produced by faculty at various universities coupled with interactive activities, quizzes and assignments, and support forums. Students can either participate synchronously as the course is released with lectures and quizzes released with completion deadlines roughly similar to a normal university course, or asynchronously at their own pace. Completion of a course may grant some accreditation, depending on the venue. Massively Open Online Courses (MOOCs, for short) are regarded by many as a major revolution in the way university education is provided, and are evolving quickly. Many of the games and activities reviewed and proposed in this dissertation may eventually find their way into future MOOCs.

2.8.4.4 General Interest resources

Museums and art galleries have long been places for the general public to see skeletons and other preserved artifacts of the human body such as, for example, the plastinated bodies of the Body Worlds exhibits [173]. Similarly, television documentaries make anatomy more widely accessible withy high quality visualizations of the human body presented alongside descriptive narrative; notable examples from the BBC include the series "The Human Body" [174] and "Inside The Human Body" [175], from 1998 and 2011, respectively. In addition to teaching viewers about the structure and function of the body, such shows and exhibits often place anatomy into contexts that are more meaningful to non-specialists, with focuses on sports and physical activity, common health problems, and our attitudes to mortality.

Today, both museum exhibits and television show are often paired with online exhibits that often contain interactive components. Examples include the e-Skeletons [176] project and resources from both PBS Nova [177] and the BBC [178].

A major general interest for anatomy is in resources for practitioners of various physical disciplines, including yoga [179], dance [180], weight lifting [181], general sports [182], and even voice training [183]. Such resources present anatomical material in the context of the physical actions required in those disciplines, discussing the implication of anatomy for practice, common problems, and how to avoid injury. Though not obviously relevant to medical education, learning anatomy in applied contexts such as these has proven to be both motivating and effective for medical students, as evidenced by studies employing both yoga [184] and martial arts [185] to teach musculoskeletal anatomy.

Another way in which members of the general public come into contact with anatomical knowledge is in the clinic during medical diagnosis and treatment. Informed consent relies on patients being able to understand procedures that are to be performed upon them, and patients who understand the mechanisms and implications of disease are arguably more likely to behave wisely. Resources such as WebMD [186] About.com [187] and the Mayo Clinic's index of Diseases and Conditions [188] offer information concerning a variety of diseases, often alongside articles on the relevant anatomy. More specialized resources provide information on specific conditions, with extant examples for osteoporosis [189], joint injuries [190], and many more.

One final class of general interest resource concerning anatomy are those that present anatomy for artists with an interest in representing the human form. These focus on the appearance and layout of anatomy both in anatomical position and in a variety of active positions in order to assist artists in creating realistic images and sculptures.

2.8.5 General notes

While conducting this survey, three general issues stood out as requiring additional discussion and clarification.

Firstly, the resources available have changed over time not only in type but in who makes them. Older resources, including those still available online and those documented in previous surveys [92] were usually created by expert practitioners of anatomy either

individually or within university teams, whereas recent resources tend to be produced by private companies and software developers. This suggests that new groups have become interested in how anatomy is visualized and presented, perhaps due to new commercial opportunities or increased interest in access to knowledge about the body from the general public.

Secondly, many games and learning resources for younger audiences teach anatomy that is incorrect in some way. This is presumably to manage detail and complexity that could overwhelm younger learners. This seems acceptable, provided the misconceptions and simplifications taught in early stages are deliberate and corrected during later stages. Generally, games for young children focus on naming the visible parts of the body such as the limbs and facial features, whereas games for school students focus on discrete elements of the body such as bones, muscles, and individual organs. Detailed anatomical features such as spaces, fascia, connectivity, and cross-sectional anatomy are addressed only in university-level or specialist courses.

Thirdly, when studies that evaluate online resources for teaching anatomy are published, they frequently find little or no increase in student performance. Though this suggests failure if such resources are thought of as aiming only to make students learn better, this view is misleading. The advantages of different instructional technologies lie primarily in the affordances they provide students and teachers rather than in their necessarily providing a measurable improvement in learning, provided that they don't degrade it [191]. By offering means of learning that avoid the costs and emotional challenges of dissection or prosection, that allow students to see the body in ways not previously possible, that allow learning in non-traditional formats and environments, and that engage students who would otherwise have trouble learning, online resources both enrich and enlarge opportunities to learn, leading to improvement in anatomical literacy across the population at large.

Chapter 3. Goals & Strategy

In chapter 1, I presented anatomy as a field of knowledge that is uniquely challenging yet essential to practitioners of medicine and others whose work revolves around the human body. I described the current state of the art in anatomy education and suggested that computer-based learning resources offer a useful supplement to traditional methods that can both improve the learning experience and help manage its cost: by enhancing the cadaver dissection experience for dental and medical students, by better making up for the absence of that experience with students for whom it is to expensive and time consuming, and by helping practitioners maintain their knowledge later in life. To support this assertion, I discussed the specific advantages of computer-based tools for teaching anatomy and reviewed several challenges in their development and deployment. I then introduced formal knowledge representations of anatomy, specifically the Foundational Model of Anatomy, and argued for a vision of ontology-driven education where the knowledge embedded in such constructs is used to improve the quality and function of computer-based learning resources while simplifying their construction.

In chapter 2, I sought to address a series of supporting questions by presenting and discussing previous research and other background material. Some questions were philosophical, dealing with the nature of anatomy and of games, while others were technical, focusing on the technologies available for use in this project. Chapter 2 ended by presenting the results of a survey of existing resources for learning anatomy, including discussion of their various characteristics and means for classifying and understanding them.

In this chapter, I will articulate the vision outlined in chapter 1 into a specific research strategy that I have undertaken in my work. I will discuss issues of scope, as well as trade-offs and assumptions.

3.1 Scope

I define the scope of my inquiry as being the investigation of **ontology-driven methods** for teaching **adult human gross anatomy** using **games and game-like learning activities**.

The meaning of **ontology-driven methods** is a major part of my thesis, and is developed initially in section 2.1 and reinforced throughout the remainder of this dissertation. As discussed in section 2.1, I focus on **adult human gross anatomy** as being a well-defined and central subset of the overall study of anatomy for which the FMA, my primary ontological tool, is optimized. Finally, as discussed in section 2.1, I use a fairly loose definition of the word "**game**" that encompasses activities that do not fully meet the term's definition but that share an interactive and engaging nature with activities that do.

3.1.1 Assumptions

In order to justify my work and in order that I might focus more narrowly, I invoke the following assumptions:

- Firstly, I assume that anatomy can be learned from images and 3D models rather than direct experience of the human body, without the introduction of major misconceptions or confusion. I do not pit image-based learning against contemporary approaches such as lectures, dissection labs and living anatomy, but assume that images are useful learning tools worthy of exploration in their own right. I accept this assumption because images have been used significantly in teaching anatomy since at least the mid-19th century, with the publication of Gray's Anatomy [154], and I have found no evidence to suggest that such problems exist.
- Secondly, I assume that computerized learning resources consisting of text and images are equivalent to printed text and images except in their different affordances. That is, I assume that text on a computer screen is as effective a learning resource as text in a text book. As mentioned, these two resource types have different affordances: books have a physical heft that some claim makes them

easier to read, while digital devices have screens that some find tiresome to look at for long periods of time. Digital resources, however, benefit from improved navigation through aids such as searching and hypertext. These affordances suggest different modes of usage of either medium, but do not make one inherently better than the other.

- Thirdly, I assume that in most circumstances involving gross anatomy, images constructed using 3D models can be made as effective at conveying information as images constructed by hand using pen and ink or other drawing tools. I note that this claim is contingent on the availability of 3D surface models but assert that this is a limitation of the content, not of the medium. I furthermore note that the addition of scene view controls such as view and rotation to 3D scenes is sometimes but not always helpful, depending on application needs and the spatial ability of a given student [163].

- Fourthly, I accept as given the claim that games and game-like learning activities are capable of motivating students and stimulating learning in at least some circumstances. I refer to several literature reviews and reports to support this claim [192][193][194][195] but do not attempt to justify it any further. As an aside, I note that much of my discussion of prior work in section 2.8 addresses the psychological and educational strengths of various types of activity, and that the discussion of the games I have designed in Chapter 6 addresses these same issues. I observe that games have a proven track record as engaging activities in and of themselves that closely focus attention over a period of time and assert that the key question in their usage is not whether they are effective but how best to embed educational material so that effective learning is stimulated without undermining the game experience and its motivational side effects.

- Finally, I assume and assert that it is not necessary for new methods of teaching to exceed traditional learning activities on measures of retention provided that they have affordances that complement existing methods, for example by reaching

students who are left out by traditional education. My work does not seek to replace existing methods of teaching, merely to supplement them.

Through these assumptions, I assert that games and game-like learning activities have sufficient potential for education that they are worth studying. By taking these assumptions as given, I am intentionally limiting the scope of my research by removing the need to justify them further, except as needed to evaluate the tools and systems I produce.

3.1.2 Audience

Though most people think of anatomy as a subject that is taught as part of medical, dental, and nursing programs at either an undergraduate or graduate level, most of us learn anatomy at some point, either casually and formally as part of school science and sexual education programs. Furthermore, when faced with medical difficulties, we must learn anatomy in order to understand medical procedures that might be applied to us. The primary difference between these circumstances is the level of detail; that is, the level of granularity at which anatomical structures are described, and the types of relationships between them.

I have chosen not to focus solely on any one of these groups, instead considering anatomy education in general, though I have not spent any time on tools for very young children. That said, since the level of detail required for high school students is lower than that required for medical students, my prototypes, which are limited by available 3D models, will most likely be more suitable for students at that level. Model availability is the main limiting factor to supporting higher level students, and as additional model sets become available the methods and designs I propose can be extended to support more advanced students.

3.2 Research Goals and Strategy

The Foundational Model of Anatomy provides a detailed and computable representation of knowledge about human anatomy. In its depth and breadth, it arguably contains more

information about the body's structure than even the most expert of anatomy teachers. It offers a powerful example of how disparate knowledge, from many domain experts, can be concentrated into a single representation that can be queried to answer questions about the domain, leveraged to give meaning to data sets through its controlled terminology, or used as a foundation to build new representations of some subset.

The central theme of my work is the exploration of the extent to which such representations can support the creation of educational tools, activities and games. I claim and hope to show that ontologies such as the FMA can be leveraged to address many existing constraints and challenges to their construction and open up new possibilities for design. Though my examples and efforts all focus on the domain of anatomy, I believe that the lessons learned from my work can apply to any domain of learning for which an ontology similar to the FMA is available.

3.2.1 Original goals

In my original proposal, I made the following two claims:

- That scene generation based on queries across an ontology such as the FMA could be used to automate the creation of customized content for educational games and learning activities.
- That ontologies such as the FMA could be used as the basis of tutoring tools within educational games and learning activities.

To investigate these claims, I proposed the construction of a scene generation framework based on the FMA, a student model built around a Bayesian network derived using heuristics applied across the FMA, and a method for automated activity selection. As a parallel activity, I also proposed a structured design process that would generate a collection of game and activity concepts that could be implemented using the tools constructed.

3.2.2 Final Research questions

Since writing that proposal, I have refined and reformulated my original claims into three research questions that focus less on whether it is possible to use an ontology to address challenges in creating learning tools and more on how this is best done:

A. How can an ontology be used to support the creation of content for learning tools and activities?

B. How can an ontology be used to support tutoring?

C. How can games and learning activities be designed to take advantage of an ontology-driven educational platform?

Questions A and B are fundamentally technical questions in that they ask how an ontology-driven educational platform might be built. To answer these, I have developed a range of prototypes that apply the FMA to achieve the specific functionality necessary to support games and learning activities. Question C, on the other hand, is a design question that I have chosen to address by characterizing the design space for games and learning activities that take advantage of an ontology-driven educational platform similar to the one I have developed.

3.2.3 Caveats

I have recognized since early on that the research goals I have laid out are broad and ambitious. In order to increase my chance of producing meaningful contributions, I have sought to constrain my ambition somewhat with the following caveats:

- I do not suggest that my automated tools for student modeling and tutoring will in any way be sufficient to replace human tutors. Though this is the long term ambition of the entire field of Intelligent Tutoring Systems, I am under no delusion that I will accomplish this in my work. My goal is instead to explore the role that domain ontologies such as the FMA can play in augmenting and simplifying the construction of educational tools and activities.

- I have no desire to replace existing methods of teaching anatomy such as dissection and prosection with solely computer-based ones. The role of dissection is a hotly debated topic within anatomy education, and despite having participated in a dissection class, I recognize that I do not have the expertise to weigh in substantially on this argument. Instead, I hope to facilitate the creation of educational tools that can supplement dissection and living anatomy by, for example, making it easier for students to learn at home, facilitating memorization, and supporting students in retaining and reinforcing their knowledge after they have graduated. I also hope that computer-based solutions might be of value to students and members of the general public who do not have access to the dissection experience.
- I do not seek to design the ideal game for teaching anatomy, nor do I believe that such a beast could exist. Anatomy is too varied a subject, as are the methods students rely on to learn. No game could address all equally. Neither do I seek to produce polished game artifacts that are ready for deployment and distribution. My intent, rather, is to explore and document the design space and propose directions and possibilities that might be taken up by future researchers and developers.

3.3 Dissertation structure.

The remainder of this dissertation is structured as shown in Table 1.

Table 1 - Discussion of hypotheses

Chapter	Method of Investigation
4	Question A – Ontology-driven content generation
5	Question B – Ontology-driven tutoring tools
6	Question C – Designing learning activities for anatomy
7	Sample learning activities
8	Conclusion

Chapters 4 through 6 address each research question in turn, with full details of methodology, outcomes, and reflection on how those outcomes answer the questions proposed. Chapter 7 integrates these results into a pair of sample learning activities built on the software tools presented in the preceding chapters. Finally, chapter 8 summarizes my findings, suggests future research, and offers concluding thoughts.

Chapter 4. Ontology-driven content generation

The Foundational Model of Anatomy is a deep, computable representation of knowledge about the human body that provides a strong foundation on which to build educational applications. In particular, it can be used as the basis for content generation tools that transform high-level descriptions, such as "the nerves of the left arm", into scenes and descriptors that show or describe the necessary structures. Such tools can, in turn, be used within educational applications to generate problems, activities, and visualizations. The FMA supports content generation in two ways.

- Firstly, we can use it to determine which structures ought to be included in a piece of content. If we require a body assembly puzzle (see section 6.2.8) showing the bones of the skull, the muscles of expression, and the nerves derived from cranial nerve V, we can, by running queries over the FMA, determine which structures are necessary.

- Secondly, we can use it to us determine how structures should be displayed. In a 3D scene, we might wish to use traditional anatomical coloring, with crimson muscles, red arteries, blue veins, and green lymph vessels. Alternatively, we might wish to label structures according to their vascular supply, the nerves they connect to, or some other anatomical attribute. We might even wish to label them using external research or clinical data. Depending on the labeling scheme required, the FMA either provides or facilitates access to information that can be used to determine how each structure should be displayed.

- Thirdly, we can use it to provide symbolic content such as labels and facts that can be used to generate textual descriptions, questions, game clues, and so forth.

This idea of using the FMA to create anatomical content has been around for some time, with early expression in the Digital Anatomist project [196] and, more recently, in the Biolucida and Intelligent Virtual Cadaver projects [197][198]. My work builds on these earlier initiatives by contributing the Anatomy Engine, a suite of web-based content generation tools along with applications that illustrate their potential. The key goal of this effort is to automate the process of creating 3D anatomical content as much as possible in order to reduce the time and expertise required to produce visualizations; enable interactive scenes such as atlases, tutorials, and browsers; and support problem and activity generation within educational games.

This chapter presents the Anatomy Engine, a software framework that enables the construction of applications that rely on 3D content generated from the FMA. Four end-user software applications are presented to illustrate different ways that it may be used:

- The Anatomy Viewer Application, or AVA; a 3D anatomy viewer that can be deployed either standalone, in its own webpage, or embedded, within any arbitrary web page, such as online course materials and reference documents.
- The Scene Builder; an ontology-driven authoring tool that allows users to assemble anatomical scenes by selecting content with queries over the FMA and supporting manual tools.
- The Data Visualizer; a tool for interactively visualizing essentially any anatomically indexed data available online
- The Anatomy Explorer; a visual frontend for the Foundational Model of Anatomy itself that allows users to research and answer questions, visualize and validate the ontology, and create content interactively without manually running queries.

Users interact with the Anatomy Engine and all four of the applications presented herein entirely through their web browser. This is advantageous for several reasons: firstly, almost all users are familiar with the web, reducing the learning required to use the software. Secondly, the web is largely platform agnostic, and users of virtually any device

capable of browsing the web can, in theory use these applications if support for WebGL has been enabled. Thirdly, no marginal installation or deployment effort is required; once the core Anatomy Engine is installed on a server, any user may log on and use the system. Finally, maintenance, upgrades, and content improvements need only be performed once on a single central server to become immediately available to users, allowing constant refinement.

The remainder of this chapter is laid out as follows:

- Section 0 gives an introduction to all of the software components, with discussion of their general structure and features;
- Section 4.2 discusses the Anatomy Engine in detail, including implementation details, open challenges, and further potential development;
- Sections 4.3 to 4.6 present each of the applications in depth, including usage notes, implementation details, and planned extensions and enhancements;
- Section 4.7 describes the method and results of a study employing expert review to validate the usefulness of the applications and, by proxy, the framework. Results include suggestions and ideas inspired by study participants for extending and enhancing both the applications and the framework;
- Section 4.8 concludes by summarizing this chapter's answer to research question A along with a list of the major open challenges and opportunities for future work presented throughout.

4.1 Overview

The Anatomy Engine and the applications built upon it have several features in common:

- All are constructed as client-server web applications. That is, users interact with the software via a web browser while state is preserved between user sessions on a server.
- All utilize the same software technologies. Client side code, primarily display and interaction logic, is written in JavaScript using WebGL and the lodash, jQuery, and three.js libraries. Server side code, primarily data management logic, is written in Java using the Spring MVC, myBatis, and Hibernate Validation libraries, and running within Tomcat 7. Data storage is handled by a postgreSQL database.
- All run within the same software context. That is, the code for the framework and all four applications is deployed within a single web application archive. This is for ease of development, and is not required. This is important, because it would greatly inconvenience third party developers and limit the usage of the framework if applications had to be deployed alongside the framework on the same server. As it is, third party applications interact with the framework using web services, meaning that they can be deployed anywhere and they can even operate solely as client side applications. Furthermore, the same server can be used by multiple applications.

4.1.1 Anatomy Engine

The Anatomy Engine is built as a set of six components that can be used together or independently, depending on the needs of the application developer. These components are:

- The Asset Manager, a repository for 3D models, with metadata linking them to the FMA.

- The Style Manager, a tool for specifying "style sheets" that, much like CSS, can specify rules that define how models are displayed in a scene, depending on their various attributes.
- The Query Manager, a wrapper, interface, and caching service for the Query Integrator, a web service developed by the UW Structural Informatics Group that allows the construction and execution of queries not just across the FMA but across any ontology or data set that can be expressed in OWL or some form of XML.
- A Role-Based Access Controller (RBAC) for managing users and their access to the various resources housed in the server.
- A text completion service for terms in the FMA. This removes the need for users to know the FMA's particular terminology for each structure.
- The Common Graphics Application (CGA), a lightweight, browser-based 3D graphics application that can be extended to create more sophisticated applications. All of the applications presented in this chapter have been built by extending the CGA.

With the exception of RBAC and CGA, the components of the Anatomy Engine are lightly coupled web applications running within the same container, with interaction limited to certain narrowly defined interfaces. Though not yet implemented this way, it is planned for all interaction to take place via web services, allowing the various components to be run independently on different servers and allowing them to interact with multiple instances of each other. This permits federation of the services provided, though deployments of that nature are not discussed here any further.

4.1.2 Anatomy Viewer Application

AVA, the Anatomy Viewer Application, is a single-page JavaScript application based on the Common Graphics Application. AVA is fairly straight-forward – it takes a description of the objects that should be shown in a scene and produces an interactive 3D scene in the browser using WebGL. AVA can operate as a standalone web page or be embedded within

other web pages, allowing the inclusion of interactive 3D anatomical scenes in existing learning material.

AVA is described as separate from the Anatomy Engine but its relationship is more complex in that AVA is invoked within the Asset Manager to show previews of models. AVA is also leveraged by the Scene Builder and the Data Visualizer as a means of presenting completed scenes and visualizations to the user.

4.1.3 Scene Builder

The Scene Builder is a web application that uses queries to enable rapid authoring of anatomical scenes. A scene is composed of one or more fragments, which come in two types: query derived fragments are created by selecting a query and specifying its parameters, while asset set derived fragments are created by selecting anatomical structures from a list of those available as assets in some set. A fragment also specifies the asset set and style sheet used to display the anatomical structures within it.

All scene construction occurs within the browser and allows users to specify what is to be shown with high-level statements based on queries. A user might, for example, request "the nerves deriving from cranial nerve V" by selecting a "nerve continuity" query and entering "cranial nerve V" using the text completion service.

Once created, a scene can be saved to the server, from where it can be displayed to users using AVA. Scenes may also be exported, in which case they are packaged using a standalone version of AVA and all necessary asset files. Exported scenes may be hosted as regular files on any server and are not dependent in an way on the Anatomy Engine server.

Like AVA, the Scene Builder is built around the CGA. The Scene Builder, however, extends the CGA substantially, adding additional HTML user interface elements and server-side interaction with web services.

4.1.4 Data Visualizer

The Data Visualizer allows users to build visualizations from scenes created in the Scene Builder by modifying each structure's display parameters according to values in some data set. Style expressions are used to interpret data, which is obtained from queries run using the Query Integrator. Visualizations produced by the Data Visualizer are 3D analogues to choropleth maps, where different geographic regions are colored according to data associated with that region; for example, a map showing literacy levels on a state by state.

Display parameters are determined using style expressions that are resolved against data associated with each structure. If an expression resolves to a value of true, a style is applied. For example, style expressions may select from three styles depending on whether a data value is above 5, between 2 and 5, or below 2.

The Data Visualizer is interactive, in that users may switch between scenes, data sets, and visualization schemes at any time, allowing them to compare different data sets over the same set of structures. As in the Scene Builder, visualizations may be saved and exported for re-use in other web resources.

4.1.5 Anatomy Explorer

The Anatomy Explorer provides a visual browser for the Foundational Model of Anatomy. It presents anatomical structures in a 3D scene alongside detailed information about those structures, retrieved from the FMA. Users navigate either by searching for particular entities or by following relations. Though they cannot be shown in the scene, entities without associated assets can be retrieved symbolically, allowing users to explore the entire FMA.

As users explore, they can choose to add structures to the scene either individually or as a group according to some relation; for example, by adding all muscles that insert on a currently selected bone. This scene is primarily useful as it provides a visual index to the information they have examined, but it can also be viewed as a potential output – that is,

the Explorer can be seen as a ontology based approach to scene authoring that is perhaps more intuitive for users than asking them to directly invoke queries in the Scene Builder.

Other applications of the Anatomy Explorer include using it as a navigational tool for other data sets keyed to the FMA, as a teaching aid for use by tutors explaining the relationship of parts of the body to one another, as a learning tool for revision by students or as an inspection tool for FMA authors. The key advantage of the Anatomy Explorer is that it provides a visual interface to an otherwise largely impenetrable ontology, while still promoting a rigorous understanding of the structures of the body through the FMA's formal knowledge representation.

4.2 In Depth – Anatomy Engine

The Anatomy Engine is a set of libraries and tools that support the development of applications based around anatomical content. Though administrative web interfaces are included, it is expected that most users will interact with the Anatomy Engine through an application.

The Engine is made up of six lightly coupled parts: the Asset Manager, the Style Manager, the Query Manager, the Role-Based Access Manager, the text completion service, and the Common Graphics Application. The relationships between these parts are shown in Figure 12; each is discussed in more detail below.

With the exception of the CGA, these parts are accessible via web services using AJAX and JSON and configured via a browser-based administrative interface. The CGA is a stripped down WebGL application on top of which other applications can be built.

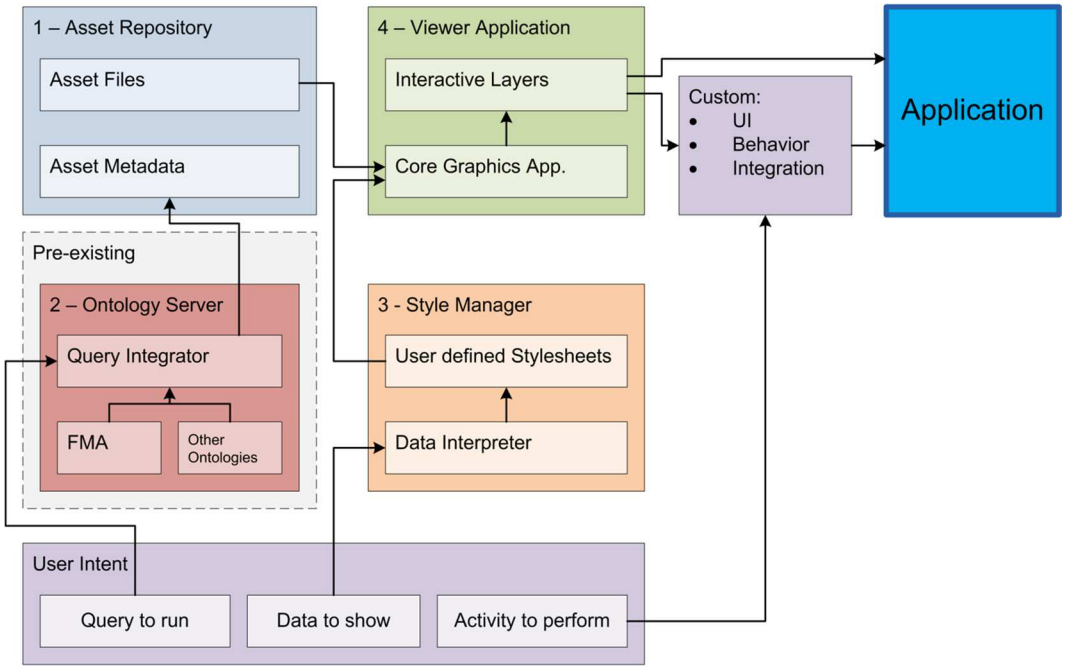


Figure 12 – Anatomy Engine – System diagram

4.2.1 Asset Manager

The Anatomy Engine assembles scenes of human anatomy from 3D models representing individual anatomical structures. The Asset Manager is the component responsible for maintaining these models and making them available for use. Specifically, the Asset Manager incorporates the following functionality:

- A repository of 3D model files on disk for use in scene construction, called assets.
- A database containing metadata and statistics about each asset, including:
    - Information about an asset's source, ownership, and licensing,
    - Bounding box and centroid,
    - A reference to the model file on disk,
    - The ID and name of the FMA entity a model represents,

- A list of tags corresponding to anatomical entity types that the represented entity is an example of; for example, the asset "Left Femur" belongs to "Long Bone", "Bone", and several other types,
- A grouping mechanism whereby each asset is part of a set of assets that can be safely displayed together.

- A web interface for maintaining this repository.

Figure 13 provides a collage of the various user interface elements that make up the Asset Manager.

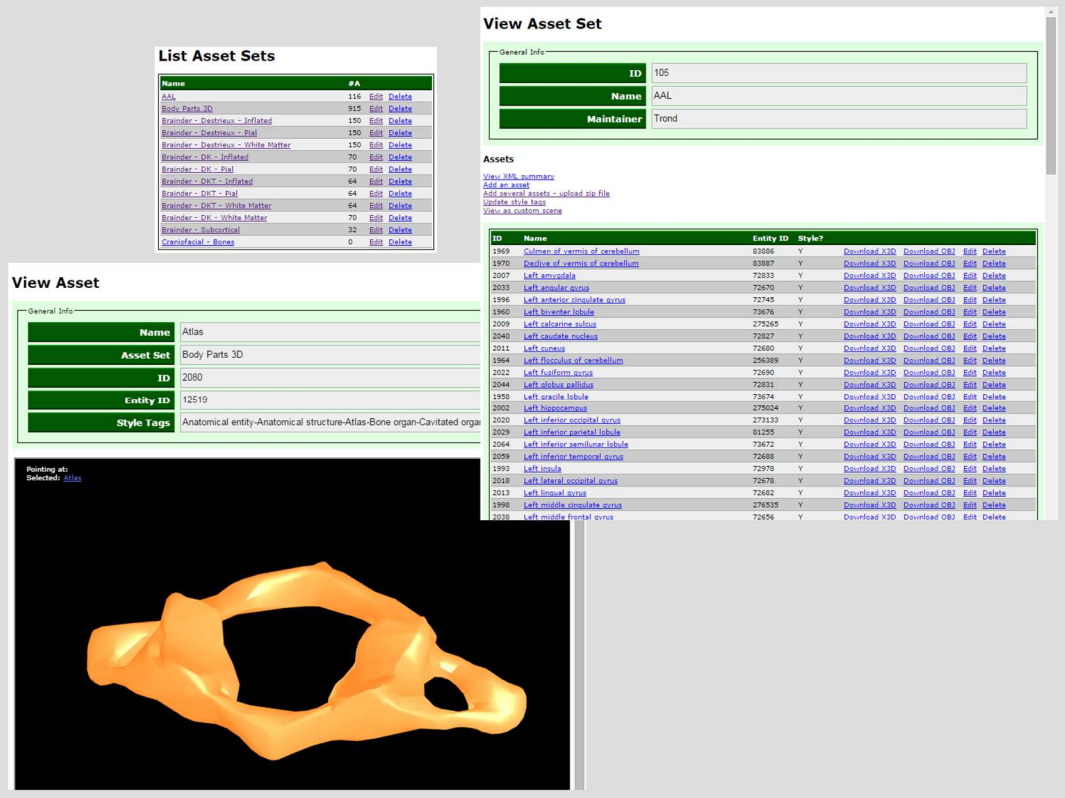


Figure 13 - User Interfaces - Asset Manager

Throughout the Engine the term "model" is often avoided in favor of the term "asset". This is consistent with terminology used in the games industry and implies the existence of assets that are not 3D models. Though the current version of the Engine does not support this, the content generation approach it exemplifies could reasonably be extended to 2D images, audio sections, and text. The term "model", when used, refers either to a 3D mesh when loaded in the CGA or to a model file on disk. The term "asset" refers to an element of content, related to some entity in the FMA, from which content can be composed.

4.2.1.1 Asset Availability

To be at all useful, the Asset Manager must be supplied with a collection of assets that represent a broad range of anatomical structures. Unfortunately, however, high detail models of the whole human body are not available at the same granularity as the FMA. This lack of models is one of the greatest limitations of the Anatomy Engine at this time. Though full body models are not available, high detail models of various parts are available and have been incorporated into the framework:

- The BodyParts3D model set [199], produced by researchers at the University of Tokyo, is the basis of the highest quality asset set currently available within the Anatomy Engine. It consists of 915 assets representing the bones, muscles, viscera, and brain of the human body. Though it contains excellent detail for muscles, it contains almost no vasculature, nerves, or lymphatic vessels. Furthermore it is insufficiently granular – bones are represented as whole units, with no division into smaller parts such as landmarks and regions. Nonetheless, it is an excellent asset set for scenes showing the whole human body.
- The AAL and Brainder [200] model sets contain high detail representations of the brain, divided according to several brain parcellation schemes (AAL, DK, DKT, and Destrieux). Models are available of the pial, white matter, and inflated brain surface. The Anatomy Engine incorporates ten asset sets based on these models, with each set containing between 64 and 150 assets.

- The Craniofacial model set [201], produced by researchers at NYU, contains models of the bones, vasculature, and nerves of the head and neck, with excellent granularity. The craniofacial model set has not yet been fully incorporated into the Anatomy Engine as substantial model editing work is required to slice models into chunks at the same granularity as the FMA.
- Models from Digital Anatomist project [202] are available for the skull, the thoracic viscera and the brain. Though realistic and of high detail, these models will require substantial labor to convert them into a usable format, and are thus not yet available within the Engine. As these models were created by merging segmented slices of the human body, they have a layered appearance not dissimilar to a step pyramid. This is a common feature of anatomy-derived models to which smoothing algorithms have not been applied.

The Anatomy Engine supports the creation of new assets by importing models that have been keyed to entity IDs in the FMA. Obtaining additional models, however, is either time consuming, expensive, or both. Models can be created in one of two ways:

- Artistic models are created by artists using 3D authoring tools such as Blender or Maya, based on sketches, photographs, and other images of real anatomy. Artistic models tend to look very good, conveying an image of anatomy that is well-structured and elegant, with smooth surfaces and well-formed organs. This is deceiving, however, as real human anatomy tends to be much messier, abounding with variations and deformities, blood vessels that do not follow the simplest path, and changes due to aging and injury such that there is no such thing as completely normal human anatomy. Artistic models also suffer from being interpretations of reality by a fallible human who may make mistakes or ignore certain details.
- Anatomy-derived models are re-constructed directly from images of real anatomy. While CT and MRI can provide detailed images of the body with resolutions approaching 1mm, these images tend to be noisy and do a poor job of distinguishing different types of soft tissue from one another due to them sharing

similar physical properties. Alternatively, models may be constructed based on images from the Visible Human projects from the US [156], China, and South Korea [203], which provide detailed images of a human body sliced into layers as thin as 0.16mm. Being based directly on actual human bodies, anatomy-derived models are more authentic than artistic models, showing person-specific abnormalities and variations, as well as detail that is often omitted in artist derived models. Unfortunately, however, the process of constructing anatomy-derived models can be arduous, as automatic processes for producing and labeling 3D models directly from un-segmented images do not yet exist with the fidelity required for many applications. Depending on the algorithms used, anatomy-derived models also often look layered, as they are created from a stack of image slices.

Though the assets available within the Anatomy Engine do not have ideal detail or coverage, higher quality models are available for purchase from several sources. Unfortunately, they are quite expensive, with a full body model set from Zygote, of middling quality, detail, and granularity, costing around $25,000.

4.2.1.2 Implementation

The Asset Manager is implemented as a web application written in Java with Spring MVC running on Tomcat. It is exposed to users for administration via a set of dynamic web pages and is accessible to applications via RESTful web services (to obtain asset metadata) by direct HTTP request (to obtain model files), and via a Java API.

Via the administration interface, users can:

- Manage assets and asset sets (creating, updating, deleting)
- Upload a zip file containing a descriptor and some set of model files, thereby creating assets in bulk
- Trigger the tagging of assets based on the FMA class hierarchy of the entity represented by each asset. Since this task can involve hundreds of query

executions, it sometimes takes some time to complete, and thus runs as a background process.

- View an individual asset or a set of assets selected from a list using AVA, the Anatomy Viewer Application.

In the current version of the Anatomy Engine, all assets correspond to 3D models. Each model is stored in two formats: Wavefront OBJ, and X3D. All of the applications currently built using the Engine's application use OBJ files, but X3D files are kept for use with early prototypes of the system and, potentially, for use with WebGL based X3D browsers that are under development by various working groups.

4.2.1.3 Further Development

Several open challenges and opportunities for further development exist for the Asset Manager:

- Though all assets in a given set are assumed to share the same coordinate system, assets from different sets often do not. This means that assets in scenes composed from multiple sets will often appear unrelated in space. While it is possible to convert between coordinate systems by scaling, rotating, and translating models, this can only ever result in rough alignment unless the models were derived originally from the same individual human body. This is because no two human bodies are identical – we vary in terms of body weight, limb proportions, and so forth. Even our internal structure varies: veins are notoriously variable in the paths they take, and many other structures vary somewhat from person to person. More advanced transformation tools that, for example, scale the body part by part, or allow structures to flex in order to connect with one another are conceivable, and could be incorporated into the Asset Manager. Even basic affine transformation tools would be helpful to some extent.
- As discussed above, the usefulness of the Anatomy Engine is coupled tightly to the range of assets available. Various tools have been produced and proposed to

facilitate the creation of models, including the SIG's AnnoteImage [204] and my own BodyBuilder [205] proposal, but the task remains significant and fully automated approaches to producing models of requisite detail remain out of the reach of current computer vision algorithms. In addition to better models of the material anatomical structures we all know (bones, muscles, etc), models are also needed for immaterial and less obvious entities such as spaces, both real and potential, landmarks, points, lines, planes, and compartments bound by fascia. Yet another complication is that the human body wastes no space – between distinct structures lie connective and fatty tissues that are largely formless and thus ignored in most model sets. Since it is unlikely that anyone will undertake to produce a fully detailed and granular asset set of the whole body for use in the Anatomy Engine, progress is likely to come from the piecemeal acquisition and production of models to satisfy individual application needs. This is not ideal as it undermines the ability of the Engine to create scenes of any part of the body on demand.

- Models derived from human anatomy, particularly those derived from particular individuals, may be subject to regulations constraining their use and distribution. Similarly, licensing rules or the requirements of particular users may limit how models can be used. One approach to addressing these issues is to move from a single server model to a federated model, where multiple instances of the Anatomy Engine are run on different servers by different institutions. This is not significantly different from the idea of different file libraries on different servers – if a user wants a document from Wikipedia, they obtain it from Wikipedia's website.

- Assets are keyed to entities in the FMA through an entity ID. There is no particular reason that these IDs should be limited to the FMA. Assets could reference other ontologies such as RADLex or the OCDM; even arbitrary identifiers from some data set could be supported, if needed. By removing the limit that assets must be associated with the FMA, the Engine can be made to support ontology based

content generation and visualization of arbitrary structures, from mice to car engines.

4.2.2 Style Manager

Asset model data is stored in OBJ and X3D files. These files contain mesh data, but include no information about how a mesh should be displayed. Instead, the Engine uses style sheets, in which the display parameters for each asset are selected by matching information about the entity that asset represents against a set of style rules. Style sheets are a well-known technique for assigning display parameters to content, with the CSS standard [206] in almost universal use on the web. To my knowledge, style sheets have not been used in 3D scene generation before, though conceptually similar systems are used in many computer game engines to style and re-use character models.

Style sheets are managed by the Style Manager, which is responsible for the following functionality:

- A collection of style sheets, stored in a database
- A mechanism for resolving expressions.
- A web interface for administering style sheets, shown in Figure 14.

4.2.2.1 Implementation

The Style Manager is implemented as a web application written in Java with Spring MVC running on Tomcat. It is exposed to users for administration via a set of dynamic web pages allowing them to manage style sheets and styles (creating, updating, deleting), and accessible to other components of the Anatomy Engine and applications via a Java API.

Unlike the web's Cascading Style Sheets (CSS), a standard that inspired the Style Manager's implementation, style sheets in the Anatomy Engine are stored in the relational database rather than in dedicated files, and are managed using a form-based web interface, shown in Figure 14.

Each style sheet consists of the following information:

- A name
- An ordered list of styles, each consisting of
    - An expression (the rule).
    - A set of display parameters (the declaration). Display parameters are discussed below.
- A set of tags. Tags allow applications to indicate the properties of certain style sheets. The Data Visualizer, for example, uses tags to differentiate style sheets that operate on query data from those that operate on entity types.

Styles are applied through a matching process where the metadata for each asset is examined against each style's rule to determine whether that style applies. Styles are considered in order; once a match is found, that style is applied, and no further styles are considered for that asset.

Style rules come in two forms with different matching semantics:

- Tag style rules are simplest and were developed first. Tag style rules specify a string that is searched for amongst all tags associated with an asset in the Asset Manager database. If found, a match occurs. Asset tags are generally generated from the hierarchy of FMA entity types to which the entity an asset represents belongs. Example tags include "bone", "vein", and so forth.
- Expression style rules apply lightweight logical or mathematical expression to data values associated with an asset. If the expression evaluates to true, a match occurs. Support for expression style rules was developed in response to a need for more sophisticated asset coloring within the Data Visualizer application.

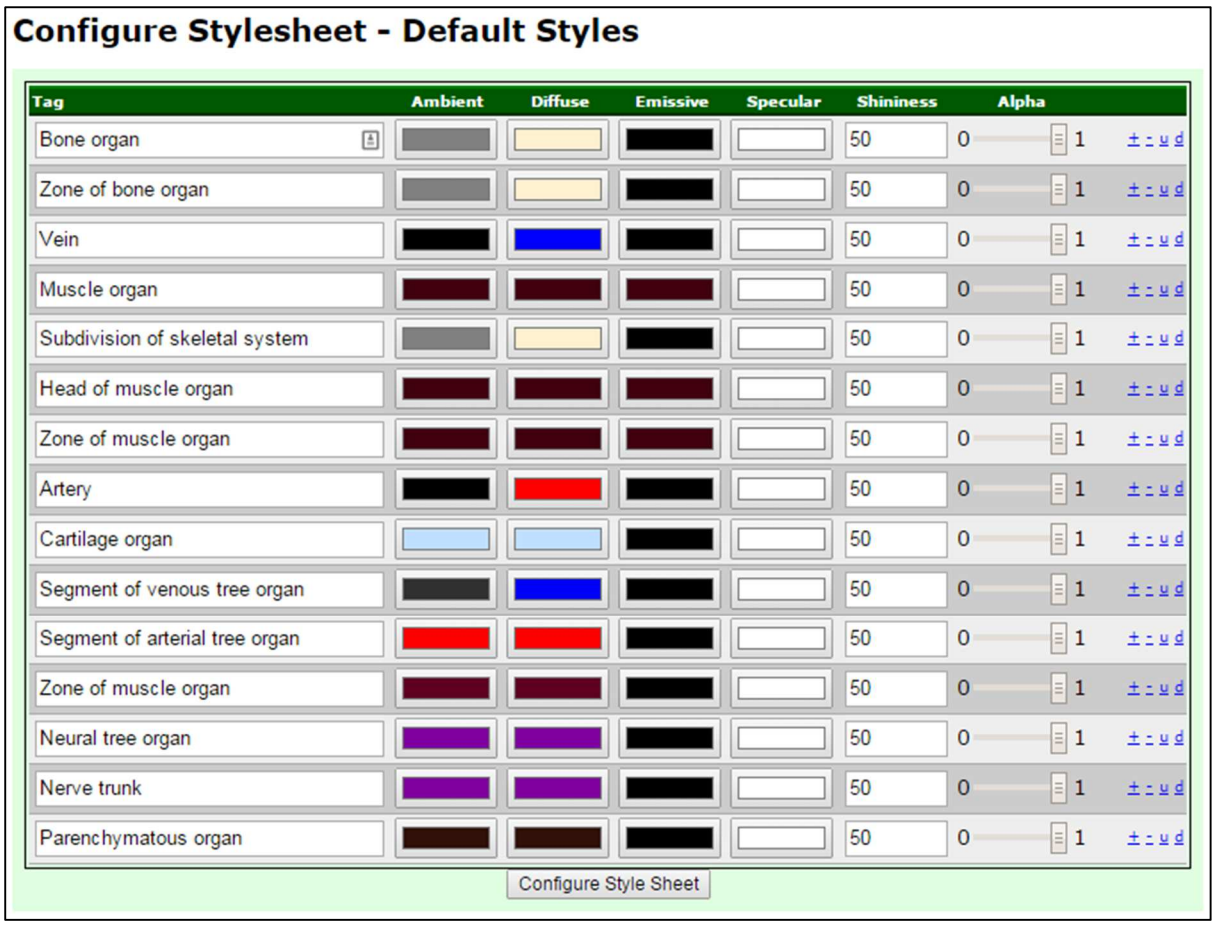


Figure 14 - Sample stylesheet. Style declarations are shown using HTML "color" controls.

Style declarations consist of a set of values corresponding to the standard material and lighting parameters defined by OpenGL for use with the Phong lighting model. These parameters are:

- Ambient color (light reflected from undeclared "ambient" sources)
- Diffuse color (the "matte" component of a surface's reflectivity)
- Specular color (the "glossy" component of a surface's reflectivity)
- Emissive color (light emitted by a surface)
- Shininess (the narrowness of the angle at which a surface reflects specular light)
- Alpha level (how opaque the object is)

For a detailed presentation of the Phong lighting model, see [207] and [208].

4.2.2.2 Further Development

Several open challenges and opportunities for further development exist for the Style Manager:

- The syntax currently used by style rules is relatively ad hoc and simple. Further development of this syntax to unify the two types of rule and support more sophisticated logic would enable the creation of more complex visualizations. Possible extensions might include: allowing the use of more than one data value in a single rule, allowing data values and tags to be mixed in the same rule, subsuming tag rules within expression rules, and supporting a broader range of mathematical operators and functions.
- Style declarations could be extended to support textures and shaders. This could be as simple as allowing users to select from a set of pre-defined options or as sophisticated as allowing users to set shader parameters or even code their own. Textures would allow for more realistic looking assets, but would need to be shape agnostic unless they are tied to a particular asset. Shaders are much more flexible than Phong shading, and support both the creation of different material characteristics for different structure types and the generation of textures for use in information visualization.
- Presently, styles are applied on an all or nothing basis. By allowing the specification of "partial" styles, style sheets could be created that are additive, allowing different display parameters to be applied in response to different rules for a single asset. For example, the emissive parameter might be set according to one data value, and the diffuse parameter according to another. This approach is unlikely to be particularly useful with Phong shading, and may require shader support to be of much value.
- The display parameters used by the Phong lighting model are not intuitive, with proper use only possible with understanding of the model's details. One approach to making the parameters more intuitive and thus the system more usable is to

provide a "preview" model that changes appearance in response to changes in the display parameters. Interfaces like this are commonly used by professional 3D modeling tools.

- The web interface for configuring style sheets was designed with non-technical users in mind, using, for example, color selection dialogues in place of color triplet declarations. As an alternative, style sheets could be represented in a text format similar to that used by CSS. This representation would most likely be more efficient for experienced users, as it would avoid the need for them to manipulate form elements individually, while also supporting copy / paste and find / replace operations. Non-technical and less experienced users could continue to use the web interface.

4.2.3 Query Manager

If assets and styles are the ingredients from which the Anatomy Engine assembles scenes, queries provide the recipe. Queries are executed using an external service called the Query Integrator (QI) that was developed by other researchers within the Structural Informatics Group. The Anatomy Engine manages its relationship with these queries using a component called the Query Manager, which has the following features:

- A database containing information about queries that may be used by the Engine and related applications, including a list of required query parameters
- A mechanism for executing queries that:
    - Invokes queries in the Query Integrator
    - Uses a cache to accelerate query response times
    - Parses query results into a list of entities for which assets should be included in a scene
- Tools to assist in debugging queries

The Query Manager is best thought of as a service provider for use by other components of the Anatomy Engine and related applications. It provides a straight-forward mechanism for consulting the FMA, and though the Engine uses it primarily for simple content generation, other applications such as the tutoring tools and Anatomy Explorer described in later sections use it also to run more advanced queries.

4.2.3.1 Implementation

The Query Manager is implemented as a web application written in Java with Spring MVC running on Tomcat. It is exposed to users for administration via a set of dynamic web pages and is accessible to components of the Anatomy Engine and applications via a Java API. It interacts via web services with a software service called the Query Integrator that runs queries over the FMA and other ontologies on its behalf.

Via the administrative web interface, users can:

- Maintain the list of queries registered for use with the Anatomy Engine (creating, modifying, deleting)
- Manually issue a query for testing purposes
- View the current cache of query responses
- Clear the cache

Queries are not stored in the Query Manager's database, but referenced from the Query Integrator and stored with the following information:

- The query's ID in the Query Integrator
- The query's name and a short description
- A list of the parameters required to execute that query. Each parameter has:
    - A human readable label
    - A variable name expected by the Query Integrator.
    - A flag indicating that a parameter expects a valid FMA ID

    - A flag indicating that a parameter expects a FMA Name and that integration with the text completion service is worthwhile

- A set of tags. Tags allow applications to indicate the properties of certain queries and are generally used to filter the queries that should be listed in some context. The Data Visualizer, for example, uses tags to ensure that only queries that return data that can be visualized are listed in its user interface.

When executed, a query's raw text response is stored as a cache line in the Query Manager's database. When execution of the same query is requested again with the same parameter values, the cached response is used if it is newer than some limit age; currently, the limit is one week. Caching greatly improves the Anatomy Engine's performance by alleviating the Query Integrator's slow response time, which sometimes exceeds several seconds. It also causes occasional problems, however, when responses that cause errors are cached (usually due to issues with parsing or unforeseen problems in the design of a query). This can be addressed easily by clearing the cache, but can cause havoc when a difficult query is being debugged.

Briefly, the Query Integrator is a tool for running queries over XML documents, with special consideration given to semantic web documents expressed in RDF and OWL. Queries can be written in SPARQL and XQuery, both web standards, as well as in the extended versions of these languages vSPARQL and DXQuery. Other languages are available, but have not played a role in my work. The Query Integrator uses an approach called "view integration", where query results can be treated as data sources in their own right, allowing hierarchical composition of queries. For more detail on the Query Integrator, see [56].

In response to queries, the Query Integrator returns XML documents, with responses to SPARQL queries formatted using the RDF schema. Each RDF document contains a list of RDF entities, each expected to include the rdfs:label attribute and the fma:id attributes. The label is used to describe the entity in interfaces for humans, while the ID links the entity to assets that represent it. Other attributes are captured and stored as key / value data fields that may be interpreted by various applications. This latter mechanism is used

by the Data Visualizer to obtain data for visualization, and by the Anatomy Explorer to obtain lists of entities that are related together.

4.2.3.2 Further Development

Several open challenges and opportunities for further development exist for the Query Manager:

- Presently, no type checking or other validation is possible for query parameter values. This proves problematic when queries are made accessible to users who are not familiar with a query's expectations, as improperly formed parameter values can lead to unforeseen results with little to no useful feedback. For validation to be possible, the Query Manager must be able to store information about how each parameter is validated, then apply that validation.
- With over 400 queries written for the Query Integrator, one should, in theory, increasingly be able to re-use existing queries instead of writing new ones. The challenge in doing this is finding the right query for the right job. Queries are listed in the Query Manager by name, but there is currently no easy way to search for a particular query or filter the query list to find queries similar to one's needs. This discovery problem has plagued use of the Query Integrator, and limits the accessibility of the Anatomy Engine overall – it is not reasonable to expect most users to write their own queries or interrogate query code to determine which query best suits their needs. A better way of organizing and navigating the collection is needed.

4.2.4 Role-Based Access Controller

As the Anatomy Engine grows and is used by more people, there will inevitably be a need to distinguish between users. Individuals may wish to protect or share content they have created, and users must be identifiable so that rights of authorship and licensing can be respected. Applications built using the Engine will also likely require user tracking, for

example to separate records of different student's performance in some learning activity. Finally, user management is necessary to support basic security on the web.

The Role-Based Access Controller (RBAC) provides tracking of users, user groups, user roles, and permissions. It is by no means a novel contribution, but is an implementation of a well-known approach to user management in business applications. It is described here for completeness sake.

RBAC provides the following functionality:

- A database of users, including records for:
    - Users, with general user profile information and an encrypted password
    - User Groups, allowing roles to be assigned to many users as a group
    - Roles, being collections of permission that may be assigned to users and groups
    - Permissions, being tokens that are assigned to users via roles and groups, and checked when security is necessary
- A web interface for managing the above
- A mechanism for securing all server-side content including a login screen and site access rules.

4.2.4.1 Implementation

RBAC is implemented as a web application written in Java with Spring MVC running on Tomcat. It is exposed to users for administration via a set of dynamic web pages and is accessible to components of the Anatomy Engine and applications via a Java API. The RBAC component integrates with the server container via Spring Security to provide login and access control functionality.

RBAC's administrative web interface allows users to manage users, user groups, roles and permissions, and offer no additional functionality. Access control rules for different parts of the Engine are specified in a configuration file set at deployment time.

4.2.4.2 Further Development

Further development of Anatomy Engine will likely require the following further development of RBAC or some successor security component:

- Users ought to be able to control the use and modification of resources they create. This requires a mechanism for defining which users have access and which do not. Access Control Lists (ACLs) are a particularly flexible approach to doing this, used widely, for example, within the various UNIX operating systems. Implementation of ACLs within the Anatomy Engine would require modifications to most of its components, as well as a substantial rework of the RBAC component.
- Previous sections have discussed the possibility of federating certain components within the Engine. For this to proceed, some scheme for managing user accounts across multiple servers must be defined.

4.2.5 Text Completion Service

Users are asked to specify FMA entities by name in a variety of situations within the Anatomy Engine and its associated applications. This occurs notably during query execution and within the Scene Builder and Anatomy Explorer applications. To improve usability, a text completion service is provided. This functionality is accessible via a web service and a small unit of JavaScript code that hooks it into any HTML form element. As users type, the text already entered is used to search a database of known FMA entity names, with matches presented immediately beneath the form element so that the user may select them.

4.2.5.1 Implementation

The text completion service is implemented in two parts: a web application written in Java with Spring MVC running on Tomcat and a unit of JavaScript implemented using jQuery and the jQuery UI extension. The service is not directly exposed to users but is to components of the Anatomy Engine and applications via a web service.

FMA entity names are loaded into the database at deployment time, and are not maintainable by users. This manual process has thus far proved acceptable because the FMA's list of entity names has changed only infrequently.

4.2.5.2 *Further Development*

As it addresses a fairly simple need, there is little scope for further development of the text completion service.

Two useful extensions are envisioned, however:

- At present, the only controlled terms supported are FMA entity names. The service could be extended to help users select FMAIDs or select controlled terms from other ontologies or vocabularies that might be used with the Anatomy Engine. Such an extension would likely rely on type information for query parameters to select which set of terms should be used.
- Recently, the FMA has been converted from a frame-based representation to the W3C's OWL format. This has supported architectural changes that allow more frequent updates to the working copy of the FMA used in the Query Integrator. In turn, there is now a need to regularly update the cached names used by the text completion service so that there is no mismatch. A logical approach would be to add a scheduled service that periodically updates the text completion database using the results of queries across the current version of the FMA.

4.2.6 Common Graphics Application

Once a scene descriptor has been assembled from assets and styles using the results of a query, it must be presented to users graphically. It is at this level of user interface and presentation that most of the variation between applications of the Anatomy Engine is apparent. That is, while all of the applications presented in this dissertation use the same server side components to generate scenes, they each vary significantly in their interface needs. Despite this, all share certain core functionality – a 3D graphics engine, camera control, and basic UI event handling. Rather than requiring that each application be

developed independently, resulting in several code bases with significant overlap, the Anatomy Engine provides a stripped down application kernel that can be extended to produce other applications. This kernel is called the Common Graphics Application (CGA), and provides the following functionality:

- A graphics engine that, given a list of assets and styles, renders a 3D scene into an HTML5 canvas element within a web browser using WebGL and three.js. This graphics engine supports:
    - OpenGL ES vertex shaders
    - A full scene graph
    - Background object loading and caching
    - Arbitrary user interface development via HTML elements laid over the 3D canvas.
    - Scene embedding within other HTML documents
- An "examine" mode camera that allows users to inspect the scene from any arbitrary perspective.
- An extensible interaction control layer that, in addition to supporting standard JavaScript mouse and keyboard events, supports hover and selection events of 3D objects in the scene.

4.2.6.1 Implementation

The CGA is implemented as a JavaScript application using WebGL, three.js, jQuery, and lodash. Users interact with the CGA directly within the browser using the mouse and keyboard. Developers create their own applications using the CGA using the Decorator and Observer patterns applied to its base controller and user interface classes to add their own interaction control code, initialization and control methods as needed, and their own user interface code.

There are three main aspects to the CGA – the graphics engine, the interaction control layer, and the camera, each described in the sections below.

4.2.6.1.1 Graphics Engine

The graphics engine acts as a wrapper for low level rendering code provided by three.js. It is responsible for:

- The application's main loop. During each frame, interaction control events are handled, scene updates applied, and the scene is rendered
- The 3D objects and scene graph. A scene graph stores the contents of a 3D scene hierarchically as a tree structure, with the transformations applied at nodes closer to the root applied sequentially to those nodes linked below them. The CGA's default scene graph is quite simple; it is less an oak tree and more a palm in that it consists of a line of several linked transformation nodes with a spread of leaf nodes attached to the last node in the line. Nodes in this line are thus global transformations that affect the whole scene. Other scene graph structures are, however, possible, to support more sophisticated applications. By default, all objects in the scene are transformed such that the overall centroid of the scene is at (0,0,0).
- Modification of the scene, including the initial loading of resources. An application built on the CGA can adjust existing scene content at any time and may also request the addition of new, freshly loaded objects. All resource loading occurs asynchronously – applications continue to function normally while new objects are loaded.

4.2.6.1.2 Interaction control layer

The interaction control layer allows the creation of complex interfaces using an approach based on the observer and decorator patterns. All incoming events from JavaScript and three.js are intercepted by the `InteractionManager` class, which maintains a list of Controller classes to which they are dispatched. Mouse hover and click events over the 3D

canvas trigger picking in which a ray is cast into the scene to determine if it intersects any 3D objects. If this is the case, additional `objectHovered` and `objectSelected` events are generated.

The CGA provides a basic controller class that handles help requests and interprets mouse events for use by the camera. Applications are expected to provide their own Controller classes to interpret any additional interaction events they require. The system is set up so that functionality can be split across multiple controller classes, allowing easier re-use; for example, a controller that intercepts selection events and ties them into a user interface class that shows the currently selected object might be re-used by multiple applications.

The CGA's `InteractionManager` class also includes a help system that interrogates the Controller classes registered with it to determine what commands they understand. These commands are then listed together in a help interface that can be made visible by hitting the "?" key or clicking the help icon in the corner of the application.

4.2.6.1.3 Camera

The camera spans the interaction control layer and the graphics engine, taking events from one to control the other. The behavior of the default CGA camera is based on the "examine" mode defined in the X3D specification, though other cameras can be registered with the graphics engine. An "examine" mode camera has a fixed point about which it revolves, controlled by horizontal and vertical mouse motion (when activated by holding the right button down. The camera itself can also be rotated around the axis along which it looks by holding down the up and down arrow keys.

The camera's initial position and perspective may be specified when the CGA is initialized. If no viewpoint is provided, the camera is placed on the outward Z axis, at a distance of three times the scene's maximum extent. This ensures that, by default, the camera is far enough from scene geometry for it to be rotated in any direction without the two intersecting. The camera's distance and center of rotation can be redefined by the user using the mouse wheel and mouse cursor, respectively.

4.2.6.2 Further Development

Now that the CGA has been incorporated into several other applications, care must be taken when developing it further. A versioning scheme can allow applications running on older versions to be preserved without updates when the CGA changes, but has not yet been necessary. Before much further development is undertaken however, a scheme should be defined.

Several further developments to the CGA have been considered:

- A "fly" mode camera is desirable for use in some applications. Cameras of this sort are controlled by pitch, yaw, and roll commands along with forward motion at some speed. They are so named as they cause a user to feel as if they are flying through a scene.
- Better support for more sophisticated scene graphs would be helpful. Currently, the CGA graphics engine API is most effective for relatively simple scenes in which objects are positioned in a group as in a display of some kind. This is suitable for exhibit-style anatomy scenes, but is insufficient to support more advanced interactive applications, such as games.
- Improved management of memory. JavaScript's usage of memory is inefficient and poorly managed by default, leading to browser crash events when very large scenes are loaded. Improved management of the assets loaded into memory should alleviate this problem significantly.

Throughout development, functionality has generally first been developed within some particular application before being migrated to the CGA when it has become apparent that it would be of use within other applications.

4.3 In Depth – Anatomy Viewer Application

The Anatomy Viewer Application, called AVA for short, was the first application developed using the Anatomy Engine and remains the simplest. Its purpose is simply to present an

anatomical scene to users. It was developed before the CGA as a standalone application, and much of its functionality eventually went on to form the core of the CGA.

In addition to the basic functionality provided by the CGA, AVA provides the following features:

- A scene summary panel that lists structures shown in the scene.
- A "dissection" mechanism whereby users remove structures from the scene to look more deeply
- Support for pre-defined viewpoints set by the scene's author
- Support for links to external information resources concerning scene contents.

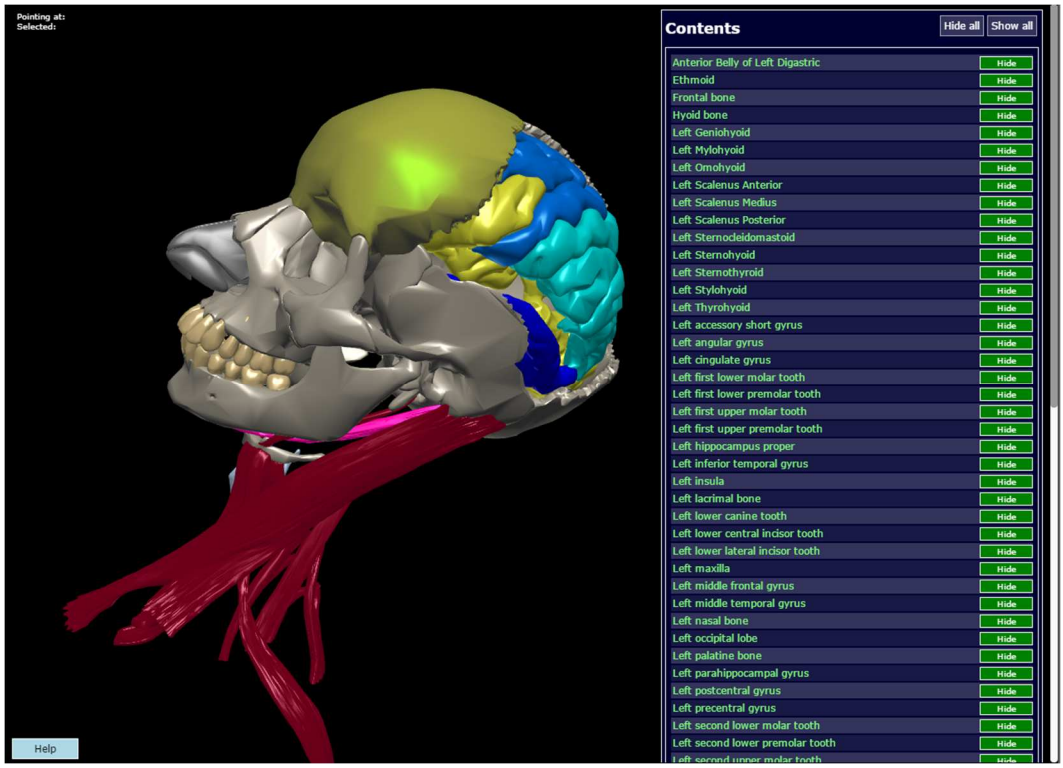


Figure 15 - User interface - AVA

4.3.1 Usage

Usage of AVA is straight forward – a scene is loaded and shown to the user, who may control their view by using the CGA's examine camera mouse controls and pop up a list of structures shown in the scene by pressing "C". The currently selected and currently pointed at structures are identified in a text panel at the top of the screen, and users can control the visibility of structures using buttons on the contents panel or by pressing "V". Figure 15 illustrates a typical scene shown using AVA.

AVA can be embedded in other web resources such as, for example, Wikipedia pages, tutorials, or other applications using HTML iframes.

4.3.2 Implementation

AVA is implemented as a JavaScript application using WebGL, three.js, jQuery, and the CGA. Users interact with AVA directly within the browser using the mouse and keyboard. AVA can run from the Anatomy Engine's server or it can be exported as a standalone HTML document that can operate without any connection to the server.

As the simplest application built using the CGA, AVA serves as an example of how such applications can be built, and is documented accordingly.

4.3.3 Further development

Though it is desirable to keep AVA fairly simple, the following two pieces of functionality seem worth the extra complexity:

- Support for multiple pre-defined viewpoints. In some circumstances, multiple views of the same anatomy are required, and viewpoints may have names. Users of AVA ought to be able to select pre-defined view points from a list as well as controlling the view directly themselves.
- AVA's user interface elements are built with certain assumptions about the size of the viewport. If AVA is embedded within a viewport of less than about 600 by 600 pixels, these UI elements block out the scene, making them useless. Ideally, they

should be built to scale more readily. Small viewports are particularly relevant when AVA is used to visualize anatomy as part of a larger webpage within an iframe.

4.3.4 Extension - The Ghost Anatomy Project

Like the CGA, AVA was built to support easy modification and extension. The Ghost Anatomy project, produced in Spring 2014 in conjunction with a group of undergraduate students, offers an example of this.

These students, Connie Huang, Ngoc Do, Ashish Chandwani, Alyssa Trinh, and Ted Tagami, developed Ghost Anatomy as their capstone project for graduation from the University of Washington's Information School. The project involved the creation of a prototype spatial augmented reality visualization system in which a prism placed on the center of a large format display screen gives the illusion that virtual 3D content is hanging in space. To achieve this, the screen display is divided into four quadrants corresponding to the four sides of the prism such that they are reflected off its sides and appear to float in space.

The software used for this project was an extended version of AVA largely developed by myself, along with a series of interface extensions built by the students using the Leap Motion Controller.

The Ghost Anatomy system was displayed at the Information School's annual student research fair along with approximately 150 other student projects, and received the Audience Choice award.

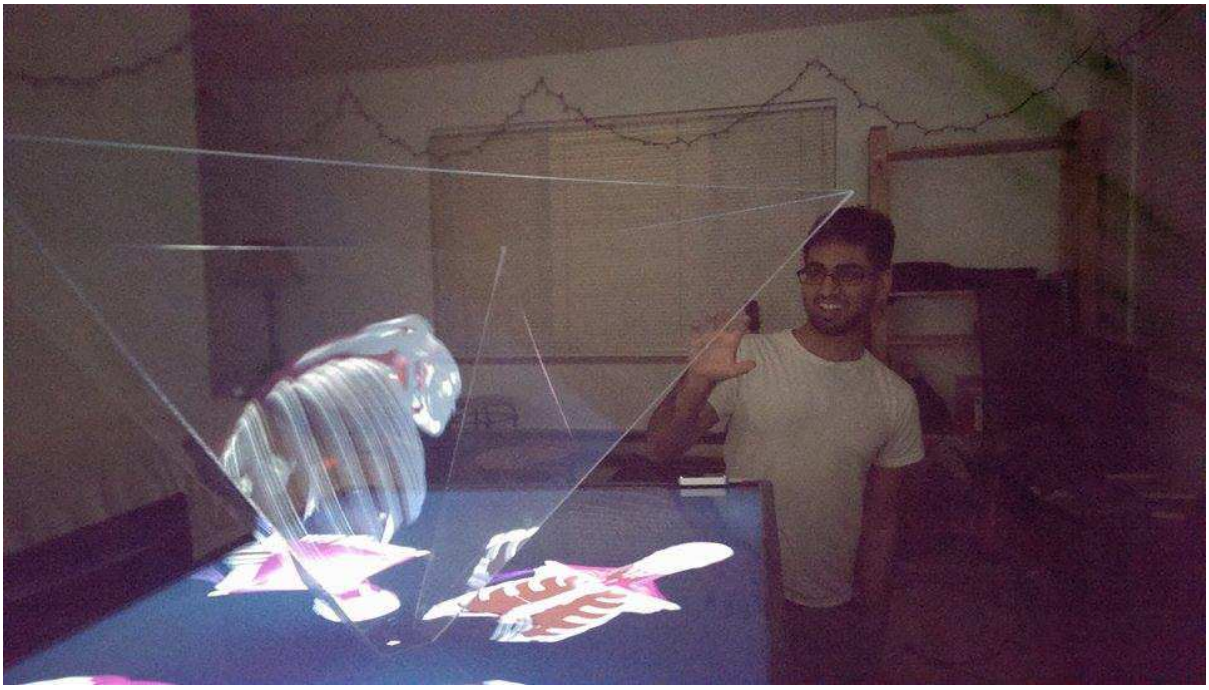


Figure 16 - Ghost Anatomy project, with Ashish demonstrating its use

4.4 In Depth – Scene Builder

The Scene Builder wraps the content generation features of the Anatomy Engine, providing an interface through which users can create and share anatomical scenes. It builds on the CGA and employs the asset, query, and style managers to create scenes that can be displayed independently of the builder using AVA. To some extent, the Scene Builder is the workhorse that takes the potential of the Anatomy Engine and puts it together into a usable tool.

Scenes within the Builder are composed from fragments produced either from the results of a query or from assets selected from a particular asset set. Anatomical entities are described within fragments as members, each an asset paired with additional display parameters. Finally, scenes have viewpoints, specific perspectives from which a scene is initially displayed. Once created, scenes may be saved for later editing, viewed using AVA, or exported alongside the standalone version of AVA for use in some other context.

In addition to supporting scene creation and editing by users, the Scene Builder supports applications that either need a scene to build on or want to save scenes of their own. As examples, the Data Visualizer builds visualizations on top of scenes produced using the Scene Builder, while both the Visualizer and the Anatomy Explorer use the Scene Builder's scene storage API to save scenes created in those tools.

Overall, the Scene Builder provides:

- A platform for the creation of scenes using high-level instructions expressed as the queries and query parameters or by selecting entities from a list.
- A service for the storage of scenes so that they can later be used in other applications such as AVA and the Data Visualizer.

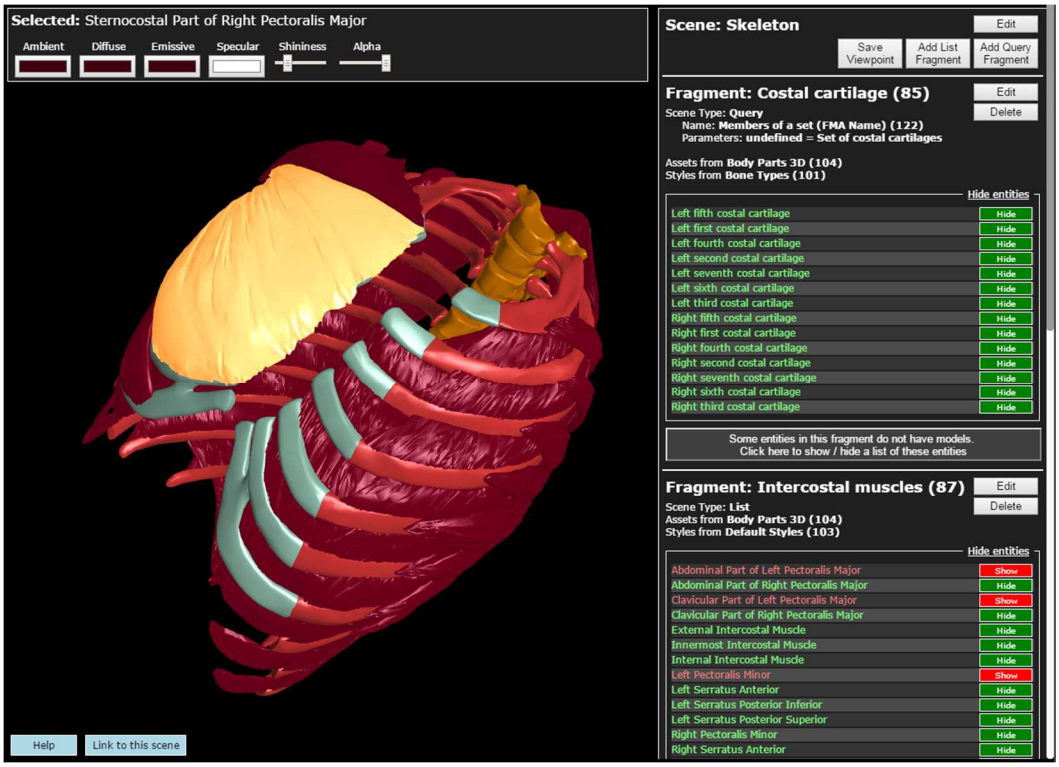


Figure 17 - User interface - Scene Builder

4.4.1 Usage

The Scene Builder's user interface is split into two parts: a series of administrative pages embedded within the Anatomy Engine's overall administrative interface, and a JavaScript application where they are able to visualize and build scenes.

Using the administrative pages, users may:

- Create a new scene
- View a scene
- Export a scene for deployment on another server
- Load a scene for editing

Within the JavaScript application, users may also perform the following operations:

- Change a scene's name and description
- Add a query scene fragment to a scene

- Add a list fragment to a scene
- Modify an existing scene fragment
- Change whether a particular scene fragment member is visible in the scene
- Modify the display parameters of a particular scene fragment member
- Set the scene's default viewpoint

As the Scene Builder asynchronously saves all changes to the server as they are made, there is no separate save action.

Full instructions and worked examples of the use of the Scene Builder can be found in section A.1 in the Appendix.

4.4.2 Implementation

The Scene Builder is implemented as a web application written in Java with Spring MVC running on Tomcat, paired with a JavaScript application written using WebGL, three.js, jQuery, and the CGA. It is exposed to users for administration via a JavaScript application and a set of dynamic web pages, and is accessible to other components and applications of the Anatomy Engine via a Java API and web services.

The administrative web interface is simple. From it, users can create new scenes, view existing scenes using AVA, export scenes for standalone deployment on another server, or open the Scene Builder JavaScript application to work on a scene. Within the JavaScript applications, users may edit all aspects of a scene, as described in the "Usage" section below.

The Scene Builder operates on Scenes, Scene Fragments, and Scene Fragment Members. A scene is a whole artifact that can be loaded for viewing in AVA; a fragment is a logical unit within a scene containing assets from the same set, using the same styles, and selected via the same mechanism; and a fragment member represents an individual entity represented in the scene. Scenes are stored in the database, which includes the following information and links for each scene:

- Name
- Description
- Viewpoint – the default position and perspective matrix for the camera when viewing the scene.
- Zero or more Scene Fragments, each consisting of:
    - Name
    - If used, the query ID and parameters from which the fragment was generated
    - ID of the asset set used to represent entities in the scene
    - ID of the style sheet used to set asset display parameters
    - Zero or more Scene Fragment Members, each consisting of:
        - ID and name of the entity contained in this member
        - ID of the asset used to represent the entity
        - Saved display parameters for the asset
        - Saved 3D transform to be applied to the asset
        - A visibility flag

Scene fragment members store their own display parameters, despite the fact that, in principle, these could be derived from the fragment's linked style sheet. This enables the Scene Builder to store scenes that have been styled in some unconventional manner, such as those generated by the Data Visualizer. Scene fragment members also store a transform; this is not currently used by the Scene Builder itself, but was included to support future development.

Depending on the queries used, an entity may exist within multiple fragments in the same scene. If those fragments use the same asset set, assets for that entity will occupy the exact same space in the viewer. This leads to visual artifacts and unstable interaction

behavior, and so the Scene Builder JavaScript application currently prevents this from occurring by automatically making one of any pair of overlapping objects invisible.

Scenes may be composed of fragments derived from either queries or list selection. List fragments subvert the query driven generation approach to scene construction, but, when only simple scenes are required, are often preferable as they are both easy to use and learn. Third party fragments also exist; these are generated when scenes are saved from other applications and are similar to list fragments but cannot be edited from within the Scene Builder's JavaScript interface.

The Scene Builder's web services are RESTful. An HTTP GET request will retrieve an existing scene, while HTTP PUT and POST requests save a scene by creating a new scene object or modifying an existing one, respectively. Scene data is encoded using JSON.

4.4.3 Further development

There are several open challenges and possibilities for further development of the Scene Builder:

- Though the initial viewpoint of a scene can be specified, scene authors may require scenes with multiple key viewpoints that a user can select from. The Scene Builder ought to support this; modifications to AVA would also be required.
- Similarly, scenes currently show one set of visible assets, and one set only. The Scene Builder could easily be extended to support different sets of visible assets that could be selected between by the user. This would, for example, facilitate scenes where multiple layers of the body are shown, as in the transparent plates found in some textbooks.
- As mentioned above, when the same entity appears in multiple fragments using the same asset set, mesh geometry is overlaid, resulting in unpredictable behavior. The compromise of preventing this by forcing additional copies of any entity to be invisible works, but it is problematic if transforms are to be allowed on individual

fragments and members. For example, under this policy, a scene showing two adjacent views of the body could not include the skeleton in both views.

- Rather than requiring authors to begin from scratch each time, the Scene Builder ought to let them extend existing content to create new scenes. Two methods are likely to be implemented eventually: the Builder's JavaScript will be extended to support the import of fragments from other scenes either individually or as a group into the scene current under construction, and options will be added to the Builder's administrative web interface to support the automated duplication of an existing scene. This addition would significantly improve the usability of the tool, particularly when several related scenes are required.

4.5 In Depth - Data Visualizer

An entity's appearance within a scene can be determined automatically from its type using style sheets or specified manually using the Scene Builder. The Data Visualizer provides a third method, whereby display parameters are assigned to an entity by resolving style rules against data associated with that entity.

Data is accessed using queries, which requires that it be stored in either an XML-based format or in some other format that can be converted to XML. Data must also be annotated using either the FMA or some other scheme that can be mapped to the FMA by a query.

This capability allows users to visualize data that is related to anatomy in a manner similar to how choropleth and other thematic maps visualize geographic data, as shown in Figure 18. Visualizations are created interactively in the browser: users specify their contents by loading an anatomical scene on which to visualize data, a query used to obtain data, and a style sheet to map data into display parameters. Once created, a visualization may be saved as a new scene and made available for viewing using AVA.

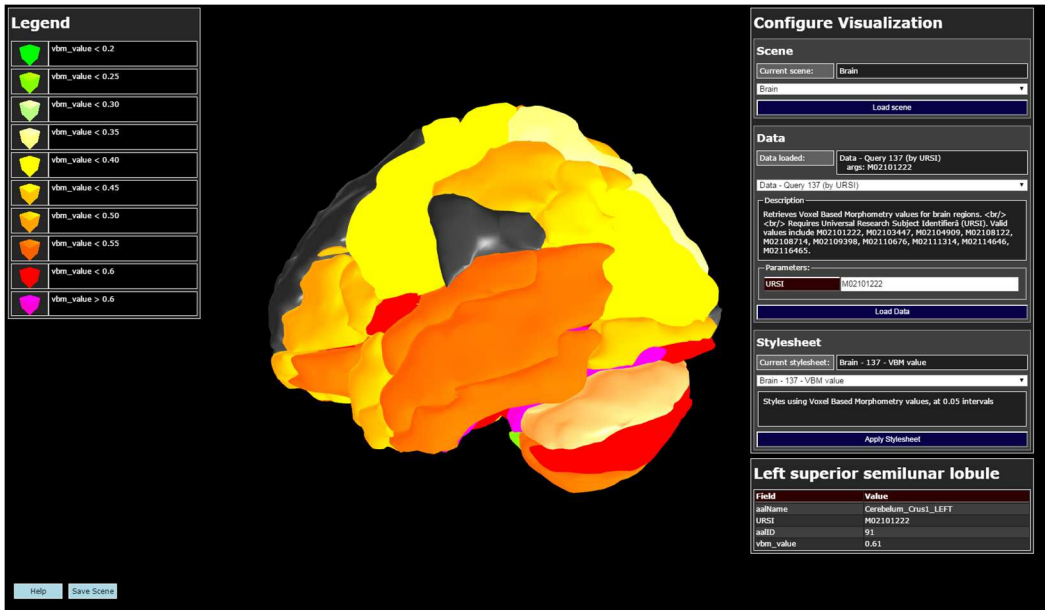


Figure 18 - Example choropleth maps: a brain map made using the Data Visualizer (left) and a map showing the proportion of 2011 Australian census respondents who identified as Anglican (right).

The main limitation of the Data Visualizer is that control is exercised over display parameters on a per-entity basis. As a result, data cannot be visualized unless it is indexed using the entities from the FMA or some related ontology or term list. Furthermore, each entity may have only one value per type of data and, since assets can only be assigned a single set of display parameters, the amount of data that can be visualized at once is limited.

Consequently, spatially indexed data cannot be visualized unless it is first transformed to use anatomical annotations, ideally from the FMA. For example, a collection of data points indexed by spatial coordinates within the body can be transformed into a data set with anatomical indexing by assigning to each entity the average value of all data points considered to lie within it. Similar methods may be applied to 3D voxel arrays and continuous data. Unfortunately, determining the spatial extent of each entity is harder than it might seem. There is no standard for unambiguously mapping spatial coordinates to anatomical entities throughout the body, and such a standard would be extremely difficult to create, as it would need to take account of the way human bodies vary from individual to individual. The closest we have come to such a standard are standardized

brain coordinate systems, such as Talairach, which use key anchor points to convert between spatial coordinates from a particular brain to coordinates in a standard brain which can be indexed to anatomical structures using a brain atlas. This process is effective, but not 100% reliable, as variation in brain structure and proportion between anchor points is not accounted for.

4.5.1 Usage

All user interaction with the Data Visualizer occurs within the JavaScript application. Using controls in the configuration and persistence panel, a user can:

- Load a scene,
- Load data,
- Load a style sheet, or
- Save a visualization.

Each session begins with a blank visualization; no scene, query, or data is loaded by default. Once loaded, a scene or style sheet is used until a different one is loaded to replace it. Data, once loaded, remains until it is overwritten by different data. The visualization displayed to users is maintained to reflect the current configuration at all times. That is, the current style sheet is immediately re-applied whenever the scene is changed or data is loaded.

Full instructions and worked examples of the use of the Data Visualizer can be found in section A.2 in the Appendix.

4.5.2 Implementation

The Data Visualizer is implemented as a web application written in Java with Spring MVC running on Tomcat, paired with a client-side JavaScript application written using WebGL, three.js, jQuery, and the CGA. Visualizations are created in the JavaScript application and may be saved using web services provided by the Scene Builder. Server-side code is relatively limited and is primarily responsible for translating requests from the client-side

application to API calls on various Anatomy Engine services. The Data Visualizer provides no programmatic interfaces for use by other applications.

Visualizations are created from scenes, queries, and style sheets as follows:

- The contents of a visualization are determined using a scene loaded from the Scene Builder's repository. Scene fragment members contribute assets if they are associated with an entity; if not, they are ignored. The scene's internal grouping of members into fragments is also ignored as are all display parameters.
- The result of a data query is parsed to extract data points. Each data point has a key and value, and is associated with a single entity. It is assumed that data points with the same key share the same sort of meaning and data type; for example, all data points with the key "height" indicate the height of an entity, measured in the same way using the same units. Keys must be unique on a per-entity basis. An error is thrown by the Anatomy Engine's Query Manager whenever, during parsing, multiple values are found for the same key and entity. Multiple data queries may contribute data points to a single visualization, in which case query results are loaded and parsed in some order specified by the user. Each query's data points are merged with those from any previous queries, with entity / key conflicts resolved by overwriting the older value. Query parameters are specified by the user before invocation. If a single parameter for a given query is flagged as requiring an entity ID, the query is invoked once for each entity in the scene and the results are combined. Otherwise, the query is run only once.
- Asset display parameters are determined by running a style sheet over associated entities and data. See section 4.2.2 for information on how style sheets are applied.

Scenes, queries, and style sheets may be loaded in any order, with the exception that only the most recently loaded data value is kept for any given entity and key. All data retrieved by a query is stored, irrespective of whether it is associated with an entity. This allows the contents of a visualization to be changed without running all data queries again.

The Data Visualizer's user interface is divided into several parts:

- A 3D viewport where visualizations are shown while they are being created. This viewport spreads across the whole screen such that other user interface elements appear to float above it and is configured so that the camera's center of rotation is not in the center of the screen but in the center of the area not occluded by other user interface elements when they are visible. This reduces the chance that other UI elements will obscure 3D content on lower resolution screens.
- A large panel on the right of the screen, called the configuration panel, containing controls that allow the user to configure the visualization. It is divided into three sub-panels, one each for controlling the scene, query, and style sheet used to generate the visualization.
- A small panel on the right of the screen, directly below the configuration panel, called the data panel, where any data that has been loaded for the currently selected entity is shown.
- A panel on the left of the screen, called the legend panel, where 3D material swatches and descriptions for each of the styles used in the current visualization are shown. Its size varies depending on the complexity of the visualization. Material swatches show a miniature 3D scene using a particular set of display parameters. They are produced for each style by rendering a cube using that style's display parameters in a temporary WebGL context, flattening that context into a data URI, and then using that URI as the source for an image element.
- A hidden panel, called the persistence panel, allows users to save visualizations to the Scene Builder's scene repository. This panel is revealed using the "S" key or by clicking a button in the lower left of the screen.

A screen shot showing the Data Visualizer interface can be seen in Figure 19.

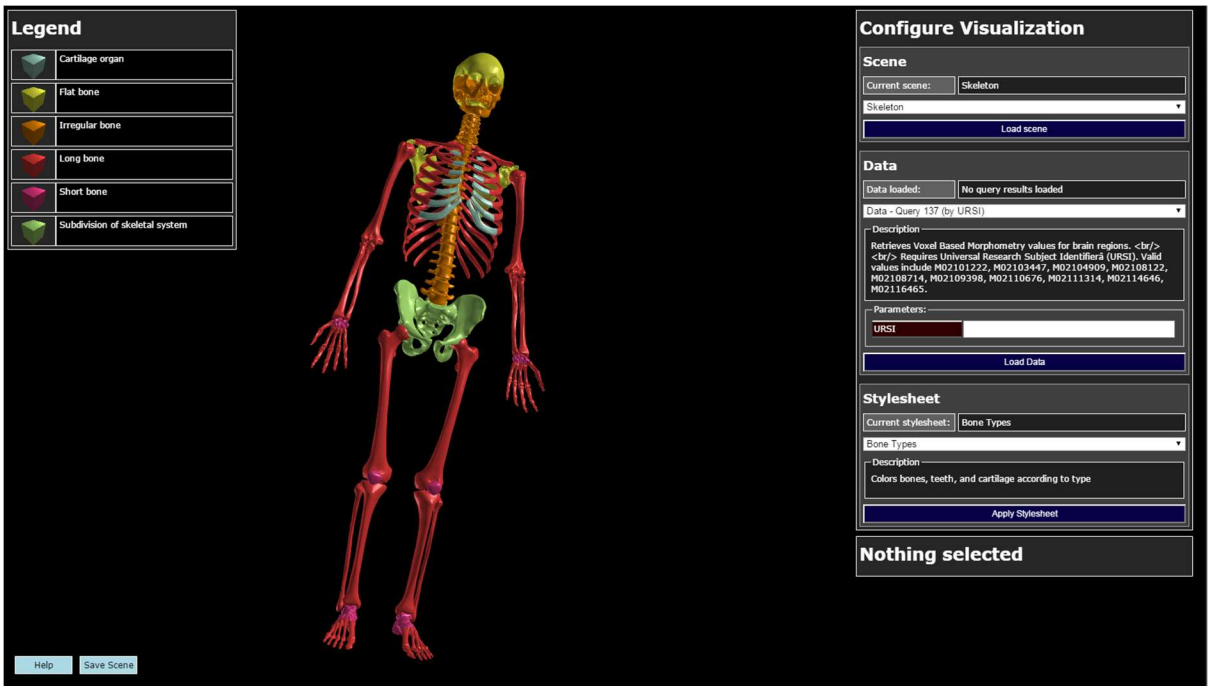


Figure 19 – Data Visualizer UI

The Data Visualizer has no server-side data storage capabilities of its own. A visualization may be saved using the Scene Builder's repository if it is first flattened into a scene descriptor, but this results in the configuration of the visualization being discarded along with any loaded data. Visualizations must be created from scratch during each user session.

4.5.3 Further development

As discussed above, the Visualizer requires a one-to-one relationship between data points and entities such that each data point is associated with an entity and no entity is associated with more than one value for each key, preventing the display of spatially indexed data. One approach to solving this problem is to use queries to convert data from spatial to anatomical indexing before delivering it to the Visualizer. Methods in which data is visualized in some way other than simply defining the display parameters of each entity are possible, provided a mapping exists between the data's spatial coordinate system and that of the assets used:

- Arbitrary spatially indexed data points can be visualized as markers overlaid on a scene. This allows viewers to interpret the association between data points and the structures themselves. As an example, Figure 20 shows a visualization of activation sites overlaid on a sample brain, generated using a prototype extension to the Data Visualizer, based on data from the BIRN data set described in [209] using queries described in [56]. Note how data points do not precisely align with the assets shown; this is due to the difficulties in mapping between individual and general coordinate systems.

- Spatially indexed data that is organized as a 3D array of voxels can be visualized by using it to generate an image texture for each asset in the scene. This method is more complex, and may be best implemented using vertex shaders. The same problems with mapping between coordinate systems continue to apply.

- Multiple data values can be visualized on a single entity using structured textures of various sorts such as crosshatch and vector patterns. The effectiveness of such methods is unknown, though highly successful methods exist for 2D visualizations in various domains, such as meteorology.

Another way in which the Visualizer might be improved is in the management and presentation of completed visualizations to users. Currently, visualizations are saved and presented to consumers by flattening them into the Scene Builder's scene repository then displaying the results to users with AVA. This has two main problems: firstly, all details about how a visualization was created are lost and, secondly, since AVA is a relatively basic viewer, it is missing user interface features that would be desirable when viewing visualizations, such as a legend, notes, and comparison controls.

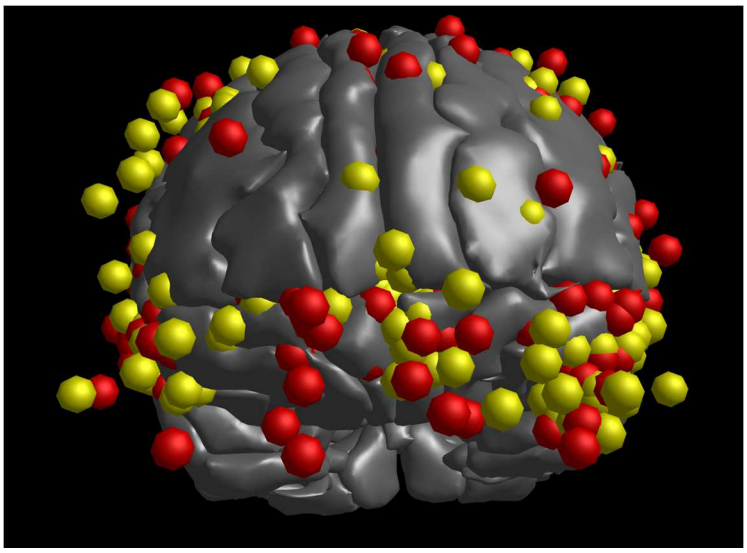


Figure 20 – Example of overlaying markers on 3D scene

To address the first problem, the Data Visualizer must be extended to store visualization configurations in the database itself. This would involve the addition of server-side code to interact with the database and provide a web service API along with client-side user interface code. At a minimum, each configuration should list the source scene, style sheet, and ordered queries required to generate a visualization. Additional data, such as a title, description, and other metadata would also be helpful.

Solving the second problem would require implementing a special viewer for visualizations that is capable of loading a visualization from its configuration and displaying it along with a legend and any metadata such as a title or description. Since users often want to compare multiple visualizations at once, this viewer should be built to allow users to alternate quickly between multiple visualizations or, alternately, show multiple visualizations alongside each other, perhaps sharing the same camera controls.

4.6 In Depth - Anatomy Explorer

The Anatomy Explorer provides a visual frontend to the Foundational Model of Anatomy itself. It presents detailed reference information about particular anatomical entities alongside a 3D scene showing entities and their spatial relationships to one another. Though the Explorer provides access to all entities and relations within the FMA,

annotations and provenance information are not currently available, nor are some free-text properties such as synonyms and entity names in other languages.

The Anatomy Explorer is suitable for users who want to:

Find information – the explorer is easier to browse and navigate than, say, a text book, and presents information in a consistent format that is easy to peruse.

- **Learn** – the explorer supports learning activities such as research and problem solving that force students to actively process anatomical information, thereby improving retention and integration with existing knowledge.
- **Create content** – the explorer offers an intuitive and interactive means of constructing content. Scenes can be constructed directly in the explorer or created in the Scene Builder based on information from the Explorer.
- **Find and fix problems** – the explorer can help ontology authors, content creators, and query writers debug their work either by correcting their understanding of how structures are related or by finding errors and omissions in the FMA itself. Omissions, when found, can be fixed within the FMA using Protégé, and are made available to the Explorer within one week.
- **Understand how the FMA works** – the explorer is a reference tool that can help anyone who wants to work with the FMA in a technical capacity understand its structure and function.

The Anatomy Explorer presents anatomical knowledge using the same formalism as the FMA. Each concept, structure, substance, and space within that body of knowledge is represented as an anatomical entity, with an ID, a label, and a list of relations with other entities, where each relation has a type and can be interpreted as a fact about that entity stated in subject-predicate-object form.

It is easiest to understand this representation as a directed graph with entities at its nodes and relations on its edges. Relations are directed, with the source entity considered the subject, the destination the object, and the relation's type the predicate. For example, a

relation with type **Has related part** that originates at ***Left_scapula*** and terminates at ***Left_coracoid_process*** asserts that the left scapula, a bone in the shoulder, has a regional part called the left coracoid process.

Entities within the FMA are internally defined by their relation to one another – using it, one can know that there is a bone called the scapula that has a regional part called the supraglenoid tubercle to which a muscle called the biceps brachii is attached, allowing one to reason about those structures without knowing what they look like. Relations, however, cannot be understood from the contents of the FMA alone, and must be understood by appeal to definitions laid out by the ontology's creators. For example, the FMA distinguishes between regional parts, constitutional parts, related parts, and systemic parts. The exact meanings of different types of relation are often subtle and may vary from common usage, and thus correct interpretation depends on them being fully understood. The history of how different relation types have been used within the FMA is also important; systemic partition is an older conception of how structures are divided into parts that is incomplete and has been discounted in favor of related and constitutional partition. A user who encounters systemic part relations attached to some entities and not others must understand this lest they interpret the absence of these relations on the latter entities to mean that they have no parts of that sort. The subtlety of these definitions has been found to be an obstacle to the use of the Anatomy Explorer by non-ontologists, as discussed in section 4.7.2.

Knowledge is loaded into the Explorer one entity at a time whenever the user "explores" that entity. Exploration occurs in response to the user selecting an entity by clicking on it in the scene, following a relation from another entity by clicking on an entity name in the information listed for that other entity, or searching for it by name. When an entity is explored, all information about that entity is loaded and presented in a panel onscreen, with relations organized into groups, each presented in its own subpanel. All relations involving the currently explored entity are shown, regardless of whether it is the subject or the object.

Though the Anatomy Explorer is primarily intended as a frontend to the FMA, it could easily be adapted as a frontend for other ontologies, including non-medical ones, provided that they can be accessed using the Query Integrator.

4.6.1 Usage

Almost all user interaction with the Anatomy Explorer is through a single page JavaScript application. The sole exception is an initial configuration web page where the asset set and style sheet used in that Explorer session are specified.

Using the JavaScript application, users may:

- Explore an entity, which loads and displays information about that entity. An entity can be explored by:
    - Clicking on it in the scene.
    - Searching for it by name.
    - Navigating to it by clicking on its name in information provided about some other entity.
- Add and remove assets from the scene either individually or in groups; see below for details.
- Save the contents of the scene to the Scene Builder's scene repository.
- Load the contents of a pre-existing scene into the Explorer for browsing.
- Clear the scene.
- Highlight a group of entities that are related to a target entity by a specific type of relation.
- Review the history of which entities have been explored in that session with the Explorer. Users may also move back and forward through this history using the keyboard.

- Drill into an entity, which loads and displays information about the entities that extend from that entity following relations of a particular type.

Full instructions and worked examples of the use of the Anatomy Explorer can be found in section A.3 in the Appendix.

4.6.2 Implementation

The Anatomy Explorer is implemented as a web application written in Java with Spring MVC running on Tomcat. Users interact with the Explorer primarily via a single page JavaScript application built using WebGL, three.js, jQuery, and the CGA. The Anatomy Explorer uses web services to interact with the Scene Builder's scene repository, both to save the entities selected during exploration for use in a scene and to load entities from a scene to facilitate exploration. Server-side code is relatively limited and is primarily responsible for translating requests from the client-side application to API calls on various Anatomy Engine services. The Anatomy Explorer provides no programmatic interfaces for use by other applications.

The Anatomy Explorer's main JavaScript application presents information about anatomical entities alongside a 3D scene in which those entities can be displayed. The asset set and style sheet used to construct that scene must be selected on startup by using a form on a page in the Anatomy Engine's main interface. At present, there is no way to change that selection once made, nor are there any defaults. Unfortunately, this prevents direct invocation and so it is impossible to create a URL that opens the Explorer with a particular entity already explored.

The Explorer's main interface is divided into several parts:

- A small panel in the upper right of the screen, called the search panel, where users can explore an entity by entering its name. The input box in this panel uses the text completion engine to help users find correct entity names.
- A large panel on the right of the screen, underneath the search panel, called the exploration panel, where all information about the currently explored entity is

shown. This panel is often taller than can be shown on the screen and thus incorporates a scroll bar that supports mouse scroll wheel interaction. The contents of this panel are organized into:

  - A header, listing the currently explored entity and its FMA ID.
  - Information about any asset associated with that entity and buttons to control its presence within the scene.
  - A series of subpanels containing relations. Each subpanel contains relations of the same type, along with buttons to control the presence in the scene of assets associated with any entities involved in those relations. By clicking on a subpanel, a user may highlight all entities listed in that subpanel that are currently present in the scene. Relations are shown as a pair of boxes containing the names of entities, joined by an arrow. When interpreting that relation, the leftmost box is the subject, the rightmost the object, and the title of the subpanel it is contained within is the predicate.

- A hidden panel, called the history panel, which lists the entities a user has explored in the current session, providing an easy means for them to navigate back to a previous state. This panel is revealed using the "H" key or by clicking on a button in the lower left of the screen.
- A hidden panel, called the drill panel, which shows the results of drilling, an advanced exploration action described below. This panel appears automatically when that action is carried out.
- A hidden panel, called the persistence panel, through which users can save the Explorer's currently loaded assets into the Scene Builder's scene repository, allowing its use as a non-traditional authoring tool. This panel is revealed using the "S" key or by clicking a button in the lower left of the screen.

The Anatomy Explorer's overall user interface can be seen in Figure 21, while Figure 22 shows a more detailed view of the exploration panel.

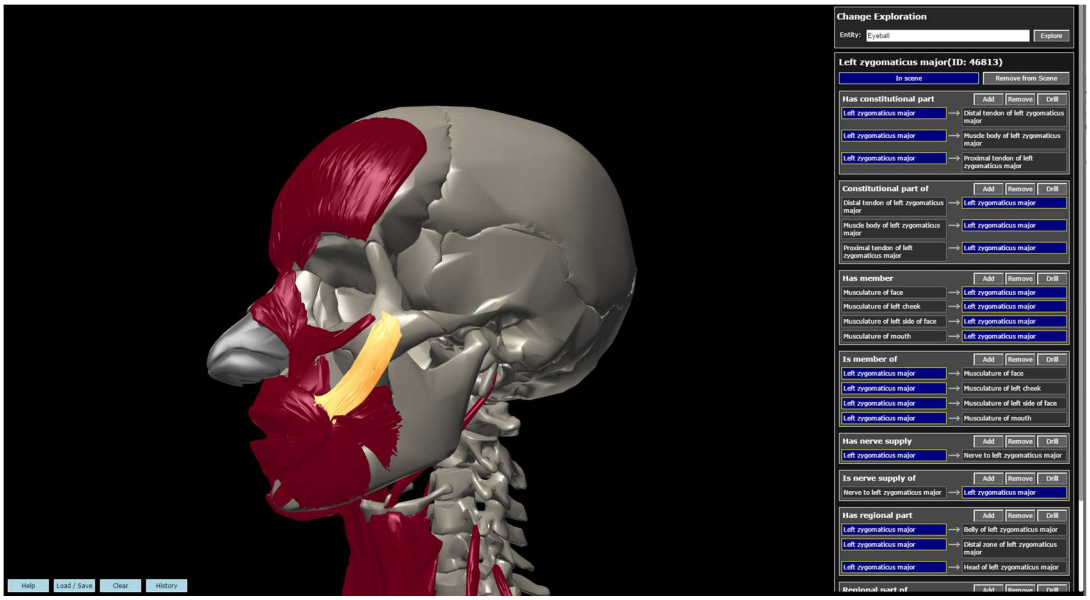


Figure 21 – Anatomy Explorer

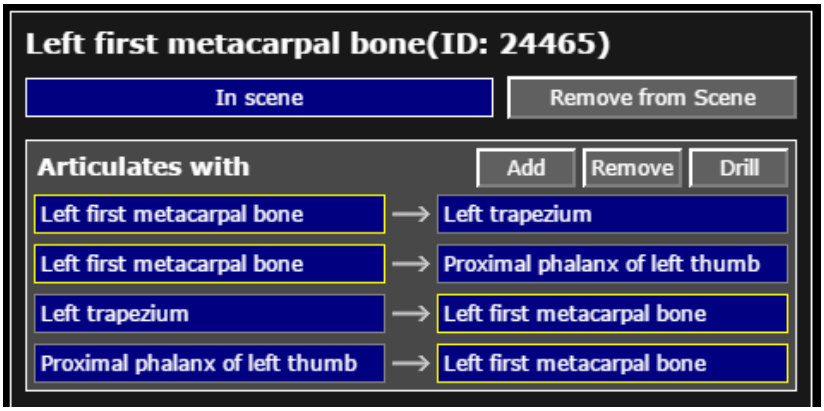


Figure 22 - Detailed UI

The most common user action taken while working with the Explorer is "exploration", where a user chooses an entity they are interested in and information about that entity is loaded into the exploration panel. Exploration can be triggered in several ways, as documented in the section labeled "Usage" above. When exploration is triggered, the following occurs:

- A query request is issued that loads from the FMA all relations involving the target entity as either the subject or the object. This step is skipped if relations have already been loaded for that entity as, when service from the Query Integrator is slow, relation loading can become a major bottleneck.

- The asset representing the target entity, if present in the scene, is highlighted. Any previously highlights are cleared.
- The contents of the exploration panel are replaced with information about the target entity. As a result, only one entity can be explored at any time.

A more advanced form of exploration called "drilling" requires that users indicate both a target entity and a type of relation. This action loads all entities that can be traced back to the target entity by unbroken chains of relations of the target type where the first relation in those chains have the target entity as their subject. Like exploration, drilling uses a query to load entities and relations from the FMA. Once loaded, these entities are presented hierarchically in the drill panel, as shown in Figure 23.

The contents of the Explorer's 3D scene are controlled by the user using buttons in the exploration and drill panels. Assets may be added and removed in three ways:

- Individually, by exploring the associated entity and clicking buttons in the header of the exploration panel.
- As a group, where all associated entities are related to some target entity by a particular relation, by exploring that target entity and clicking buttons in the subpanel for that relation in the exploration panel.
- As a group, where all associated entities can be traced back to some target entity by an unbroken chain of relations of a particular type, by drilling that target entity and relation and clicking buttons in the drill panel.

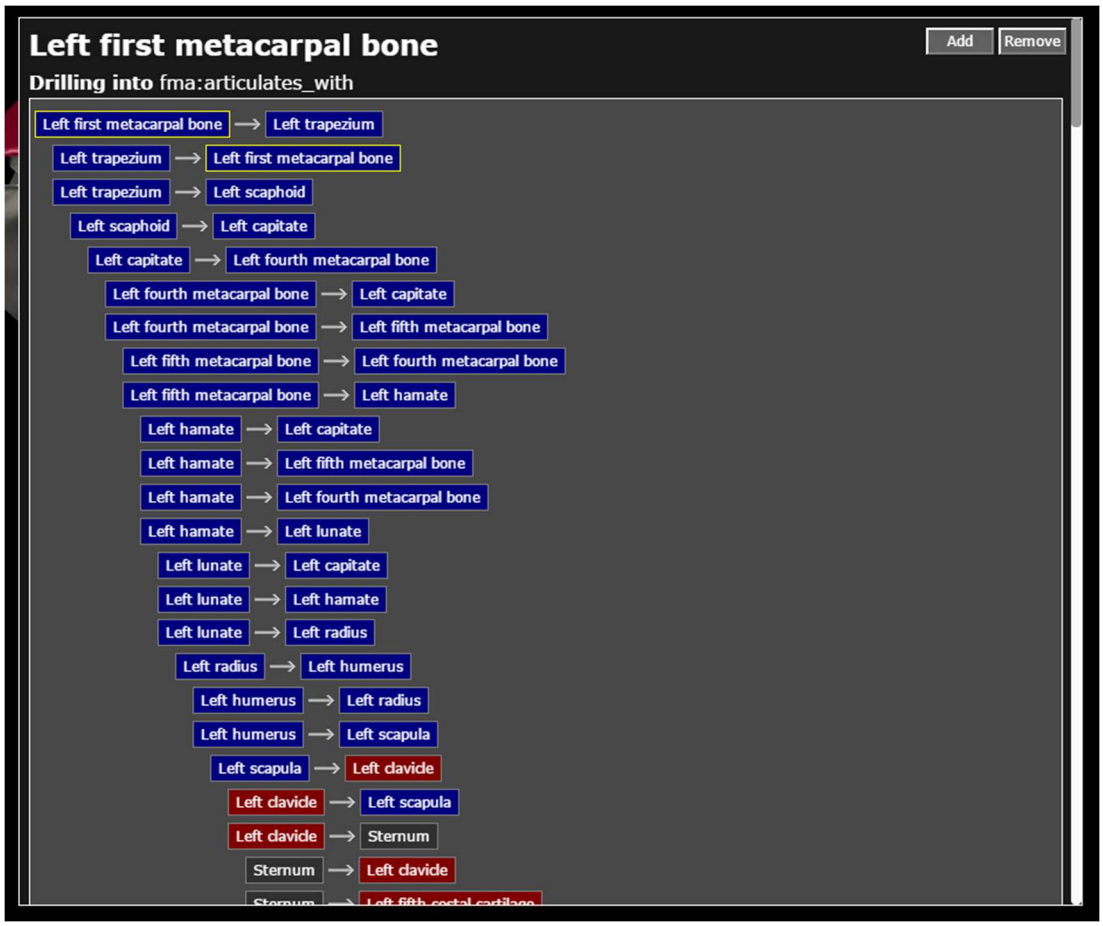


Figure 23 - Drill panel

The Anatomy Explorer has no server-side storage capabilities of its own but may take advantage of the Scene Builder's scene repository to save the contents of the current 3D scene. This allows the Explorer to be used as a somewhat unorthodox authoring tool, where scene contents are specified not by selection from a list or by the direct invocation of queries, but by browsing anatomy using the various relationships between structures in the body. This approach may be more intuitive to some users than direct authoring using the Scene Builder, though this has not been tested.

Exploration starts from scratch at the beginning of each session, but scenes can be loaded from the repository to kick-start an exploration session by offering a set of assets whose associated entities can easily be explored by clicking in the scene. This may be easier for certain exploration tasks than navigating solely via the search and exploration panels.

4.6.3 Further development

The Anatomy Explorer, as a platform, could be extended in many directions. I discuss three possibilities here:

- Support for direct invocation;
- Support for more advanced methods of navigation, such as queries and paths;
- Support for other imaging modalities.

4.6.3.1 Direct invocation

As mentioned above, in order to invoke the Anatomy Explorer, users must first select an asset set and style sheet for use in generating the Explorer's 3D scene. This selection must be made manually and, once set, cannot be changed. This prevents direct invocation of the Explorer from a URL as might be desirable in, say, course materials or a reference document.

One approach to addressing this limit is to use tags, which are already supported by the Anatomy Engine and which may be employed to mark a default asset set and style sheet that can be used in lieu of the initial configuration page. To completely eliminate that page, a configuration panel could be added to the Explorer where users can change which asset set and style sheet are currently used. Changes to the style sheet are simple to apply, requiring only that the display parameters of each asset in the scene be re-assigned. Changes to the asset set, however, are harder, as the entities for which assets are available in one asset set are not necessarily the same as in another. A reasonable approach would be to keep a list of the entities a user has added to the scene, regardless of whether an asset is available for each, and then, whenever the asset set is changed, find all assets in that set that match entities in the list and show those.

4.6.3.2 Advanced navigation

At present, users move between structures by following individual relations, using the drill function to follow multiple relations of the same type, and searching for a target structure

by name. While functional, these methods are slow and can require users to possess more knowledge of the FMA that might be reasonable. More advanced methods are possible:

- Firstly, it is often desirable to move between entities according to a path, defined as a sequence of relations following some pattern. For example, there is usually no direct relation between muscle entities and entities representing the bones they insert and originate on; instead, the muscle entity is related to some part of the bone, and the bone must be found by moving up the part hierarchy from the attachment point. So, to move between ***Biceps brachii*** and ***Scapula***, the bone to which it attaches, the following path must be followed, with * indicating that zero or more instances of a particular relation are necessary: ***Biceps brachii*** **fma:inserts_on fma:regional_part_of*** ***Scapula***. To support navigation by path, a customizable SPARQL query would be required that can be configured within the Explorer by specifying the order and cardinality of the different relation types that make up a path, perhaps using a graphical interface of some kind. Since many users may not be familiar enough with the FMA to specify parts of their own, some set of pre-defined paths may be necessary.
- Secondly, it may be desirable for users to navigate between entities that are related in some way that cannot be described using a path. In such situations, users ought to be able to directly invoke a query whose results can be listed for navigation or used to highlight assets within the scene.
- Thirdly, it may sometimes be desirable to filter the entities shown in either the results of normal exploration within the exploration panel, or within the results of special exploration actions such as drilling or following paths, as described above. A straightforward filtering operation would be to allow users to specify one or more entity types and then show only entities that are instances of that type. Another approach to filtering might be to allow users to specify a second query or relation path that entities must also match before they are shown. A user might use this

functionality to, for example, obtain a list of muscles supplied by a particular spinal nerve root, filtered to only show those that are part of the forearm.

4.6.3.3 Other imaging modalities

The Anatomy Explorer, as currently implemented, shows information about anatomy alongside a 3D scene rendered using surface models. This is sufficient for many purposes, including education, but the visualization of anatomy using other modalities is desirable, particularly for clinical users. For example, in addition to showing the currently explored entity by highlighting it in the scene, an enhanced Anatomy Explorer might indicate it in radiological images or image slices from one of the Visible Human projects. Similarly, matching highlights or lines traced between the image slice and the 3D scene might be used to show the correspondence between different representations of the same structure.

To implement such extensions, the Anatomy Engine would need to support libraries of segmented image slices and provide some kind of browsing architecture that causes the correct image to be loaded when an entity is explored as well as navigation controls that allows the user to move up and down the body. As discussed in section 4.7.2, support for radiological images was a feature requested by participants in the validation study, and so support for such images across the entire Anatomy Engine seems warranted for further development.

4.6.3.4 FMA Editing

At present, the Anatomy Explorer allows users to view information contained within the FMA, but provides no means to edit it. Consequently, when errors are detected, they must be manually reported to the FMA's curator, who may address them by updating the FMA's core representation. These changes are then released to the public representation used by the Anatomy Explorer on a weekly basis.

While it would likely be unwise to allow general users to directly edit the FMA using the Anatomy Explorer, functionality could be implemented to allow users to report problems from directly within its interface.

4.7 Validation

The preceding sections described efforts to address research question A by developing an extensible framework and four sample applications to demonstrate how an ontology can be leveraged to create tools that support educational activities. Before I can claim that these applications are truly useful, however, I must validate them with actual potential users.

In this section, I present the method and results of a validation study conducted on the Scene Builder, Data Visualizer, and Anatomy Explorer. This study was structured on the model of expert review as described in [210], and asked potential users first to work with the tools in a series of worked example activities and then asked to critique them in a semi-structured interview.

Expert review, an inherently subjective and qualitative method, was chosen over more objective and quantitative models of evaluation because it is focused on detailed critique and the generation of ideas for further development. This allows deeper insight into the strengths and weaknesses of the system, and helps lead development in useful directions. Furthermore, the applications described in this section are implemented as prototypes and cannot thus be used to assess the effectiveness of the overall approach each embodies. That is, while substantial effort was expended in their development to make them easy to learn and use, they are by no means polished and perfect examples whose performance and nonfunctional attributes such as likeability, learnability, and efficiency are worth quantifying. Instead, each is a first step towards a useful tool that should be evaluated in terms of questions such as "can you see how this would be useful", "what other features would you like", and "what limitations do you see with it as currently implemented" that rely on the subjective expertise of potential users.

In order to ensure that each study participant was able to speak from a position of familiarity and reasonable proficiency with the applications, a hands-on approach to expert review was taken. Each study session involved three activities, one on each of the three

applications studied, and each consisting of detailed worked examples, as in a tutorial, followed by one or two free tasks, where the participant was asked to complete a task but was not given specific instructions on how to do so. These last tasks were included to force participants to think actively about using the tool rather than just following instructions.

4.7.1 Method

I conducted the study over a four-week period in March and April 2015. Eight potential participants were approached to participate, two from each of four groups of stakeholders identified as having an interest in these applications: two educators, two students, two anatomical ontologists, and two engineers involved in developing tools for communicating information with an anatomical component.

Each participant participated in a single one-on-one session at a time and place convenient to him or her. Two sessions were conducted remotely, using Skype; all others were conducted in person. Participants used either their own computer or a computer provided by the researcher, depending on whether the participant's computer had sufficient graphics capability to run the applications with at least 30 frames per second.

Each session took between two and three hours, and involved three hands-on activities followed by a discussion session. The activities focused on the Scene Builder, Data Visualizer, and Anatomy Explorer, respectively. Each activity was defined by a worksheet containing detailed instructions; these are included in Appendix A. The researcher was present to answer questions and provide guidance throughout the activity portion of the study. Discussion was semi-structured – a list of discussion topics was used to ensure that, at a minimum, each discussion covered the same ground, though, in general, discussions ranged fairly widely depending on the interests of each participant. This list of discussion topics is included in Appendix A.

The study design was reviewed and approved by the University of Washington's Human Subjects Division under application #49798-EB. Consent forms were collected for each participant and are stored in a secure location accessible only to the researcher. Audio

recordings were taken during the discussion portion of each session and were used to assist in interpreting notes taken during those discussions. These recordings have since been destroyed.

4.7.2 Results

The Anatomy Engine was validated with seven users, representing four user groups:

- Two technologists, one an expert in brain-related research and clinical informatics with some knowledge of the FMA, the other an expert in the FMA and query design.
- Two ontologists, one the primary FMA curator, the other an anatomist with experience contributing to the FMA as well as teaching anatomy to medical students.
- Two instructors, one teaching neuroanatomy to medical students, the other teaching pre-med anatomy to undergraduates. Neither had prior experience with the FMA
- One third year dental student, with anatomy learned through the UW's gross anatomy class and various dental specialty classes. An additional student was to be included but was unable to participate at the last minute due to other obligations.

Not all participants completed all three activities. Some had tight schedules that limited the time available, while others worked through the instructions too slowly or spent a lot of time asking questions. When time was short, I helped participants by explaining instructions, having them skip less important tasks, and demonstrating some features myself. One was unable to attempt all three activities due to time constraints; I demonstrated the Data Visualizer to them instead. In general, participants did the Scene Builder activity first, followed by the Data Visualizer and then the Anatomy Explorer. However, in some cases, the Data Visualizer and Anatomy Explorer were switched in order.

All of the participants reported that they saw at least some value in all three of the applications. Some participants focused comments on improvements to usability and the

technical attributes of the applications, while others focused attention on desired functionality and potential uses of the system. Though no participant thought any of the applications was useless, some questioned whether other existing tools, such as the Zygote Body, could perform similar functions. Most participants came to understand and appreciate the role of the FMA in supporting the applications.

Participants proposed a wide variety of modifications, ideas for further development, and adaptations to fit particular use cases. These proposals and issues are presented and discussed in the subsections that follow. Overall, the message gained from running this study was that:

- These applications are useful and are of value and interest to members of all four stakeholder groups.
- Both the Scene Builder and Anatomy Explorer could be deployed immediately to support education in their current form.
- The primary limitations of these applications, as currently implemented, are model detail and some user interface issues, though the currently available models are sufficient for undergraduate anatomy and certain specific medical school applications.
- All three applications are promising and offer fertile grounds for further development and customization.

4.7.2.1 Scene Builder

In general, participants found the Scene Builder to be straightforward and immediately useful, though a variety of criticisms and suggestions were made.

4.7.2.1.1 Defining scene contents

Participants found the idea of using queries to select assets easy to understand at a high level, but were often confused when asked to select a query themselves. Several participants expressed doubt that they would be able to correctly select and use queries without significant documentation and time to experiment. This is concerning, but perhaps

unsurprising given that the use of queries relies on understanding, to at least some extent, the organization of the FMA and the meaning of its different relation types. For example, a query entitled "Articulates with (FMA Name)" is not particularly clear to users unless they understand what articulation means in the FMA and what an FMA Name is. Though the user interface provides descriptions for each query, users did not tend to read these thoroughly, and instead just followed instructions rather than thinking about queries for themselves. Most agreed, however, that with a bit more time to get used to the interface, they could use queries to create at least basic scenes.

Another concern related to query fragments was that entities often had names that they would not otherwise use. For example, adult teeth are listed in the FMA under "Secondary Dentition", which two participants found unintuitive. Participants did, however, praise the text completion service as being very helpful in finding the correct name when configuring a query. Finally, users felt that they would rarely, if ever, use queries that require FMA IDs instead of FMA names. This makes sense – it would be unreasonable to expect users to remember the numeric IDs that the FMA associates with entities. Queries with FMA ID parameters, then, are primarily useful when called programmatically, as by the Data Visualizer.

Participants found list fragments easier to understand. The usability of the list interface received criticism, however, particularly if the number of assets available to choose from was large. Multiple participants wondered if a search filter of some kind could be used to decrease the number of assets shown at once; for example, entering "Left" into a text box at the top of the list to eliminate those assets whose names did not contain "Left".

Participants found the interface for showing and hiding entities within a scene intuitive to use, though time-consuming in certain cases. In particular, participants requested the ability to hide or show all entities in a single fragment with a single button press.

4.7.2.1.2 Display parameters

Some participants found the interface for adjusting asset display parameters confusing, as it allowed them to control all six parameters of the Phong shading model, five associated

with lighting and reflection, and one to define the level of transparency. Based on ideas from several participants, a better approach would be to allow users to choose from a gallery of pre-defined material templates that could be customized using a single color control. For example, by selecting a template called "glossy" and configuring it with the color blue, the system would generate display parameters to make a surface with those characteristics. Full color controls could still be made available to advanced users via a hidden panel.

Some participants went the other way, suggesting that the system might support advanced rendering using vertex shaders. These would be pre-defined and presented in a gallery along with basic material templates, and would allow users to make assets appear to be made of different materials such as glass, stone, and so forth.

Participants also requested usability improvements such as wider slider bars, the ability to reset display parameters to those defined by the base style sheet, and the ability to select multiple assets and apply display parameters to them all simultaneously.

4.7.2.1.3 Lighting

Two participants asked for the ability to control lighting within the scene. By default, each scene has two light sources, the first placed by scaling the maximum X, Y, and Z coordinates of the scene's assets, the second diagonally opposite. Users might add or remove light sources, change their location, or adjust the color of the light they give off.

4.7.2.1.4 View Management

Participants generally liked and found useful the ability to specify a scene's default viewpoint, but requested three additional features:

Firstly, participants wanted to be able to create multiple viewpoints of the same scene. This was particularly the case in creating scenes for use in course materials, and one participant emphasized that the ability to see anatomy from multiple angles was one of the main strengths of 3D for teaching anatomy that should therefore be better supported.

Additional viewpoints would be specified in the Scene Builder and made available for users to choose between by an extension to AVA.

Secondly, most participants wanted to be able to easily move to key views corresponding to the different anatomical directions. Alongside this feature, participants asked for some kind of 3D compass that would show the current orientation of structures with respect to the rest of the body. In the Biolucida project, all anatomical scenes were shown with a compass consisting of a skeleton in anatomical position that rotated along with the rest of the scene so that users could easily interpret the angle at which they were viewing the main scene from; such a feature could be adopted.

Finally, participants wanted to be able to associate labels and text with different views so that they could describe to users what they were currently seeing. These labels would then be presented in whatever user interface was used to change between viewpoints.

4.7.2.1.5 In-scene labels and text

Some participants, particularly educators and the student, suggested that it would be useful to be able to label individual assets and asset features in the scene using pins. Different ideas were suggested for how this might look. A basic implementation might include labels floating in the scene with lines connecting to particular points, while more advanced implementations might allow labels to be hidden or triggered from outside the scene, using JavaScript. Also desirable was the ability to include text resources directly in the scene, perhaps with links or references to labels within the scene itself. Further research is required here to determine the most useful and intuitive ways of including such content.

4.7.2.1.6 Exporting and Deploying scenes

Participants universally praised the fact that resources created with the system could be embedded within other web pages, as this allowed them to be integrated with other learning resources. Furthermore, some participants were particularly impressed with the Scene Builder's ability to export scenes so that they could be re-deployed on other servers,

independently of the Anatomy Engine. One educator went so far as to describe these two features as "absolutely key" and said that a major drawback of previous systems was the fact they were often only available within research environments that tend to be both unreliable and unstable.

4.7.2.2 Data Visualizer

Of the three applications studied, participants received the Data Visualizer the least positively. Several remarked that while they thought the Data Visualizer would be useful to some users, it was not particularly useful to them. This is perhaps unsurprising, as data visualization is an activity largely performed by researchers and clinical practitioners, none of whom were included in this study. Nonetheless, some positive remarks were made, and several useful suggestions were gathered.

4.7.2.2.1 Visualization methods

Several participants were concerned that the Visualizer was only able to visualize data at the level of individual assets, and suggested that this would limit use of the application for true data visualization, particularly with the relatively macro-scale assets currently available. To stimulate discussion, the point marker and texture generation approaches mentioned in section 4.5.3 were described to participants for comment. Most found these approaches interesting and thought they had promise but were not particularly inspired by them, saying that they had trouble determining how useful they would be without an example. Again, participants cited the fact that data visualization was not a major part of their jobs. One technical participant, with an interest in the visualization of neurological data, thought both approaches were worth exploring, with texture generation a higher priority than point markers.

4.7.2.2.2 Defining style sheets

Participants varied in how interested they were in the ability to define new style rules using expressions in the Style Manager. Technical participants found the ability interesting and suggested various expansions on the types of expression supported, whereas non-

technical participants were generally uninterested and wanted to just be able to use rules that had been pre-defined for them. Participants of both types thought that individual style rules ought to have labels as well as expressions, as the current practice of using the style rule alongside color swatches in the legend panel was thought to be generally confusing. Similarly, the ordering of styles in the legend panel was criticized as being unclear; particularly as the expressions that make up style rules do not sort well alphabetically. Two suggestions were offered for how styles should be ordered: alphabetically according to the label requested above, or automatically, matching the order of application defined in each style sheet. A third option is to allow style sheet authors to explicitly provide a listing order.

4.7.2.2.3 Educational usage

Both the student and educators liked the ability to rapidly switch between style sheets, as it allowed them to compare the different ways structures in the body can be classified. However, they felt that neither the Data Visualizer nor AVA, the viewer, were quite the right fit; the Data Visualizer because it emphasized authoring and the viewer because it lacked several desirable features. In particular, they praised the Visualizer's legend panel, and suggested that a similar feature should be implemented in the viewer for scenes created using the Scene Builder.

As with the Scene Builder, most participants suggested that it would useful to be able to assign visualizations a title and description as well as allowing them to include additional text and in-scene labels. They again requested more advanced view management, including the ability to set viewpoints for visualizations rather than simply relying on the viewpoints in the underlying scene. Finally, participants thought that a visualization ought to be able to be configure with multiple style sheets so that users could switch between them in the viewer rather than having to load a separate visualization for each.

Based on participant comments, it seems that the Data Visualizer addresses two types of need. Educators want a tool to create visualizations to explain different ways the body is categorized and divided, while researchers want a tool that allows them to explore and

communicate anatomically indexed data. As currently implemented, the Data Visualizer better serves the latter need, but could be forked or modified to create a version better suited to education.

4.7.2.3 Anatomy Explorer

Overall, participants found the Anatomy Explorer to be the most impressive of the applications studied, in large part because of the sheer amount of information it makes available. Some initially found it intimidating, and several had trouble understanding it fully, though all acknowledged that the general idea made sense and felt that they simply needed more time with it. Educators and the student were particularly quick to dive in and explore how particular knowledge was represented in the FMA. One participant remarked that they had never really understood what the FMA was about until they saw it in the Explorer.

After coming to grips with the Anatomy Explorer's structure and interface, most participants saw some use for it in their field: educators wanted to use it to support problem-based learning, students to consult it as a reference library, and ontologists to validate their work and identify missing information. Technical participants had no direct uses for the application, but were interested in how it might be adapted and extended to other ontologies.

4.7.2.3.1 The FMA and its complexities

While participants were impressed with the scale of the FMA as seen through the Anatomy Explorer, exploration inevitably revealed inconsistencies and missing information. This was by no means an indictment of the FMA – all participants recognized the immense and virtually never-ending effort involved in constructing it. The ease with which errors were found, however, suggested that the interactive and visual nature of the Explorer makes it particularly suited to searching for holes and errors, in order to validate the ontology. The FMA's primary curator, who participated in this study, took notes as he worked and stated that the Anatomy Explorer provided an excellent way for him to review his work and asked if he could begin using it for that purpose.

The experience of participants with the Anatomy Explorer emphasized how alien and precise the FMA is compared with how anatomy is described colloquially. Participant comments noted a similarity between the formal expression of knowledge in the FMA and the formal way anatomy is described in a medical context, suggesting the possibility of using the FMA to teach skills of thinking and talking about anatomy in addition to conveying the declarative knowledge contained within.

Participants found the FMA's formality confusing in several ways:

- Some participants found it difficult to interpret relations, both in terms of understanding the precise meaning of each type of relation and in terms of correctly reading the direction of a relation. More than half of all participants stopped to read relations aloud from the exploration panel before acting on them, several made errors in interpreting them, and a couple got lost while following them and had to begin again from the search panel. Several ideas were suggested: a guide or help button to explain the meaning of particular relation types, improved UI to make relation names stand out more, and a verbal representation of relations rather than the symbolic one currently used.
- Participants were often unclear as to which entities ought to be related to each other. In some cases, this was due to the way the FMA relates structures very precisely, leaving broader relations to be inferred; for example, muscle origin and insertion on bone is represented by a relation between muscle entities and entities that represent particular attachment points rather than entities that represent whole bones, leaving that more macroscopic relationship to be inferred by considering bone partition. In other cases, users were confused by the way the FMA groups entities together. For example, users found the various groupings of muscles in the arm difficult to understand as there are several hierarchical groups whose names are difficult to interpret.
- Participants had trouble working with lateralized and general entities, where a given anatomical structure might be represented in multiple ways. For example, three

entities represent the femur bone – a general entity ***Femur***, and two lateralized entities ***Left femur*** and ***Right femur***. In particular, they had trouble determining which relations applied to the general form and which to the lateralized form. For example, muscle connectivity is defined on the general entity, while blood and nerve supply is on the lateralized entity. Participants often could not find relations on one entity, and found it confusing to have to switch to the other to find what they needed. This distinction makes sense from an ontological standpoint, but is confusing to users who are not familiar with that way of thinking. The Explorer, however, is a good platform for learning about such distinctions.

4.7.2.3.2 Scene construction and browsing

Once they understood how to navigate between structures and interpret relations, participants found scene construction in the Anatomy Explorer comparatively easy.

A major limitation, however, was the fact that assets were only available at a single level of granularity and that level varied from structure to structure. Many muscles, for example, are represented with a single asset while others are represented as several assets, each a different belly or head of that muscle. For example, there is no asset for ***Left biceps brachii***, but there are assets for ***Short head of left biceps brachii*** and ***Long head of left biceps brachii***. As a result, it is impossible to create a scene showing all muscles of the arm by simply adding assets for all entities with a relation **fma:member_of** ***Musculature of left free upper limb***.

This limitation emphasizes the need for improved browsing and navigation tools whereby users might enter patterns to traverse multiple relations at once. In this case, the pattern ***Musculature of left free upper limb*** **fma:has_member* fma:regional_part*** would deliver a list of the entities needed to create a scene showing the musculature of the left arm.

Merely providing the ability to specify patterns, however, would not be enough – some library of pre-existing patterns would be required so that non-specialist users could navigate without needing to understand the FMA's structure in depth.

A related feature idea that arose from discussion with one participant is the ability to look up information about how a pair of entities is connected. In this case, the user would identify two entities, and a query would be issued to identify all paths between those entities. Various limitations would need to be imposed on such a query to prevent cycles or extremely long paths, but such a feature would allow users of the Explorer to answer questions of the form "How is X related to Y", which, according to this participant, is of relevance to students.

4.7.2.3.3 Usability

Though all participants eventually became comfortable with the Explorer's interface, some suggestions were made to improve usability:

- Information in the exploration panel is very dense, which makes it difficult to track what one is looking at when content is scrolled. This could be addressed by increasing the space between content elements, introducing stronger visual markers between elements, or changing the shade of adjacent elements as is often done in lists.
- The rationale behind the ordering of subpanels within the exploration panel was unclear to users. This is because subpanels are ordered according to the IRI assigned to that subpanel's relation type, while subpanel titles are human readable phrases describing it. IRI ordering was chosen over ordering by label as it tends to keep inverse relations close together; for example, the inverse pair of relation types Has regional part and Is regional part of appear next to each other when ordered by IRI, but are far apart when ordered by label. Neither scheme is ideal, so a more sophisticated method may be necessary. One option might be to define an explicit ordering that groups similar types of relation together; for example, grouping relations related to partition, spatial relations, and relations related to connectivity.
- The search panel is currently titled "Change Exploration" and has a button labeled "Explore". Both labels are accurate within the nomenclature defined by the Explorer, but users did not find their meaning obvious and several suggested

changing the panel's title and button label to "Search for Entity" and "Search", respectively.

- Both the history panel and the ability to move back and forth through previously explored entities are features that previous users had asked for frequently before they were implemented. Participants in this study, however, barely used them until prompted, often in response to frustration, despite being introduced to them in the instructions. Once reminded, participants found moving back and forth with the keyboard to be both easy and natural, suggesting that the problem was one of prominence, not of functionality. Possible ways to improve the situation might be to increase the size of the history button or add prominent back and forward buttons to the exploration panel, with tooltips indicating the keyboard shortcuts.

4.7.2.4 Other

In addition to the issues described in the subsections above, participants gave comments and suggestions concerning the framework overall.

4.7.2.4.1 Model quality

Overall, participants found the quality and detail of the models used in the various applications to be satisfactory or better, with several participants describing them as impressive. Typically, participants with more experience with 3D models had more moderate opinions. When asked about the granularity of models, most participants thought that models representing smaller structures would be desirable. Both educators and the student thought that the current models were suitable for undergraduate anatomy education, as well as general medical education in specific domains.

Participants also agreed that models and markers to represent anatomical spaces, foramina, and other immaterial anatomical entities were desirable, as well as models to represent different layers of some structures, such as the fascial layers of the skin or the various mater layers of the brain. In addition, both educators and the student emphasized that the models of bones, in particular, should include all named features, particularly

foramina, as knowledge of these passages through the bone is essential to understand the path of many smaller nerves and blood vessels.

4.7.2.4.2 Integration with image resources

Participants confirmed the assumption, discussed in section 4.6.3.3, that it would be desirable to be able to create content using image slices such as CT, MRI, or photographs from the various Visible Human projects. A number of possible applications are apparent:

- One participant suggested that it would be useful to be able to link anatomical entities in the 3D scene with histological images of the appropriate tissue type. This participant suggested that these images could appear in the viewer, but a more robust and extensible approach would be to extend AVA such that JavaScript events could be triggered when a user clicks on an entity in the scene. Then, AVA could be embedded within a webpage with scripts listening for those events to trigger loading and displaying content about that entity, potentially including histological images, text, and dissection photographs. This would allow AVA to be integrated much more closely within a larger information resource than is currently possible.
- An image slice can be incorporated into a 3D scene by plotting it on a plane that appears in the scene at the appropriate position to align its contents with the assets in that scene. Multiple slices can be incorporated by, for example, allowing the user to browse between slices, showing only one a time. Since, however, it is not possible to align image slices and 3D models perfectly if they are not derived from the same source, it may not be ideal to incorporate slices in a scene in this way. An alternative would be show image slices adjacent to the scene and trace lines between assets in the scene and points on the image, perhaps highlighting parts of the image to indicate particular structures. If needed, a surrogate plane could be drawn into the scene to indicate the approximate location that the image slice corresponds to. Again, users could browse between image slices by using arrow keys to move up and down the body or, perhaps, by clicking in the scene. Figure

24 shows a prototype extension to an early version of the Anatomy Engine demonstrating how this could look.

- The approach behind all three of the applications considered in this study could, in theory, be applied to create an image slice browser server similar functions. A Scene Builder equivalent could be used to identify structures to be marked in image slices within some browser, while equivalents to the Data Visualizer and Anatomy Explorer could show information highlighting certain regions in those slices, again accessible via an image browser. A major hurdle to achieving this, however, is that metadata must be available for all image slices, associating regions within each of those images with FMA entities so that the appropriate highlighting could be applied. That said, a few segmented image slice sets are available; for example, the Korean Visible Human incorporates approximately 8500 image slices, with over 1000 structures segmented and identified [211].

Unfortunately, as currently implemented, the Anatomy Engine does not support image slices. That said, it could be adapted with some effort by, for example, treating image slices as a different type of asset with mappings between regions on those images to particular, and then working with styles and queries in much the same way. An image browsing viewer would need to be implemented, but there is little reason that ontology-driven methods could not be applied to image slice libraries as well as 3D scenes. This would be a promising direction for future research.

4.7.2.4.3 Volume rendering

The Anatomy Engine uses 3D scenes rendered using surface models. This approach is widely supported, easy to understand, and straightforward to apply using libraries such as OpenGL, WebGL, and X3D, and is thus the method used in most modern 3D applications such as, for example, games, movies, and virtual reality.

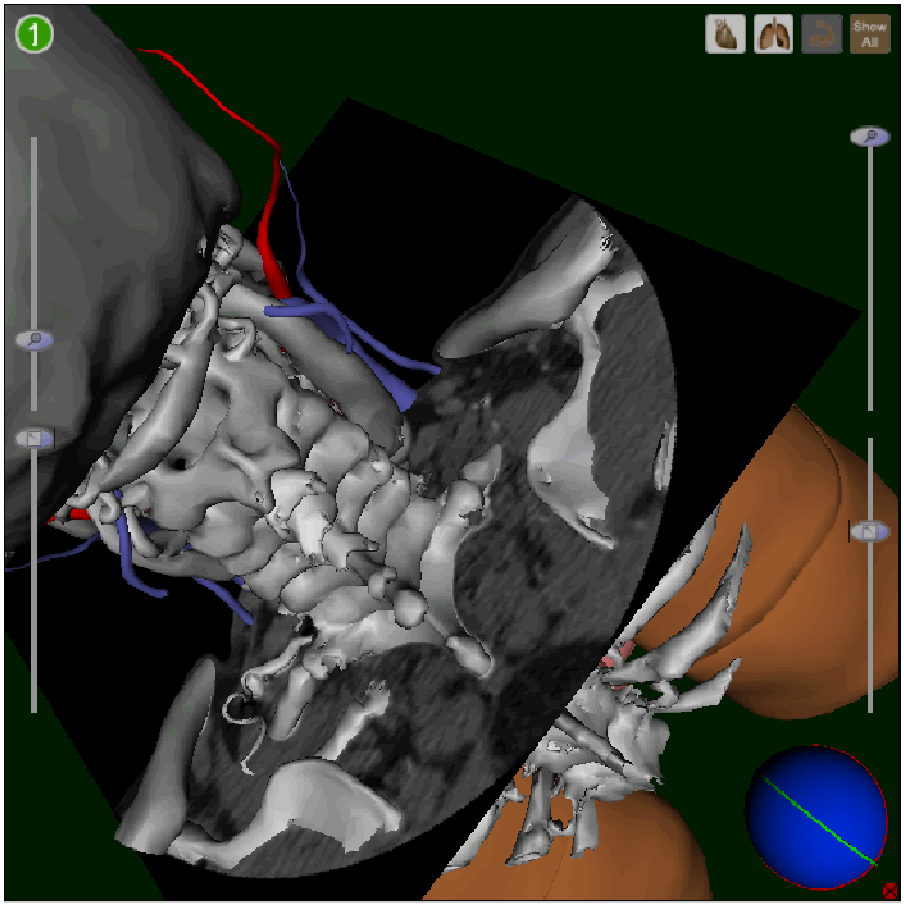


Figure 24 - 3D scene layered over CT slices

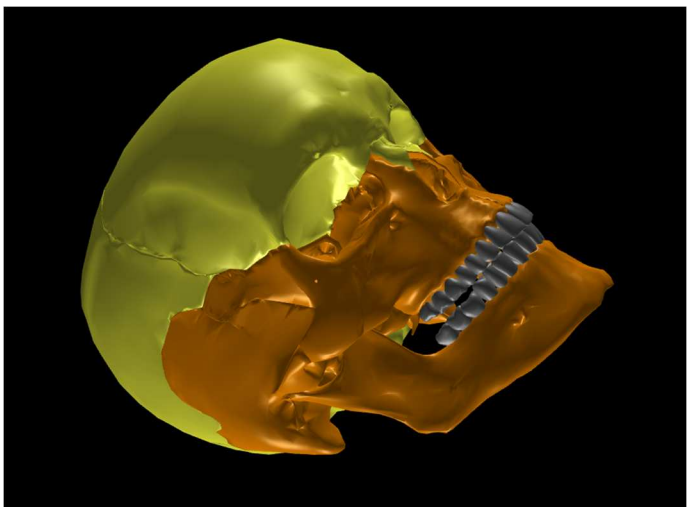


Figure 25 - Surface model scene, labeled by bone types

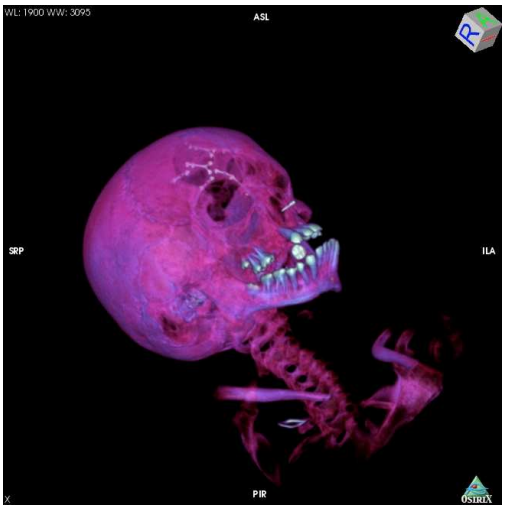


Figure 26 - Volumetric scene, from Osirix Phenix data set

An alternative approach, called volume rendering, is used widely in radiology to render the 3D data collected using CT, MRI, and other imaging techniques. In this approach, an imager measures density or some other response from each point in a three-dimensional grid overlaying the body. These values, called voxels, are then rendered in a scene using some mapping function to determine how transparent and what color each voxel should be. Figure 25 and Figure 26 show the skull rendered using surface and volume techniques, respectively.

In principle, the ideas behind the Anatomy Engine could be applied to volume rendered content by, for example, generating a mapping function to highlight or reveal particular structures. A key impediment to doing this is the fact that volume rendering is much more computationally intensive than surface rendering, in large part because surface rendering is commonly supported by dedicated graphics hardware, though, no tools yet exist for deploying interactive volume rendered scenes on the web. Nonetheless, volume rendering has several advantages over surface rendering:

- Volume rendering is capable of showing internal structure by making certain voxels fully or partially transparent, by allowing users to dynamically adjust the mapping function to change the visibility of different layers, or using a clipping plane to cut through a structure to see it in cross-section. Surface models do not support this, though multiple surface models can be used together to create a similar effect in limited circumstances.
- Volume rendering can be applied directly to the data produced by imaging techniques such as CT or MRI, without any additional processing. Surface rendering, on the other hand, operates on mesh files that can be generated from volume data once individual structures have been segmented out from it. Though volume data can be rendered without segmentation, however, it is not useful from the perspective of the Anatomy engine until it is, as it is not otherwise possible to identify which portions of a scene correspond to which structures.

- Volume rendered scenes tend to look more realistic than surface rendered content. This is due to a number of technical factors, but is also due to the fact that volume data is usually extracted from a real human and thus shows many irregularities and much detail that is smoothed away when surface models are made.

4.7.2.4.4 Extensions to AVA

A number of suggestions were made for improvements to AVA, including:

- The ability to display a scene title and some amount of descriptive text.
- Text labels that hover within the scene with arrows traced from individual labels to particular 3D points, much as pins and labels are used in an anatomical dissection.
- A way to link assets within the scene to URLs or JavaScript events, so that users can click on a structure to, for example, load more information about it from, say, Wikipedia. Support for this feature is already implemented in the CGA, but is not supported by any applications, including AVA.
- Support for more advanced view management, including descriptions for each viewpoint and the ability to select between multiple viewpoints, perhaps from a list or drop down.
- A mechanism to help users understand the scene's orientation with respect to standard anatomical views, and the ability to apply those views. A compass of some kind could be used.
- A legend panel, similar to that implemented within the Data Visualizer, so that users can easily interpret the color of different assets.

4.8 Conclusion

This chapter sought to answer research question A, re-stated below:

> **How can an ontology be used to support the creation of content for learning tools and activities?**

To answer this question, a software framework called the Anatomy Engine was presented, along with several applications built using it. This framework employs the FMA, an ontology representing knowledge about human anatomy, to organize and give meaning to 3D models, label those models using various schemes, and provide knowledge that can be interrogated using queries to define how those models are used as content within arbitrary educational applications.

Four applications built using the Anatomy Engine were presented. Of these, AVA is a viewer utility employed by the other three, while the Scene Builder demonstrates how the FMA, working through the Engine, can be used to support content authoring. The Data Visualizer applies it to transform research and clinical data into anatomical visualizations, while the Anatomy Explorer illustrates how an ontology can be used as the basis of an interactive reference tool.

Finally, a validation study was presented, in which these applications were examined by expert participants belonging to four potential user groups. These experts found that the applications could have direct value in their work and suggested a variety of extensions, improvements, and new application ideas.

4.8.1 Future work

The results presented here leave open many possible avenues for future work, including:

- Continuing development and completion of the FMA, as discussed in 4.7.2.3.1.
- Improved tools for query authoring, as well as query search and re-use.
- The completion of a number of extensions to these applications, as documented in the subsections labeled "Further Development" throughout this chapter.
- The implementation and evaluation of a number of potential applications based on the framework, as documented throughout section 4.7.2 and in Chapter 6.
- The acquisition of new models.

- The development of methods for handling model sets that represent anatomy at multiple granularities.
- The extension of the Engine to other ontologies, both in medicine and other domains.
- The extension of the Engine to support content based on libraries of image slice and volume rendered, as discussed in sections 4.7.2.4.2 and 0, respectively.

Chapter 5. Ontology driven education

It has long been known that students who learn through one-on-one engagement with a human tutor perform substantially better than those who learn in other settings, with the difference in average performance found to be as large as two standard deviations [23]. Human tutors aid students in several ways, including by:

- Customizing learning activities to balance difficulty with ability [25];
- Scaffolding learning and building confidence by confirming correct actions and gently reminding them of these in the face of mistakes [26];
- Providing personalized evaluation that supports the ability of students to assess their own progress.

Each of these functions has, to some extent, been implemented within research software in the field of intelligent tutoring systems; see [212] for an overview. In addition, each function involves domain knowledge to some extent, and could therefore benefit from an ontology-driven approach.

This chapter answers research question B, re-stated below:

How can an ontology be used to support tutoring?

To do so, a tutoring framework is presented that illustrates three roles an ontology may play in supporting the tutoring functions described above. This framework is intended as an early prototype, and while it is adequate for the purpose of demonstrating the potential of ontology-driven education, its performance is hardly optimal and it cannot hope to compete with cutting edge systems that have been built and refined over many years of research. The rhetorical goal of this chapter is to propose and demonstrate the potential

of ontology-driven tutoring as a new approach that complements and can be leveraged by these existing methods.

The framework consists of three components:

- The Curriculum Manager, a tool for managing curricula created from knowledge within the FMA and made available to other components as a domain model.
- The Student Model, a probabilistic representation of estimates of student knowledge, constructed on demand from a curriculum specification and the FMA.
- The Assessor, a web service that assesses assertions about anatomy against the FMA's canonical knowledge base.

An additional component, the Recommendation Engine, discussed in section 7.2, creates an educational feedback loop by using the results of student activity, assessed and modeled using the components above, to guide students on to future activities. Though it is properly considered a tutoring component, as its primary function is to provide students with guidance, its discussion is postponed to chapter 7, as it is not itself ontology-driven and, furthermore, ties in closely to the broader learning activity framework described there.

The fact that names are given to these components in the list above should not be taken to suggest that they are separate standalone applications, connected only through public APIs. They are not. The Curriculum Manager and student model, in particular, are tightly coupled, being essentially different interfaces onto the same data representation. The Assessor and Recommendation Engine directly invoke both the Curriculum Manager and the student model, to assess a student's progress and update the model, and to determine which activities a student should do next, respectively.

Classically, intelligent tutoring systems are built around the trinity model, discussed in section 2.6.1, where the overall problem of how to provide tutoring support is broken down into three sub-problems that can be addressed individually: a domain model that represents the material to be taught, a student model that describes an individual student,

and a tutoring model that embodies the various logic and functionality required to guide learning.

This model applies broadly to the software components described in this section: curricula created with the Curriculum Manager provide domain models, which are built on to create the student model. The Assessor performs basic tutoring functions that are built on by the Recommendation Engine described in chapter 7, while more advanced scaffolding and tutoring support must be left to the user interface of specific learning activities, as described in the example game concepts of chapter 6.

Each of the three ontology-driven components is presented in one of the sections below, with information on usage, implementation, and various avenues for further development. A final section concludes the chapter with a short summary and an overview of possible future work.

5.1 Curriculum Manager

Before a human tutor can teach a subject, he or she must first understand it. The same applies to software tutors, which must have access to some formal representation of subject matter that can be used to pose problems and assess student actions. In the trinity model, this representation is called the domain model.

Domain models represent subject matter in a variety of ways, as discussed in section 2.6.1.1. The FMA is an example of one type of representation, an ontology, and, as such, it could in principle be used as a domain model within tutoring applications. Unfortunately, however, its sheer size makes it far too unwieldy to be incorporated directly for this purpose. To address this problem, the Curriculum Manager provides a means for educators to create much smaller ontologies from the FMA by specifying the entities and relations that are relevant to particular educational activities. Once specified, these smaller ontologies, called curricula, can be used by game and activity templates to generate problems, select content, and interpret student actions. In addition, they provide a foundation on which student models can be constructed.

There are several benefits to using the FMA, a reference ontology, to create a domain model instead of allowing educators to create their own representations:

- Firstly, it requires much less effort, as educators must simply select relevant entities and relations rather than defining them themselves.
- Second, it reduces the chance of errors. Reference ontologies such as the FMA are subject to ongoing review and maintenance that, arguably, should result in them having far fewer errors than other representations might.
- Thirdly, it reduces variation between domain models. Even if domain models created by individual educators were to use some standard format, their content would nonetheless vary, both in terms of how entities are defined and named, or in terms of how the relationships between entities are articulated. The FMA ensures that all curricula use the same library of concepts.
- Finally, the FMA affords interoperability, as it uses a standardized vocabulary of labels for entities and relations that can be interpreted by other tools such as the tutoring components described elsewhere in this chapter, the content generation tools presented in chapter 4, and learning activities such as those proposed on chapter 6.

In essence, the Curriculum Manager treats the FMA as a complete domain reference that can be interrogated by educators to create smaller, more manageable domain models on demand for particular activities.

5.1.1 Usage

Educators interact with the Curriculum Manager through a series of dynamic web pages within the same overall web application as the Anatomy Engine.

From these pages, users can:

- List existing curricula.
- Add, Edit and Delete curricula.

- Add, Edit and Delete curriculum items within a curriculum.
- Add FMA entities to a curriculum item by:
    - Selecting them one at a time from a drop down,
    - Running a query,
    - Checking boxes in a list of entities for which assets are available in some asset set.
- Remove FMA entities from a curriculum item.
- Add and remove FMA relations from a curriculum item.
- Specify whether entity names and shapes are to be included within a curriculum item. Though it may seem that this information should always be included, a distinction exists between activities that teach students to recognize and identify particular structures, and activities that teach information about those structures; for example, a game about connecting blood vessels.
- Enrol students in a curriculum, triggering the creation of a student model, as described below in section 5.2.

Once created, curricula can be accessed by other components directly in code or via a web service API. Game and activity templates, in particular, would be expected to be implemented client-side and use this latter interface.

Changes may be made to a curriculum that already has students enrolled in it, in which case the relevant student model is extended or trimmed as appropriate. Educators making changes to a curriculum that students are enrolled in should probably warn those students, as changes can significantly affect the scores calculated within the student model.

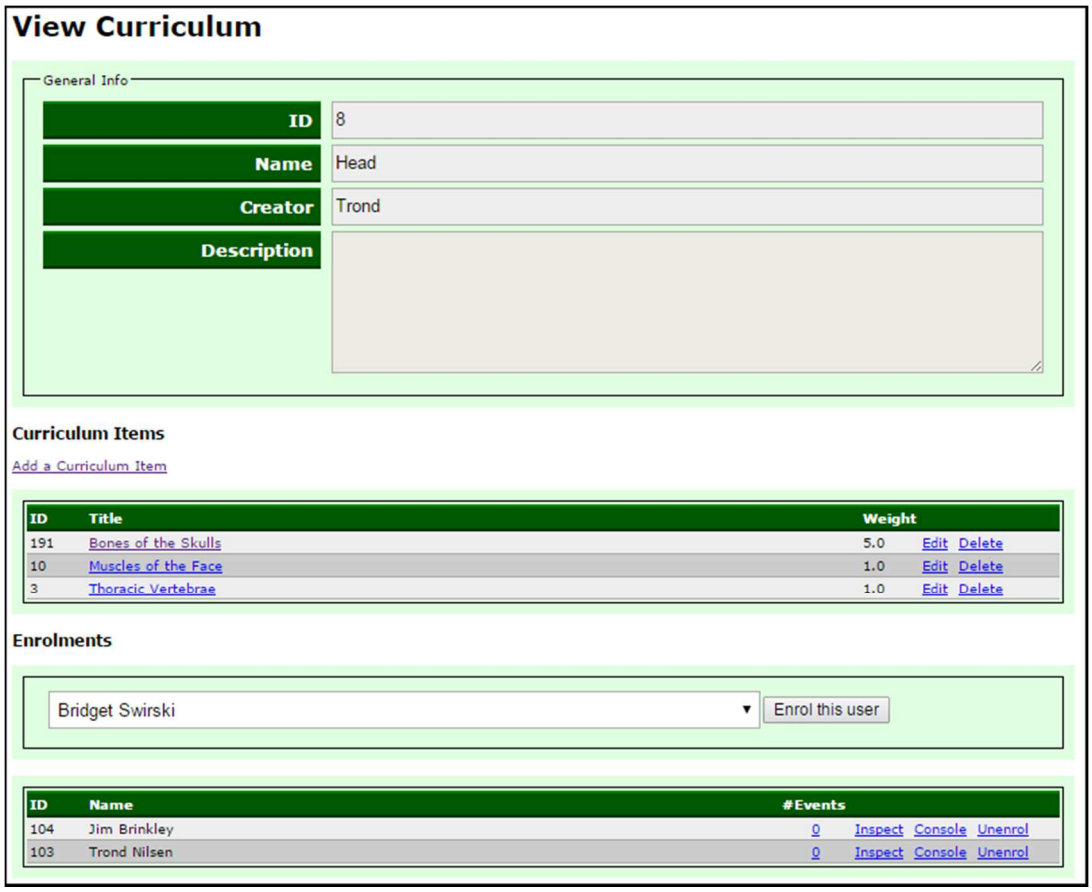


Figure 27 – Curriculum overview, including curriculum items and student enrolments

5.1.2 Implementation

The Curriculum Manager is implemented as a web application written in Java with Spring MVC, running on Tomcat, and using the Neo4J graph database for storage. It uses the Anatomy Engine's Query Manager to interact with the FMA, as well as its Asset Manager to aid the selection of entities. Programmatic interfaces are available primarily via direct invocation in Java. Conceptually, a curriculum is divided into several curriculum items, each of which is comprised of a number of entities and relations. There are two reasons for this:

- Firstly, it allows different parts of a curriculum to be weighted differently within the student model. For example, an educator teaching the names of the bones of the skeleton to high school students might place more emphasis on the bones of the skull and the limbs than they would on the bones of the spine and the rib cage,

as these are generally identified by number and distinction between them is not usually necessary at that level.

- Secondly, it affords an additional level of granularity for use in configuring activities around a curriculum. For example, a curriculum focused on the head and neck may include curriculum items for the bones of the skull, the muscles of the neck, and so forth. Activities may be configured to focus either on particular curriculum items or on the curriculum as a whole. Doing so in this way rather than simply splitting the curriculum into several smaller curricula is helpful both from an organizational standpoint and, more importantly, because it allows a single student model to cover several related activities. For example, using the aforementioned curriculum, an activity might be configured to teach the bones of the skull. Other parts of the curriculum would not appear within that activity due to this configuration, but student actions would be applied against the whole curriculum, allowing multiple types of activity teaching different aspects of anatomy to be integrated together into a single curriculum.

Each curriculum contains the following information:

- A name,
- A description,
- The name of its creator,
- A collection of curriculum items.

Curriculum items, in turn, are comprised of:

  - A name,
  - A description,
  - A weight,
  - A list of FMA entities, each represented by a unit called an EntityKnowledge, or EK,

  - A list of types of relation, each represented by a unit called a RelationKnowledge, or RK,
  - A flag indicating whether entity naming and recognition is part of that curriculum, called the label flag. When this flag is raised, the curriculum item also has a unit called a LabelKnowledge, or LK.

Finally, ontological data is represented as follows:

- Each EK includes the name and FMA ID of its associated entity along with a list of all relations involving that entity whose type is part of the same curriculum item; each of these relations is represented in a unit called a RelationFact, or RF.
- Each RK contains the IRI of its associated relation type along an RF for every relation of that type involving an entity within the same curriculum item.
- When they exist, each LK contains a series of elements called LabelFacts, or LFs.
- Each RF has references to the EKs and RK representing the subject, object, and type of the associated relation. Relations that have a subject or object that is not contained within the current curriculum item will be related to only one regular EK, with the missing entity represented by a stub called a hanging EK.
- An LF is a placeholder that conveys no additional information in the domain model. They exist so that student models built on the curriculum have somewhere to represent estimates of how likely it is that a student is able to recognize and name an entity. Though conceptually recognition and naming are different, one being the ability to recognize a thing's shape, the other the ability to name it, in this context they are largely indistinguishable, as students are rarely, if ever, likely to learn these separately as to do so would involve associating the shape or name with some other identifier, such as an ID number, that would be extraneous to normal use.

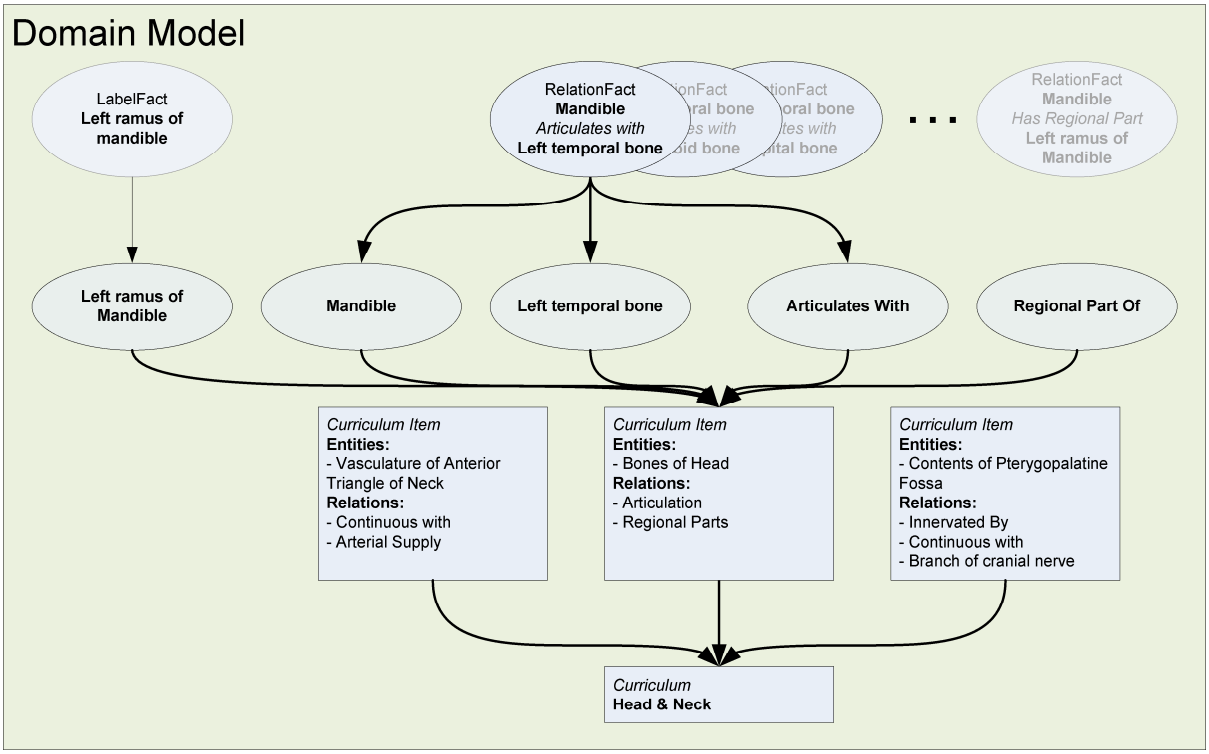


Figure 28 – Example student model, simplified

The representation described above is most easily thought of as a graph – a sample representation is shown in Figure 28. Unfortunately, it is neither convenient nor efficient to store graph-based data structures in a relational database such as the PostgreSQL database used by the other software components described. Consequently, domain models are stored using a graph database called Neo4J. Unlike relational databases, graph databases do not use a tabular representation of data, but instead use an efficient graph representation that directly links nodes and edges to one another on disk using pointers. Queries on a graph database often take the form of pattern matching expressions, usually with some starting point specified by an indexed identifier. For a detailed discussion of the various merits and drawbacks of graph databases, along with information on the specific implementation of Neo4J, see [213].

Formally, the curriculum representation described in this section is not an ontology in that relations are represented as nodes, not edges; relation types are represented as nodes themselves; and nodes exist that are not entities. It does, however, contain all of the information required to reconstruct an ontology containing the required entities and

relation types, but stores this information in a form more suitable to calculating estimates of student knowledge, as required by the student model discussed in section 5.2, below.

5.1.3 Further development

A major function of both the Curriculum Manager and the Scene Builder is the selection of entities. In the case of the Scene Builder, a user selects asset files, each of which represents an entity; in the Curriculum Manager, a user selects entities and relation types directly. Neither module could replace the other, as the Scene Builder doesn't understand relations, while the Curriculum Manager doesn't know about assets. Furthermore, a single tool that tried to do both would violate the software engineering goal of modularity by crossing the domains of tutoring and content creation. Nonetheless, there is no good reason why either component could not talk to the other using its web service API. Such an API already exists for the Scene Builder and could easily be implemented for the Curriculum Manager.

If implemented, this would allow curriculum items to be partly created by loading a list of entities from a scene, and vice versa. Given that many learning activities will require both content and a domain model and that these will generally concern the same entities, such a feature should frequently be useful.

5.2 Student Model

As discussed in section 2.6.1.2, a student model serves the rest of a tutoring system based on the trinity model by providing a repository and source for information about students that may be helpful in providing them with effective tutoring. The term "student model" can refer both to the overall student modeling component of a system and to the specific representation of a particular student, with the exact meaning usually clear from context.

Student models may contain a variety of information about students, including personal details, preferences, and emotional states. Most, however, focus primarily on representing information about a student's current state of knowledge. This aids tutoring in two main ways:

- Firstly, it helps the student and any other interested persons gauge that student's progress through the material they are expected to learn. Human educators may use this information to provide feedback or other guidance, while the student may derive motivation from a concrete sense of progress as they see the model's estimates of their knowledge improve.
- Secondly, it provides the tutoring system's other components with information that can be used to generate feedback, make recommendations, and otherwise guide the student's learning. A student model is necessary to any educational tool that seeks to be reactive to student behavior; if the knowledge level of different students cannot be distinguished, the tool is obliged to treat them all the same.

A variety of approaches exist for modeling student knowledge; a detailed review of these can be found elsewhere [75], [88]. The approach presented here is based on the idea of overlaying information about student knowledge onto a model of the domain in what is called an overlay model. A key limitation of this approach is that it affords no way of modeling any incorrect beliefs on the part of the student. That is, while an overlay model may assert that a student is not aware that that the humerus articulates with the scapula, it provides no way of modeling their mistaken belief that the foot is attached to the elbow, as this fact is not explicitly defined as false within the domain model. Despite this limitation, overlay models have been widely adopted.

The following factors must be taken into account when designing a student model:

- Student knowledge must be represented with a sufficient level of granularity and accuracy to provide useful guidance to students and tutoring tools. Unfortunately, however, there is a trade-off between model fidelity and its ease of construction. It is not possible to perfectly understand a student's state of mind, and a tutoring tool that is useful and cheap but slightly inaccurate is better than one that is nonexistent and perfect. The history of education is filled with tools and theories that are known to be flawed in some way but that continue to be used for this reason.

- For it to be useful, the model must be able to be updated quickly in response to new information. Ideally, it should be fast; at a minimum, it must be computationally tractable. Probabilistic graphical representations such as Bayes Networks provide very effective methods of reasoning about student knowledge, but the inference algorithms that operate on them are often NP-hard, depending on the network's specific design [214].
- Students ought to be able to inspect a model's estimates of their knowledge and, ideally, should be able to have the reasons for those estimates be explained in some way that is meaningful to them. An event log, for example, can list events that have been used to update the model so that the student can see the effect of each of their actions in a particular learning activity.

The model presented in this section is based on a lightweight probabilistic approach similar to Bayes Networks. Each fact in the domain model is assigned a probability that represents the system's estimate of the likelihood that a student would be able to correctly assert that fact. Group elements, such as EKs, RKs, curriculum items, and so forth each have probabilities calculated from the elements within them, with the final estimate for a curriculum representing an estimate of the likelihood that the student would be able to correctly assert any one fact in that curriculum if asked about one of them at random.

This approach provides a means of assessing student knowledge sufficiently well that games can be recommended and customized for particular students while also being easy to update and straight forward to understand. Its fidelity and level of sophistication could, in theory, be improved in a variety of ways, as discussed in section 5.2.3.

A key issue in the creation of Bayesian Networks and other probabilistic models is that there is often no objectively correct way to assign probability functions to each node. This reflects the fact that in the real world it is virtually impossible to determine in advance the exact probability of a particular event occurring, outside certain very limited circumstances. Even in a deterministic system such as a computer program, probabilities are never completely certain; an event that should always occur at a particular point in a

program may nonetheless not occur as a result of, say, a power failure. As a result, it is generally only possible to assess the accuracy and thereby the effectiveness of such a model by observing its behavior and comparing it against events in the real world. The model presented in this section has not been evaluated in this way as to do so is outside the scope and resources available to this project.

5.2.1 Usage

A student model is automatically created whenever a user is enrolled in a curriculum using the Curriculum Manager. A student model is associated with a single user, called the student, and a single curriculum. Students may be associated with multiple student models from different curricula, while curricula may be associated with multiple models from different students.

Users interact with student models using dynamic web pages embedded in the same overall web application as the Anatomy Engine. Using this interface, they may:

- List all current students.
- View a particular student, including personal information and a list of models associated with that student.
- View a student model, presented hierarchically as shown in Figure 29. Each line in this representation shows the probability estimate associated with a particular element in the model, with facts grouped inside EKs and RKs, and so forth. Each RF is listed once inside the RK and any EKs it is associated with. Since probabilities are not associated with hanging EKs, they are not given separate lines.
- View a list of all events associated with a particular student model. Each event describes an update applied to the student model as the result of a student asserting a fact. A sample list of events is shown in Figure 30.
- Manually trigger an update to a particular fact. This function exists solely for testing purposes.

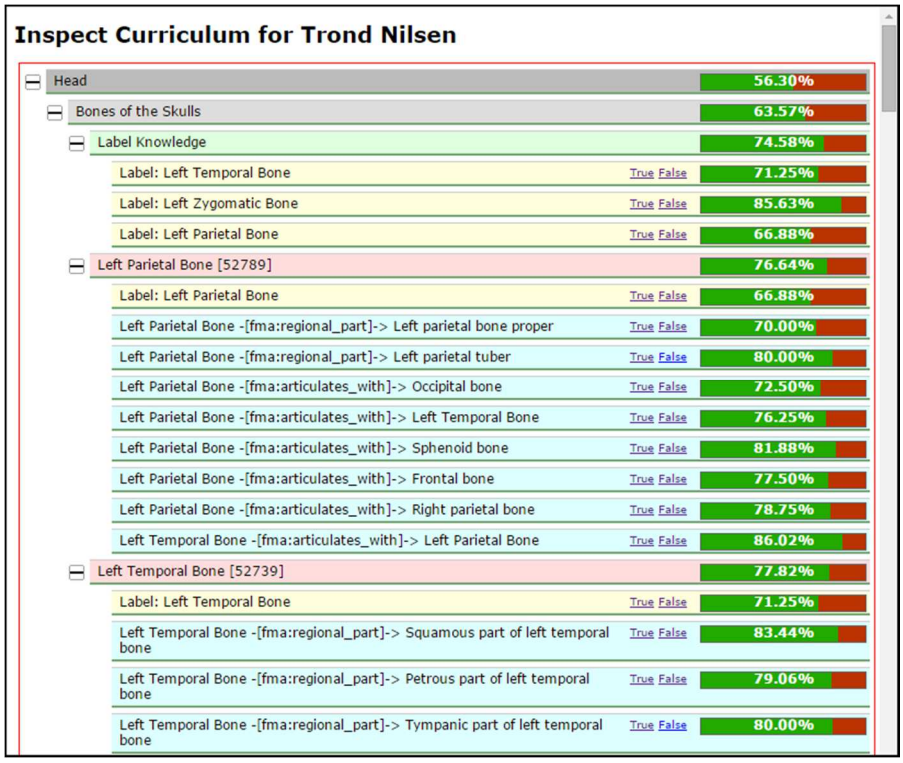
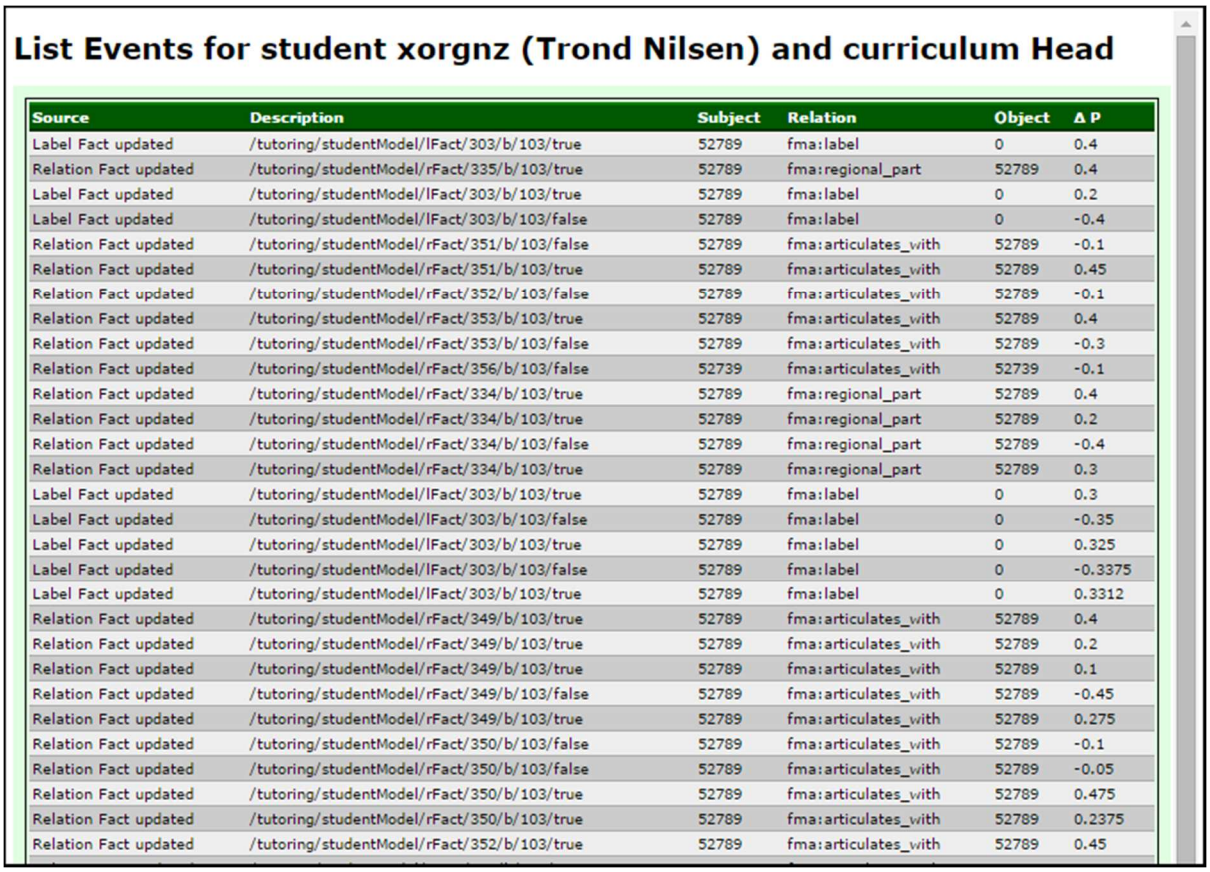


Figure 29 – User interface; Inspectable student model



List Events for student xorgnz (Trond Nilsen) and curriculum Head

Source	Description	Subject	Relation	Object	Δ P
Label Fact updated	/tutoring/studentModel/lFact/303/b/103/true	52789	fma:label	0	0.4
Relation Fact updated	/tutoring/studentModel/rFact/335/b/103/true	52789	fma:regional_part	52789	0.4
Label Fact updated	/tutoring/studentModel/lFact/303/b/103/true	52789	fma:label	0	0.2
Label Fact updated	/tutoring/studentModel/lFact/303/b/103/false	52789	fma:label	0	-0.4
Relation Fact updated	/tutoring/studentModel/rFact/351/b/103/false	52789	fma:articulates_with	52789	-0.1
Relation Fact updated	/tutoring/studentModel/rFact/351/b/103/true	52789	fma:articulates_with	52789	0.45
Relation Fact updated	/tutoring/studentModel/rFact/352/b/103/false	52789	fma:articulates_with	52789	-0.1
Relation Fact updated	/tutoring/studentModel/rFact/353/b/103/true	52789	fma:articulates_with	52789	0.4
Relation Fact updated	/tutoring/studentModel/rFact/353/b/103/false	52789	fma:articulates_with	52789	-0.3
Relation Fact updated	/tutoring/studentModel/rFact/356/b/103/false	52739	fma:articulates_with	52739	-0.1
Relation Fact updated	/tutoring/studentModel/rFact/334/b/103/true	52789	fma:regional_part	52789	0.4
Relation Fact updated	/tutoring/studentModel/rFact/334/b/103/true	52789	fma:regional_part	52789	0.2
Relation Fact updated	/tutoring/studentModel/rFact/334/b/103/false	52789	fma:regional_part	52789	-0.4
Relation Fact updated	/tutoring/studentModel/rFact/334/b/103/true	52789	fma:regional_part	52789	0.3
Label Fact updated	/tutoring/studentModel/lFact/303/b/103/true	52789	fma:label	0	0.3
Label Fact updated	/tutoring/studentModel/lFact/303/b/103/false	52789	fma:label	0	-0.35
Label Fact updated	/tutoring/studentModel/lFact/303/b/103/true	52789	fma:label	0	0.325
Label Fact updated	/tutoring/studentModel/lFact/303/b/103/false	52789	fma:label	0	-0.3375
Label Fact updated	/tutoring/studentModel/lFact/303/b/103/true	52789	fma:label	0	0.3312
Relation Fact updated	/tutoring/studentModel/rFact/349/b/103/true	52789	fma:articulates_with	52789	0.4
Relation Fact updated	/tutoring/studentModel/rFact/349/b/103/true	52789	fma:articulates_with	52789	0.2
Relation Fact updated	/tutoring/studentModel/rFact/349/b/103/true	52789	fma:articulates_with	52789	0.1
Relation Fact updated	/tutoring/studentModel/rFact/349/b/103/false	52789	fma:articulates_with	52789	-0.45
Relation Fact updated	/tutoring/studentModel/rFact/349/b/103/true	52789	fma:articulates_with	52789	0.275
Relation Fact updated	/tutoring/studentModel/rFact/350/b/103/false	52789	fma:articulates_with	52789	-0.1
Relation Fact updated	/tutoring/studentModel/rFact/350/b/103/false	52789	fma:articulates_with	52789	-0.05
Relation Fact updated	/tutoring/studentModel/rFact/350/b/103/true	52789	fma:articulates_with	52789	0.475
Relation Fact updated	/tutoring/studentModel/rFact/350/b/103/true	52789	fma:articulates_with	52789	0.2375
Relation Fact updated	/tutoring/studentModel/rFact/352/b/103/true	52789	fma:articulates_with	52789	0.45

Figure 30 – User interface; Student event log

Student models can be accessed either through the list of currently enrolled students shown when viewing a particular curriculum or by looking up a particular student on the list of all students and selecting the desired student model from a list of that student's enrolments.

5.2.2 Implementation

The student model is implemented as a web application written in Java with Spring MVC, running on Tomcat, and using PostgreSQL alongside the Neo4J graph database for storage. Users access it via a series of dynamic web pages while other software components invoke it directly via a Java API.

Each student model consists of a series of probability values related to elements within the domain model. Each fact, EK, RK, LK, curriculum item, and curriculum is associated with a single probability value, though hanging EKs are not. The values are stored on disk as records in a relational database that can be accessed in parallel with the graph database storing the domain model. All access to the student model database is carried out within joint transactions so that failures or errors in one database automatically trigger a roll back within the other, preventing the two from falling out of synch. This joint approach is used because the graph database does not well support arbitrarily large lists of values on nodes; that is, it is not efficient to maintain a table of student records on each node in the domain model, nor is it efficient to create a separate copy of the domain model graph for each student.

The probability values at each element of the domain model represent an estimate that a student would be able to correctly assert the fact or facts represented by that element when asked randomly about one of them. Estimates for whole curricula account for weightings associated with curriculum items; as such the probability value at that level represents the likelihood that a student would correctly asserts a given fact in that curriculum, given that the chance of each fact being asked is weighted according to the curriculum item containing.

A student model is created whenever a student is enrolled in a curriculum. Each model is associated with a single student and a single curriculum. During initialization, it is assumed that students probably don't know the material in that curriculum, and all probability values are set to $p = 0.2$.

Updates are applied to the student model by adjusting the probability value associated with a single given fact according to new evidence. Values for all subsequent elements in the domain model are then recalculated. The details of these calculations are discussed in the subsections that follow.

In conjunction with the domain model, each student model can be thought of as a graph representing a joint probability space. Edges within this graph are directed, with nodes representing facts proceeding to those representing knowledge elements, curriculum items, and finally a node representing the curriculum. Each node has an associated probability value; in the case of fact nodes, these values are assigned directly during model initialization and as a result of updates, while in the case of all other nodes, these values are calculated using a probability function based on the values of its ancestor nodes. This representation is closely related to Bayesian Networks, a type of probabilistic graphical model with well-defined properties and inference algorithms [214], but instead of conditional probability functions, it uses weighted averages to update non-source node probabilities, as described below. Conceptually, however, it is near equivalent.

As mentioned previously, algorithms for updating Bayesian Networks can sometimes be NP-hard. The model presented here avoids this problem by splitting each fact into three input nodes, one attached to each of its RK and two EK nodes, removing the possibility of cycles within the graph and allowing the model to be treated as a polytree rather than as a graph. This allows non-source node values to be calculated sequentially, beginning with knowledge nodes and proceeding through to the curriculum node, so long as no node is updated until all of its parent nodes have been updated. Furthermore, as fact probabilities only affect their descendants, updates can be applied by re-calculating only that narrow slice of the curriculum that corresponds to the fact node and its descendants.

5.2.2.1 Updating fact nodes

Whenever new evidence is obtained about a student's ability to correctly assert a fact, the student model is updated by altering the probability associated with the nodes representing that fact.

If evidence is positive, indicating an increase in our confidence in the student's ability, function (1) is used. Otherwise, function (2) is used, where p_t is the new probability value and p_{t-1} is the previous value:

$$\mathrm{p}_t = \frac{p_{t-1} + 1}{2} \quad (1)$$

$$p_t = \frac{p_{t-1}}{2} \quad (2)$$

This method corresponds to the common sense idea that if we were confident that an event would occur and it did not, our expectation of it occurring the next time would drop significantly, often to the point where we would say we are unsure, which colloquially represents a probability of around 50%. Similarly, if we were unsure an event would occur and it did, our expectation of it occurring again would significantly increase, but we would not be persuaded it would always occur until it did so several times in a row.

The method presented above mimics this behavior, with confidence in a student's ability to perform correctly reaching a high level ($p > 0.9$) after at most three demonstrations, even if confidence was previously extremely low.

This update scheme is entirely arbitrary, and has not been validated empirically. It is however quick and easy to understand, and exhibits approximately the same behavior that would be expected of a human tutor operating on the same evidence.

This method has two major drawbacks:

- Firstly, it does not allow for variation in the quality and strength of different pieces of evidence. A vague or mostly correct assertion must either be treated as if it was clear and completely correct or be completely ignored. This is problematic,

particularly if learning activities do not directly question students to gather evidence but instead rely on inferring assertions from their actions.

- Secondly, this method does not account for time. Three correct assertions in rapid succession should not afford the same confidence as three correct assertions over the course of a week, but the two patterns cannot currently be distinguished. To prevent students and activities spamming the model with updates, each fact is marked with a timestamp that is consulted before its probability is updated. Successive updates that increase confidence may not be applied unless they are separated by at least thirty minutes, while those that decrease confidence may be applied immediately.

5.2.2.2 Updating knowledge nodes

A knowledge node's probability value is an estimate of the likelihood that a student would give a correct answer when asked at random about one of the facts associated with that node.

Since all of the facts associated with a knowledge node have an equal chance of being asked, this value can be calculated as the mean of the probability values of each of those facts.

5.2.2.3 Updating curriculum item nodes

Similarly to knowledge nodes, the probability value of a curriculum item is an estimate of the likelihood that a student would give a correct answer when asked at random about one of the facts associated with the knowledge nodes that make up that item.

Since each RelationFact is represented by either two or three input nodes, it is not possible to simply average the probabilities of the knowledge nodes, as the contribution of each fact would not be equal. Similarly, the facts associated an RK with only three facts would be over-represented compared to those associated with one containing twenty facts. Instead of averaging knowledge nodes, then, a curriculum item node is updated by averaging the probability of all of the facts associated with either a LabelKnowledge or a

RelationKnowledge node within that curriculum item. Note that even though EntityKnowledge nodes are ignored, all facts associated with those nodes will nonetheless be accounted for, as all must be associated with a RelationKnowledge node.

5.2.2.4 Updating curriculum nodes

The probability associated with a curriculum node is an estimate of the likelihood that a student would give a correct answer when asked at random about any of the facts in that curriculum, with the chance of each fact being asked weighted according to the curriculum item it belongs to.

Curriculum items vary both in terms of how many facts they contain and what weight each is assigned. Consequently, a curriculum's associated probability cannot be calculated simply by averaging the probabilities associated with each of its items. Instead, each item's probability is scaled by multiplying it by its weight and fact count divided by the total weight and fact count of all items, resulting in a weighted average that accounts for both factors. This calculation is shown in function (3), where n is the number of items in that curriculum; $p_i, w_i,$ and c_i are the probability, weight, and fact count, respectively, of the ith curriculum item; and p_c the probability value of the curriculum itself.

$$\mathrm{p_c} = \sum_{i=1}^{n} p_i \frac{w_i c_i}{\sum_{i=1}^{n} w_i c_i} \qquad (3)$$

5.2.3 Further development

As mentioned previously, the formulation above is intended as a simple example of how the FMA can be used as the basis for a student model. As such, it has several limitations and weaknesses that could be addressed through further development.

5.2.3.1 The assumption of independence

As defined, the model assumes that all facts are independent. That is, it assumes that a student's knowledge of one fact affords no evidence for their knowledge of another. This may be true for a pair of unrelated facts concerning, say, brain structure and foot

musculature, but is surely false for facts that are closer to one another, say a pair of facts describing the tributaries to a major vein.

The main effect of this false assumption is that students must demonstrate mastery of all facts contained within a curriculum individually. That is, it is not possible to distinguish between students who have mastered only half of the facts from those whose mastery is half as strong.

The decision to embrace this obviously false decision was considered valid for two reasons:

- Firstly, by failing to model these dependencies, we are at worst underestimating a student's knowledge, which is far better than overestimating it – the former results in reduced efficiency through extra teaching, while the latter risks letting students proceed with holes in their education. Over-teaching can largely be addressed by requiring only that students pass a threshold such that we are reasonably certain that they have mastered most of the facts in the curriculum, similarly to how we are willing, in analogue education, to pass a student who receives 80% on their final exam.
- Secondly, correctly capturing all of the dependencies in the model is extremely difficult, if not impossible. Furthermore, it may not be necessary for satisfactory performance as demonstrated by naïve Bayesian classifiers, a classic approach to probabilistic classification that assumes independence between all inputs that have been satisfactorily applied in a wide variety of applications [214], [215].

An obvious improvement might be to introduce a heuristic that applies a limited update to all neighboring facts whenever a particular fact is updated. Any such heuristic is likely to introduce problems of its own, but may improve the performance of the system overall. No attempt has been made to test such heuristics, as this is outside the scope of this project. Furthermore, testing the relatively subtle effects of such a change would require data from a large number of students that is not available. Such model refinement requires an ongoing research effort with much greater contact with students.

Inverse relation pairs are a special situation in which the case for a heuristic is strong enough that its implementation is warranted without testing. If evidence is received indicating an increase in confidence in the student's knowledge that ***Mandible*** **Has regional part** ***Body of mandible***, it should similarly increase our confidence in their knowledge that ***Body of mandible*** **Is regional part of** ***Mandible***.

5.2.3.2 Model initialization

During initialization, all probability values are set to $p = 0.2$. This is based on the assumption that students have not yet been exposed to the subject matter contained within a curriculum and are thus unlikely to be able to give correct answers. This assumption, however, may not be accurate, as students may come to a class with a wide variety of levels of prior knowledge.

An alternative approach might be to allow educators to specify how models will be initialized during curriculum construction by selecting between several starting heuristics. For example, a heuristic for new material might assign low starting probabilities, while one for a refresher course might assign higher values based on the assumption that students should have some chance of knowing the material already.

In situations where students are enrolled in multiple curricula that overlap, facts from one student model might be used to initialize another. For example, if a student has previously studied the bones of the face in a curriculum about the skull, it would be reasonable to use facts from that student model to initialize those in a new curriculum about the face.

5.2.3.3 Model independence

As currently implemented, student models are completely independent from one another, meaning that updates triggered by a game associated with one curriculum would not affect student models associated another. Using the example given in the previous subsection, updates from games associated with the face curriculum would not affect facts in the skull curriculum, and so the models would diverge.

Whether this is a problem or not is to some extent determined by how educators make use of multiple curricula, which will not be understood until the system is adopted. To address this problem, student models associated with the same student could be merged so that each model uses the same fact nodes, allowing any game played by that student to, in principle, update all models.

5.2.3.4 Improved fact update functions

The fact update scheme described in section 5.2.2.1 is an approximation of a common sense idea of how people form opinions by judging past performance. It is by no means a sophisticated approach to updating, but is sufficient for the purposes of demonstrating the overall approach described here.

Much research has gone into more advanced update schemes. The method described here could easily be replaced with a more sophisticated approach known as knowledge tracing, originally devised in the 1990s [216] and still popular with current researchers [217][218]. This approach applies Bayes' rule to estimate student knowledge. Formula (4) shows a classic formulation, relabeled in (4a), where p_t is a fact's new value, p_{t-1} its previous value, s a general estimate of how likely a student is to give an incorrect answer despite knowing the correct one, and g the chance of them simply guessing. An analogous formula can be composed for situations when a student gives an incorrect answer.

$$p(KnowFact \mid Correct) = \frac{p(Correct \mid KnowFact)p(KnowFact)}{p(Correct)} \tag{4}$$

$$= \frac{p(Correct|KnowFact)p(KnowFact)}{p(Correct|KnowFact)p(KnowFact) + p(Correct|\neg KnowFact)p(\neg KnowFact)}$$

$$p_t = \frac{(1-s)p_{t-1}}{(1-s)p_{t-1} + g(1-p_{t-1})} \tag{4a}$$

Research on the specific formulation of inference rules for intelligent tutors is a domain unto itself that is outside the scope I have defined for this project, largely because the analysis and testing of more advanced update rules requires substantial data gathering efforts with test users for which time and resources are not available.

5.2.3.5 Incorporating time

Another problem with the fact update scheme discussed above is that it does not account for the passage of time. In general, it would not be reasonable to allow a student to give the same answer three times in rapid succession and have that count as three separate pieces of evidence for their knowledge. Furthermore, one would expect that estimates of a student's knowledge should decline over time, partly because the older our evidence the less likely any estimate based on it is accurate, and partly because students can be expected to forget or at least become less confident with facts as time passes.

The first part of this problem could be addressed by filtering incoming evidence and ignoring rapid updates to the same fact. Additional pieces of positive evidence received without a certain amount of time passing should be ignored, but multiple pieces of negative evidence in short succession likely indicate that a student is guessing and should thus serve to strengthen our belief that they cannot give the correct answer.

5.2.3.6 Inspectable student model usability

At present, users are able to review student models using a hierarchical interface similar to how directory structures are visualized in many standard user interfaces. This accurately conveys the information contained within that model, but does not make its structure particularly easy to understand. A graph-based interface could be easier, but the number of facts could easily make this unwieldy. Alternatively, a focus plus context visualization such as a SpaceTree [219] could be employed, in which broad tree structures are made manageable by interactively showing a small part of the tree in detail, with the rest shown using structural indicators to provide context. User studies would be required to determine which interface is best.

The events from which a model's current state is derived are made available to students in a user interface that is effective but not particularly usable, as all events associated with a particular student model are provided in a single list that would quickly become large and unwieldy. The current implementation is adequate for testing and demonstration, but could be dramatically improved by showing filtered lists of events via the main student model visualization. Students exploring their own progress in the student model could then quickly drill down into particular nodes to find out how those values were calculated from a series of events.

5.3 Assessor

An important part of the job of a human tutor role is to assess student actions to determine if they are correct or incorrect in any given situation. In doing so, the tutor is essentially consulting a domain model in their head, comparing student actions to it and describing or hinting at the source of discrepancies.

The Assessor helps games and learning activities built on the tutoring framework to emulate this ability by providing mechanisms for validating student assertions about anatomy. For example, in a game about building a skeleton up from a pile of bones, an attempt to place the left radius and the right humerus next to each other can be interpreted as an assertion that ***Left radius*** **Articulates with** ***Right humerus***. This assertion would then be validated against the FMA and found to be false and that information would then be used by the game to provide appropriate feedback to the student. Furthermore, the correctness of this assertion could be used as evidence about that student's knowledge and incorporated into the student model.

Conceived as above, assessment involves two steps. First, student actions must be interpreted into subject-predicate-object triples, and then those triples must be validated against the FMA.

It is virtually impossible to implement the first of these steps in a generalized way that will work on any arbitrary learning activity. To use examples from the game concepts described

in chapter 6, a fully generalized assessment service would have to be able to interpret both a student's choice to shoot a zombie in the leg and their allocation of a liver to a particular cooking station (see sections 6.2.2 and 6.2.3, respectively), among many other types of action. The interpretation of student actions as assertions about anatomy must therefore occur within the learning activity rather than in some service provided by the framework.

The validation step, however, is straight forward to generalize, and is the primary function of the Assessor component, as described below.

5.3.1 Usage

The Assessor presents no user interface, as it is intended solely as a service for use by other software components.

5.3.2 Implementation

The Assessor is implemented in two parts: the first a web application written in Java with Spring MVC, running on Tomcat; and the second, a set of JavaScript classes for use in developing learning activities. A web service interface is available for applications that cannot take advantage of these classes.

The current implementation of the Assessor is very simple. Assertions are received via a web service and validated using a query within the Anatomy Engine's Query Manager. Assertions are represented as subject-predicate-object triples, with entities specified by their FMAID and relations their IRI. In addition, the Assessor allows client applications to trigger updates on a student model.

Since the quantity of anatomical knowledge covered by most activities is likely to be relatively small, the approach of invoking a web service to assess each assertion against the entire FMA is also likely to be inefficient. Activity designers can ameliorate this by configuring the Assessor's JavaScript classes with a cache of pre-loaded facts that are matched against before the web service is invoked. Assertions that are found in this cache are considered correct, while those that are not are sent to the server for validation against

the full FMA. Further improvements may be made in cases where all of the facts that can possibly be asserted within an activity are known in advance. In this case, web service invocations can be disabled entirely such that any facts that do not match those in the cache are automatically assumed to be invalid.

5.3.3 Further Development

The Assessor satisfactorily meets its requirements as currently defined. No further development is suggested.

5.4 Conclusion

This chapter presented a prototype tutoring framework, based on the FMA, that demonstrate three possible answers to research question B, re-stated below:

How can an ontology be used to support tutoring?

Three components were presented, addressing between them the three parts of the trinity model for intelligent tutoring. The Curriculum Manager allows educators to create domain models from the FMA, while the student model models student knowledge using an approach based on the idea of a Bayesian Network. Finally, the Assessor supports tutoring by providing feedback to students in learning activities using automated assessment against on the FMA.

Demonstration of the framework is left to chapter 7, where a fourth tutoring component is presented that closes the educational feedback loop of student activity, assessment, and further activity by offering recommendations to students based on available activities and their current progress as represented in the student model. All four components, along with elements of the Anatomy Engine, are integrated an example learning activity template in the form of a self-generating quiz.

5.4.1 Future work

As prototypes, the components presented in this chapter offer performance that is adequate to demonstrate the potential of using an ontology to support tutoring but that is far from optimal. Many avenues for future research and development remain, including:

- More sophisticated methods of updating fact nodes in the student model, including algorithms that take advantage of time, that apply more sophisticated Bayesian reasoning, and that allow updates of variable strength, as discussed in sections 5.2.3.4 and 5.2.3.5.
- Modifying the student model to account for dependencies between different types of fact, as discussed in section 5.2.3.1.
- Improvements to the way student models are initialized, as discussed in section 5.2.3.2.
- More usable visualizations of the domain and student models, as described in section 5.2.3.6.

Chapter 6. Designing games to teach human anatomy

Chapters 4 and 5 are primarily technical in nature in that they present methods and prototypes that allow an ontology to support functionality necessary for the creation of learning activities and games. Chapter 4 presents the Anatomy Engine, my ontology-driven framework for creating 3D anatomical content, while chapter 5 presents my ontology-driven tutoring components. In this chapter, I will address the problem of design by answering research question C, re-stated below.

> **How can games and learning activities be designed to take advantage of an ontology-driven educational platform?**

There are motivations behind this question:

- Though the answers to research questions A and B provided in chapters 4 and 5 demonstrate how an ontology can be used to address technical challenges, the case for the overall vision of ontology-driven education can be made stronger by demonstrating how a variety of games and activities can be designed that take advantage of the framework proposed.
- The range of learning activities currently available online, as documented in the survey presented in section 2.8, is pitiful. Though several excellent information repositories were found, with good knowledge organization and presentation, there are virtually no compelling interactive resources available. The highest quality activities were quizzes and drill games, and while these may help students to memorize material in the short term, they do little to engage them and encourage

the active processing of information that leads to true knowledge formation. Since a central underlying claim of this work is that games and learning activities can be compelling and effective methods of education, it is necessary to demonstrate that better designs are possible.

This chapter presents a variety of game and activity concepts that take advantage of the ontology-driven tools presented in chapters 4 and 5. These designs have emerged from a combination of design discussions with various colleagues and the application of a structured design process that I have undertaken to ensure that I have explored a suitably broad area of the possibility space for design. This process, similar to a general approach to problem-solving called General Morphological Analysis [220], relies on an up-front characterization of a problem space followed by a rigorous exploration of that space, and aims to counteract the all-too-human tendency to slip into easy, well-understood design tropes rather than striving for new ideas.

The remainder of this chapter is structured as follows:

- Section 6.1 describes the methods I have employed to create game concepts. The method of General Morphological Analysis is presented along with discussions of its strengths and weaknesses and an account of my experience working with it.
- Section 6.2 presents twelve game concepts derived from my application of these methods along with discussion of each concept's gameplay, educational attributes, and integration with my ontology-driven educational framework.
- Section 6.3 offers a reflection on the material presented in this chapter and assesses it as an answer to research question C.

6.1 Method

In my previous work, I encountered many research papers that purported to "explore" the design of games utilizing some new technology such as augmented reality, location-tracking, or gesture recognition. Unfortunately, these papers rarely lived up to this challenge and instead limited themselves to presenting a single design concept and

demonstration. I am thus concerned, perhaps overmuch, with validating any claim I make that my work explores the design space of games for teaching anatomy. As a result, I included in my research proposal a plan to apply a structured design process to conceive of games to demonstrate my approach. This process was composed of an exploratory stage interleaved with several refinement stages. The goal was not simply to explore a variety of ideas, but also to select ideas for further design and eventually, implementation.

Since presenting this proposal, my thinking has evolved. Firstly, it became clear that in this context, refinement was of less value, as my goal was not to present a small number of polished games but to demonstrate the overall potential of ontology-driven learning activities and games for anatomy. Secondly, it has become clear that a structured design approach is valuable less because it claims exhaustiveness of coverage, but more because it defines a reproducible strategy that makes clear to future designers how concepts were obtained, how they might be organized, and where design effort has not yet been expended – in short, by making it clear where I have looked, it makes it clear where else they might look. Finally, I have come to realize that this structured process does not exist in a vacuum, as design ideas have emerged throughout my work, irrespective of the process.

Though I initially thought that the process I had proposed was original, I later discovered that it was not – it was in fact near identical to a problem solving strategy proposed decades earlier.

6.1.1 General Morphological Analysis

In the late 1930s, Fritz Zwicky, who was to become a noted astrophysicist and aerospace engineer, developed an approach to solving complex socio-technical problems that he called General Morphological Analysis (GMA). He championed this approach through much of the Cold War, applying it to numerous projects in a variety of fields [220]. Following his death in 1974, the method fell briefly into obscurity, from which it has lately re-emerged in computerized form in work by researchers at the Swedish Defense Research Agency [221][222].

The method is described, by Zwicky, as "*a general method for structuring and investigating the total set of relationships contained in multi-dimensional, usually non-quantifiable, problem complexes*", and has been applied in astrophysics, engine design, fuel economy, space travel, public policy, and defense planning [223][224][225]. Put simply, it involves carefully defining a problem, identifying the parameters that describe possible solutions to the problem, then considering each combination of parameters to envision possible solutions. I find it useful to think of it as a problem-solving analogue to the physical search patterns used in search and rescue.

General Morphological Analysis is applied using the following steps, with a simple example worked through in italics below each step:

1. The problem to be solved is carefully defined. If possible, decision criteria for comparing possible solutions are laid out. Where objective or quantitative criteria are not feasible, some other decision process should be defined in consultation with stakeholders. Both starting and goal states should be made as definite as possible.

   Bob's elderly and infirm grandmother, June, wants him to help her frame a picture of his daughter's graduation for her to post on the wall. June is the only stakeholder, so her subjective preference is defined as the sole decision criterion.

2. The problem is analyzed to identify ways that possible solutions might vary. A supply chain problem, for example, might vary in terms of transportation method, warehousing strategy, and so forth. Solutions may vary in many ways; as many as is practical should be identified.

   - Each varying attribute is reduced into a set of discrete values, called levels, each describing a state that that attribute might take in some subset of solutions; this requires that attributes that vary continuously be discretized. When defined this way, attributes are referred to as solution parameters.

Ideally, solution parameters should be sufficiently well-defined that each possible solution can be described by only one level within each parameter.

Solutions to Bob's picture framing problem vary by the shape of picture frame used. He thus defines a solution parameter called "shape", with levels "round", "square", "portrait", and "landscape". He also defines parameters for mat width ("none", "narrow", "wide"), mat color ("blue", "black", "red", "green", "white"), frame style ("plain", "linear", "ornate"), print size ("small", "medium", "large"), and which room the picture will be hung in ("dining room", "living room", "office", "bedroom").

3. The parameters identified in the previous step are used to define a multi-dimensional array, with one dimension per parameter. This array is thus a discrete and approximate representation of the possibility space of all solutions to this problem, with each combination of parameter levels, called a cell, describing a particular subset of possible solutions. Unfortunately, the size of this array explodes combinatorically as additional parameters are added, which can result in there being far too many combinations to consider individually.

   Bob's six parameters result in a 4x3x5x3x3x4 morphological box with 1440 combinations, more than he is willing to consider individually.

   To combat this, the box may be simplified by iterative application of the following three steps:

   - A parameter may be removed by restricting solutions to a particular level within that parameter. This method imposes a limitation on the scope of the search by eliminating all solutions associated with other levels of the discarded parameter.

     Bob discovers that because the photo being framed is wide and short it will only fit well inside a landscape frame. He thus discards the parameter of frame shape and requires that all solutions use the level "landscape".

- A parameter may be removed without restriction, allowing solutions to vary amongst that parameter's levels. This preserves the scope of the search, but reduces its rigor as the variation described by that parameter remains.

  Bob decides that the room in which the picture will be displayed is really a separate decision to how the picture will be framed, particularly as June may decide to move the picture later. Consequently, he discards the parameter of room from consideration, leaving it to vary freely.

- A parameter level may be removed. Solutions may continue to vary according to that parameter, but the scope of the search is reduced somewhat.

  Looking at June's existing framed pictures, Bob notices that she doesn't have any photographs with wide mats. He checks with her, and she says she thinks they're a waste of space. Bob decides to remove the level of "wide" from the "mat width" parameter.

Changes in box definition must be documented clearly so that the results and limitations of the analysis can be properly understood. In Zwicky's view, box simplification is often undesirable as it can reduce both the overall scope and rigor of the search, harming any claims the analysis might have of exhaustiveness.

Bob has simplified his problem to a 2x5x3x3 morphological box with four dimensions and 90 combinations, which is far more manageable.

4. Cross-consistency analysis may be applied to further simplify the box. Each pair of levels between each pair of parameters is considered, and select pairwise combinations are removed from the box. Removal may be based on definite inconsistencies such that it is impossible for two levels to co-exist, or they may be based on judgment. As before, all decisions to exclude certain combinations must be documented clearly. Though a box with many parameters may contain many pairwise combinations, there are far fewer of these than there are total

combinations; pairwise combinations grow quadratically with the number of parameters whereas total combinations grow exponentially.

Bob realizes that there is no point selecting a mat color if no mat is used. He thus simplifies his box by removing color from consideration in combination where "mat width" is "none". This leaves him with 54 combinations.

5. Solution concepts are generated for each combination of parameter levels, or cell, within the reduced morphological box. Depending on how exhaustively the box has been constructed, multiple possibilities may exist in each cell and creativity may be required to select one that is suitable. If the goal of the analysis is optimality, this will be the best imaginable solution according to the previously defined decision criteria; if the goal is exploratory, the solution that best demonstrates the essential character of that cell is desired.

   Bob finds an online picture framing service that lets him upload an image and configure framing parameters to choose from. He uploads his photo and produces images of each possible framing solution. Several ornate frames are available, so Bob mixes things up a bit and varies which exact ornate frame is used within sample solutions that have the level "ornate" for the parameter "frame style".

6. Finally, decision criteria are applied to select between the various solutions defined. Depending on context, multiple solutions may be selected for further examination, such as feasibility analysis or financial and political consideration.

   Bob shows the sample framing pictures to June, and asks her to tell them which she likes the most. She decides that she likes two solutions, one for the wall in the living room, and a smaller one for her photo wall in the bedroom.

6.1.1.1 Advantages of GMA

GMA has several advantages over other design methodologies:

- GMA attempts and largely succeeds at providing a realistic means of exhaustively considering all possible classes of solution to a problem. Though it is of course

unwise to claim that it completely exhausts the solution space, it provides a rigorous procedure that comes as close to exhaustive as is likely practical. Three significant limitations are worth noting:

  - Firstly, the method relies on the identification of all relevant parameters and parameter levels. Though it forces the consideration of improbable classes of solution within the parameters selected, completely novel ideas may remain outside the box.
  - Secondly, full consideration of all cells within the box can easily become impractical, and judgment is required to trim down the space, as suggested in steps 3 and 4 above. This may result in good solutions inadvertently being discarded before they are discovered.
  - Finally, there may be substantial variation left within each cell that GMA makes no attempt to manage. This variation may be captured by adding new parameters, but this may not be practical or possible.

- The results of GMA are not only an array of possible solutions, but also rigorous documentation of how those solutions were reached. This documentation supports future work in two key ways:
  - Firstly, it enables future researchers to see where effort has been expended and plan their work accordingly. It may also help to trigger new ideas as parameters and levels can be examined closely for extension or replacement.
  - Secondly, it enables transparency and inspection of the process by third parties, allowing them to assess, for example, whether due diligence has been applied, relevant policy considered, or educational standards met.
- GMA inhibits design fixation, the tendency of designers to retreat to safe, well-understood ideas [226], by forcing them to think carefully about each combination. Though good solutions outside the box may still require leaps of creativity, those

within the box stand a much higher chance of discovery under GMA than they would using a less structured method.

- Though problem-solving and design are not in any way objective processes that proceed the same way every time regardless of participants, a well-documented analysis performed using GMA is at least somewhat reproducible, and it seems reasonable to suppose that groups working with it independently will reach relatively similar conclusions

6.1.1.2 Challenges of GMA

There are several challenges in employing GMA:

- Problem definition may be difficult or impossible, preventing the selection of parameters. This is particularly true for so-called "wicked problems" [227], whose definition cannot be agreed on by stake-holders, change as they are explored, and have tight causal relationships with otherwise unrelated problems that are in turn often wicked; economic and social policy problems commonly fit into this category. Recent work has examined the application of GMA to problems of this sort [221].
- For similar reasons, it may be impossible to establish decision criteria for whether a problem has been solved, preventing clear comparison of possible solutions and potentially interfering with box simplification.
- Parameter selection, too, can be fraught with problems. If the goal is to exhaustively explore a space using GMA, careful attention must be paid to ensure that all pertinent parameters are captured. Possible methods include expert review, extraction from taxonomies and other systematic surveys, and inspiration from past solutions to similar problems. As the claim of exhaustiveness is likely to be subject to significant rhetorical pressure, great care should be taken to document the parameter selection process.
- As emphasized in step 3, combinatoric issues plague solution spaces with many parameters – even a relatively simple box with 5 parameters of 3 levels each

contains 243 cells, and problems with initially many hundreds of thousands of parameters have been examined in the literature [221]. Though steps 3 and 4 provide mechanisms to trim the box, these decisions have significant impact on the results of the analysis and thus require careful judgment and documentation.

- Ideally, parameters are defined in such a way that each level is clearly defined and solutions fit cleanly into a particular combination. Unfortunately, this may be impossible in situations where solutions are themselves complex. Games, in particular, are difficult to categorize – straight forward genre classification, for example, is notoriously uncertain, and all but the simplest examples fit into multiple levels. Nonetheless, no better alternative classification for style of play has yet been defined.
- When cells are considered consecutively, there is a temptation to simply extend from solutions considered in an adjacent or related cell rather than search for new ideas. Such a strategy results in ideas that technically fit the requirements of the space but do little to capture the design variation possible within it. For example, in designing learning activities, the basic concept of a quiz game might be re-purposed to satisfy several cells in parameters such as "knowledge type", "anatomical region", or "body system". This is particularly a problem when few parameters are selected, and much variation remains possible.

In assessing an application of GMA, it is important to understand that the assertion it makes is that an idea has been generated in each cell within the box, representing a unique combination of the parameters defined. The method is unable to make any claims about the exhaustiveness of parameter selection (novel ideas may yet be discovered by thinking "outside the box"); the representation of possible variation within each cell; and the suitability or optimality of solutions to actually solve the underlying problem being faced (as opposed to the problem articulated in the decision criteria). Clear documentation makes it possible to consider these limitations, and is essential to successful application of the method.

6.1.2 My approach

When I first tried to apply my method, I thought it was novel - I was not yet aware of General Morphological Analysis. Though it initially seemed sound, several methodological problems eventually became clear that paralyzed my progress. Later, when I learned of GMA and began to investigate other applications of that method in the literature, I was able to break out of that paralysis and complete the process. This section provides an account of my experience working through these problems and designing the game concepts presented in section 6.2.

6.1.2.1 Structured design

The application of my original process was plagued with false starts. On reflection, it is clear that this was because my desire for an exhaustive search entailed concerns that were nigh-impossible to address; for example, selecting parameters so that the entire design space was covered, justifying that selection, and convincingly searching the box defined by it. Eventually, I re-examined the intent behind the inclusion of design issues in research question C and came to the realization that my goals would be better served by a search that aimed to find representative examples of different parts of the space that illustrated the application of my framework.

Another breakthrough moment came when I learned about GMA and began to read accounts of other groups working with the method. These helped me realize that several of my difficulties were due to an insufficiently well-defined design problem. By adapting my research goals, as discussed in section 3.1 and the introduction to this chapter, I defined my problem as:

> **The identification of games and learning activities for teaching adult human gross anatomy that illustrate ontology-driven methods of creating educational software.**

By unpacking this definition, I defined a set of three decision criteria for selecting ideas within each cell. Good game concepts would:

1. Illustrate some aspect of the framework and its application.

2. Be compelling and effective both as games and as educational activities.

3. Taken together, be diverse, with minimal overlap and maximal coverage.

As these criteria are both subjective and inter-related, my evaluation was substantially based on personal taste, game design experience and expertise, and general reasoning. In theory, educational effectiveness is objectively measurable, but it is notoriously challenging and complex to do this well. Consequently, I have limited my consideration of effectiveness to the comparison of concepts with existing educational games, the consideration of design recommendations available in the literature, and the extrapolation of how each ought to behave from current knowledge about learning.

The next step in applying GMA is to identify possible parameters that can be later trimmed and simplified to create a manageable morphological box. Once again, I went through several early false starts, particularly as a result of worrying too much about the number of cells I could reasonably explore, thus limiting my willingness to consider different parameters. Another early distraction was the possibility of managing a larger box by sampling cells according to some scheme to obtain good coverage over the space inside the box while also allowing relatively expansive parameter selection to minimize the space outside.

Eventually, I identified six parameters that seemed reasonable for inclusion in my analysis:

- **Anatomical region** – texts, atlases, and courses frequently organize anatomy around the regions of the body; divisions might include thoracic, abdominal, head, neck, and limb anatomy.
- **Anatomical system** – anatomy can be structured according the physiological system it is part of; divisions include the vascular, skeletal, muscular, nervous, digestive, and endocrine systems.

- **Anatomical knowledge type** – as discussed in section 2.4.1, anatomical knowledge can be split up into types such as partition, recognition and naming, connectivity, and so forth.
- **Educational involvement** – learning activities can be described by the overtness of the educational material they contain; as described in section 0, games may be categorized as containing foreground, background, or contextual educational material.
- **Educational strategy** – a variety of distinct educational strategies are described in the literature and employed in games. Examples include problem-based learning, where students must solve complex problems using resources in the game, mastery learning where in-game progress is tied to mastery of educational material, and rote learning, where simple repetition of facts is used to enforce memory. Many other strategies exist, and are often overlapping and sometimes flexibly defined.
- **Gameplay genre** – critics, designers, and players commonly use gameplay genres to describe the type of actions and decisions that must be made during play. Common genres are the shooter, the real-time strategy, the story game, the resource management game, and the god game. Genres may overlap and are usually not well-defined, but no better classification is well accepted.

Given limitations on resources and time, it was impractical to include all of the above parameters. To produce a manageable box and avoid the need for sampling methods that would dilute the strength of the method, I determined that I would need to limit my box to some combination of parameters that resulted in between 8 and 12 cells.

Examination of the parameters identified yielded the following observations that were helpful in trimming and final parameter selection:

- Firstly, variation in anatomical region and system is largely overshadowed by variation in type of anatomical knowledge; for example, much of the distinctness

of neuro-anatomy is in the fact that it consists of connective and partitive knowledge, and inclusion of the knowledge type parameter could ensure that both are considered. Remaining variation is largely a matter of content, not activity structure or gameplay.

- Secondly, educational strategies can be difficult to define clearly and several should ideally be applied in tandem within the same activity. Furthermore, much of the variety they inspire is captured by variety within gameplay genre; for example, narrative games tend to suggest problem-based learning, while action games are a good platform for mastery learning. Finally, my expertise in educational psychology is somewhat limited, and thus I was not confident in my ability to accurately define and apply different categories of educational strategy.
- Finally, in games with only background or contextual educational involvement, players generally do not take in-game actions that directly or implicitly assert knowledge about the material being taught. Since knowledge assertion is key to the tutoring portion of my framework, such games are unable to illustrate its full application.

As a result of these considerations, I chose firstly to discard the parameters of anatomical region, anatomical system, and educational strategy; and secondly, to ignore the parameter of educational involvement by including games with educational material primarily in the foreground. The remaining two parameters were combined to form a 3x3 box of 9 cells, with parameter levels as follows:

- Gameplay genre:
    - **Action** – games that emphasize physical and perceptual skills along with quick decision-making. Common sub-genres include shooters, racing games, timed-skill games, and many others.

- **Puzzle** – games that emphasize "slow" thinking, deduction, and careful consideration. Puzzles usually present goals that are well-defined and static, and often have unique solutions.
- **Strategy** – games based around management, tactics, and complex decision-making. Strategy games usually involve some complex system that players manipulate or interact with. Goals may be specific or diffuse and are differentiated from those of puzzles by being dynamic and often complex.

- Knowledge type:
  - **Connectivity** – information about the connective networks within the body, including nerves, vasculature, lymph, and the connectivity of structures within the musculoskeletal system.
  - **Recognition and naming** – information about the identity of structures within the body. Recognition and naming are closely linked; names can only be applied to objects if they are first recognized, and recognition involves associating an object with an identifier, typically a name.
  - **Partition and location** – information about the way structures are divided into parts. This applies particularly to bones given their role as markers for palpation and surgery, but also to muscles and organs, particularly the brain.

Having defined a morphological box, I developed a game concept for each cell within it; these are presented in section 6.2.

6.1.2.1.1 Example – Action / Connectivity

Though GMA provides a structured way to describe and explore the possibility space of solutions to some problem, it provides little advice on how one should actually imagine those solutions.

Within the constraints imposed by each cell, I used an informal combination of free-form brain-storming and de Bono's six thinking hats [228] along with inspiration from reviewing

pre-existing computer games to generate initial ideas, then explored each in more depth to create a fully fleshed out concept.

As an example, the concept Blood Racer, described in section 6.2.1, was initially inspired by the metaphor of street maps as an analogy to the body's network of blood vessels. This suggested that racing and navigation games might offer a meaningful and fun form of play that could be employed to teach this knowledge. Several games of this genre were examined for interesting and relevant features, including Wacky Wheels [229], and games within the Grand Theft Auto [230] and TrackMania [231] series. A game concept was then iteratively developed by writing a sketch of gameplay that incorporated several of these elements and then revising it over several design sessions until a reasonably sound concept resulted.

6.1.2.2 Free-form design

Though I originally intended to rely solely on a structured design process to create game concepts, it quickly became apparent that inspiration would not strike predictably. Ideas would arise during discussions with colleagues, friends, and other gamers; while playing other games; and while building the framework. Later consideration often revealed these passing ideas as flawed or unworkable, but several led to complete game concepts:

- Blood Racer – player learns about the vasculature by navigating and racing around it with a tiny robot.
- Frankenstein Wrestling – player learns the arrangement of different parts of the body by playing an "Operation" game to repair damaged Frankenstein monsters.
- FMA Walls – player builds correct ontology segments from pieces, each an entity, relation, or group of either.
- Happy Parts – player learns about anatomical sets and other groupings using classic set-building card game rules.
- Bone Finds – player learns bone recognition in a game about archaeology.

The first three of these ideas fit clearly within the space defined by the morphological box and were adapted as examples of particular cells. Two did not and are instead listed as miscellaneous. All five are documented fully in section 6.2.

The incorporation of free-form design ideas has two implications:

- Firstly, it may have influenced the way in which I defined and applied GMA; for example, by encouraging me to define my parameters in such a way that I could include them. It is perhaps impossible to determine if this was the case and what the exact impact was. That said, it seems unlikely that any application of the method could exist in a vacuum; that is, this issue must always be present unless those applying the method are entirely new to a problem. Consequently, it seems reasonable to ignore this effect.
- Secondly, in an academic context, much emphasis is placed on originality, which is difficult to assert when ideas arise naturally from activities involving others. Though I assert that the ideas presented here are primarily my creation and were conceptualized in detail without the involvement of others, I recognize that others may have inadvertently contributed to them in their early stages. To address this, I have explicitly thanked, in the acknowledgements to this dissertation, those colleagues and friends with whom I have discussed this work. In addition, when presenting game concepts in section 6.2, I have taken care to discuss how each is related to other games and genres.

6.2 Results

In total, I developed eleven game concepts, as listed in table 2, below. Five of these concepts arose from free-form design, and six from GMA. Of the five free-form concepts, three were later used within the morphological box. The remaining two ideas are included as miscellaneous game concepts. In addition to the concepts presented here, a self-generating quiz game was designed and implemented, and is presented in section 7.4.

	Connectivity	Partition / Location	Recognition
Action	Blood Racer	Zombie Hunt	La Maison Soylent
Strategy	Neural Invaders	Frankenstein Wrestling	Bone Trader
Puzzle	FMA Walls	Body Assembly	Hidden Objects

Miscellaneous Games
Happy Parts
Bone Finds

Table 2 – Game Concepts

This section presents a sketch of each concept including:

- A summary of each game's premise; that is, the context that gameplay exists within, any conceits that must be accepted for play to make sense, and an overview of what play involves.
- Details of gameplay, including notes on goals, the types of action available to players, and any scoring mechanisms.
- Notes on the educational properties of each game, including the type of knowledge taught during play and the mechanisms by which play ought to stimulate learning.
- Discussion of how each game can take advantage of the ontology-driven framework presented in chapters 0 and 0. In particular, notes on how the framework can be used to define configurations, generate content and scenes, and perform tutoring tasks.

The completeness of each concept sketch varies depending on the complexity of each game. Puzzles and action games tend to be fairly straight forward, and the sketches of them below contain much of the information needed to implement them. Strategy games,

however, tend to involve complex model systems and simulations that require substantial design and balance work that is not included here.

6.2.1 Blood Racer

Action, Connectivity

6.2.1.1 Premise

In one possible future, much surgery and drug delivery will be performed by microscale medical robots capable of synthesizing drugs from chemicals in the body, countering inflammation, destroying pathogens, and stimulating the body to efficiently heal damage. Though these robots have great capabilities, they can only act at very short ranges, and so must rapidly travel around the body to the sites of injury or disease in order to deploy treatment.

In this game concept, the player controls one such robot within the vascular system of human patient, called the "subject". During play, the subject's body receives damage that the player must heal by guiding the robot to the site of damage via blood vessels and, once there, performing the various procedures that make up treatment.

6.2.1.2 Gameplay

This concept mixes several ideas from first and third person action games with the core idea of a racing game. The player controls the robot directly, either from a first or third person perspective, with normal pitch, rotation, and yaw controls as they move through the extended tubular structure of the vascular system.

The player's ultimate goal is to keep the subject alive. To do so, they must travel to the site of damage as rapidly as possible. To create a sense of urgency, the player is scored on how quickly they respond to each piece of damage.

The player's main in-game decisions involve selecting which path to take as they travel. Damage alerts specify the site of damage, but give no guidance on how to get there. Similarly, the player is not presented with a map of the body and so must rely on their

own sense of direction and knowledge of how different blood vessels are connected. As players could become discouraged and may even quit if they regularly get completely lost, a hinting mechanism could be introduced. This might take the form of a sensor that could be periodically deployed to learn the name of the blood vessel the robot is currently within or, alternatively, a high level map of the body showing which general region the robot is in.

6.2.1.3 Education

To successfully play this game, the player must create a mental map of the vascular system by which to navigate. Game difficulty varies based on how many of the body's many blood vessels are included, allowing players to begin by mastering core systemic blood vessels before moving on to learn about progressively smaller ones.

When players first begin, they are likely to find navigation very disorienting. To address this, some scaffolding, such as a compass or orientation markers, may be required to provide a frame of reference against which they can get their bearings and plan their route. Similarly, beginning players may benefit from assistance with direction-finding; for example, if no progress is made for some time, arrows might appear to indicate which way they should go.

The educational strength of this game is that it requires that players actively process a mental map of the vascular system. Since score is based on time, successful players will need to move beyond consulting diagrams or even consciously reasoning about navigation to instead develop the same familiarity with the vascular system that drivers have with the roads of their home town.

6.2.1.4 Framework integration

The game can be configured either using scene descriptors or the curriculum module. This supports the adjustment of game difficulty and the needs of different curricula by allowing educators to limit play to certain regions or levels of detail within the body. Configuration

specifies which blood vessels are used to build the game's map of vasculature and which entities are available as possible sites of damage.

Graphical content produced by the Anatomy Engine is difficult to use in this game as the 3D models required to create a view of blood vessels from the interior are different from those used for external views. Some conversion may be possible, however, in which case content can be generated from game configurations using the Anatomy Engine. The rest of the game could be implemented using the Common Graphics Application.

Player actions can be interpreted as assertions about anatomy – if damage exists at site S, a player's choice to enter blood vessel B from vessel A indicates a belief that doing so will take them closer to S. If B is not closer to S than A, a misconception in the player's mental model is indicated. Backtracking can be interpreted either as an error or an expression of uncertainty. One limitation of this approach, however, is that the fact being asserted, that B is closer to S than A, is not a simple fact of the sort represented by individual relations within the FMA, and thus it cannot be represented by the current student model. An alternative approach is to evaluate the player's whole path from their point of origin to the target site and modify the student model's confidence in their knowledge of the individual steps on that path accordingly. If they move to their target swiftly and without error, confidence in their knowledge of the connections along that path should increase, whereas if they backtrack and hesitate, confidence should decrease.

6.2.1.5 Variations

As specified, this game concept helps players learn blood vessel connectivity but does little to help them remember names. A passive solution would be to announce names as blood vessels are entered. Alternatively, a more abstract variant of this game could require players to navigate by specifying their path as an ordered list of blood vessels through which to pass, forcing them to think about the names of blood vessels in a way that direct control would not.

Play might be enhanced by posing challenges at damage sites; for example, at a site of infection, the player might need to hunt down and destroy pathogens, whereas at a site

of injury they might need to locate and repair damage to blood vessels or organs. Such additional action would do little to improve the educational properties of the game but could substantially increase the level of engagement players feel with the game.

The complexity of play can be increased by allowing damage to occur simultaneously or in parallel at multiple sites, forcing the player to make decisions about priority and routing between damage sites. Different types of damage might have different effects on the player's score, and may vary in time sensitivity. One consequence of this change in design is that it becomes more difficult to interpret player actions for the purposes of updating the student model as it is never absolutely clear which site a player is intending to travel towards.

6.2.2 Zombie Hunt

Action, Location / Partition

6.2.2.1 Premise

According to the stories, a zombie can only be killed by destroying its brain, or perhaps its heart. In reality, necromancers have a great deal of freedom in selecting the weaknesses of any undead they raise – the only requirement is that they be weak somewhere. Thankfully, sensor equipment is now available to detect vulnerabilities in any zombie a hunter might face. Simply focus on a zombie for a second or two, and the system will shout out their weak spot.

Shooters are perhaps the quintessential video game. Many variants exist, both in perspective (first person, third person), type of environment (scrolling, rail, open world), and control scheme (twin stick, WASD, one click), all with their own tropes and conventions.

Zombie Hunt is a first person shooter in which the player hunts down the zombies that have recently begun to terrorize a major university in the Pacific Northwest. The source of these zombies has not yet been determined, but researchers have found, at the cost of

many undergraduates, that though nigh-indestructible otherwise, all zombies have one body part where they are vulnerable.

6.2.2.2 Gameplay

Play is divided into levels during which the player moves through a 3D environment searching for the exit, collecting items and destroying enemies along the way. At the end of each level, the player's performance is evaluated using statistics that describe several aspects of play, including the percentage of enemies killed, shot accuracy, time taken, and so forth.

Like many shooters, Zombie Hunt employs a simple framing narrative that provides just enough explanation to explain the choice of levels, bosses, and goals in game. In this case, the framing narrative involves a zombie outbreak at a major university, with levels based on sites around campus.

The player's primary activities during play are moving through the game environment and destroying zombies. Movement controls use a standard WASD keyboard and mouse arrangement, and are not discussed here any further. To destroy enemies, the player must shoot zombies in their weak spot using any of the various weapons found throughout the game.

The main novelty of this game compared to other first person shooters is that players cannot effectively kill zombies until their weak spots have been identified. To identify weak spots, the player must remain relatively close to a zombie for three seconds, avoiding attacks while their sensors passively gather information. Once the weak spot has been identified, it is displayed as text hovering over each zombie's head. The player may then target that weak spot and destroy the zombie.

Weak spots are described using anatomical terminology and may be very specific, requiring the player to possess good knowledge of human anatomy to determine precisely where to shoot. Zombies may, for example, be vulnerable in the gastrocnemius or the spleen rather than simply the leg or the torso. Aiming sufficiently precisely to hit such

specific body parts is very difficult, so zombies must move relatively slowly to make play fair.

6.2.2.3 Education

This game requires players to recognize anatomical terms and recall where in the body the corresponding anatomical structure can be found. Players must think rapidly under pressure while also paying attention to other information, such as movement within the game world and the actions of hostile agents. The game's intellectual difficulty is largely determined by the size and specificity of the anatomical structures used as zombie weak spots. Intellectual difficulty is also affected by how many distractions are present and how much pressure the player is under when trying to make targeting decisions. Game difficulty in terms of anatomy should be configurable when play begins, while difficulty in terms of zombie numbers and strength should increase progressively during play as players acquire better weapons and so forth.

The ability of players to correctly determine where they should shoot given the name of the anatomical weak spot of each zombie is heavily dependent on their prior knowledge of those structures. As a result, the game serves primarily to reinforce existing knowledge rather than to communicate new information. To address this, the game might be extended with targeting assistance that is shown after a delay to coach students along. Alternatively, players might collect some resource that can be spent to get hints.

6.2.2.4 Framework integration

Some game configuration is possible using scene descriptors or the curriculum module. However, for repeated play to be of interest to players, additional content, such as game maps, enemy AI, and plot must be generated, which is difficult or impossible to do well automatically. Multiplayer play is less sensitive to repetitive content.

The graphical requirements of a 3D shooter are far beyond the capabilities of the Anatomy Engine. Furthermore, anatomical structures are not displayed directly, but merely referred

to by name and used as tests for whether player shots are effective, and so it is largely irrelevant to this game.

In theory, player shots at a target zombie can be interpreted as anatomical assertions once the name of the vulnerable structure has been displayed. Unfortunately, many other factors may limit a player's ability to hit the right spot, such as insufficient time spent aiming due to time pressure, the motion of the target, and distraction due to other events occurring in the game. As a result, it is not likely that this game can take advantage of the framework's tutoring modules.

6.2.2.5 Variants

The amount of action possible in this game is limited by the requirement that enemies move slowly so that the player has ample time to aim correctly. One option is to allow enemies to move quickly, but reduce the frequency with which they do so, for example by having them attack at range while stationary. Another possibility is to adapt this game into something resembling a classic 2D rail shooter such as Operation Wolf [232][233] in which enemies remain mostly stationary in a static game world that the player moves through at a slow, fixed rate, as if on rails.

6.2.2.6 Notes

This is not the first example of a zombie-killing shooter game that has been adapted for educational purposes. "The Typing of the Dead" (TotD) [234][235] is a typing tutor built on top of the second installment in the classic rail shooter franchise "The House of the Dead" [236], [237]. Both were published by Sega as arcade game machines alongside versions for PC, mobile devices, and some consoles. In TotD, the player confronts zombies that must be destroyed by typing words or phrases that hover over their heads. No actual shooting is involved in play.

Figure 31 - In-game screenshot from The Typing of the Dead [234], [235]

6.2.3 La Maison Soylent

Action, Recognition

6.2.3.1 Premise

La Maison Soylent offers a fine dining experience unlike any other. The proprietors guard their recipes closely, and with good reason – unbeknownst to customers, much of the restaurant's success is due to unique, high-quality ingredients obtained using cloning techniques that many would consider unethical.

The player is cast in the role of a line chef at La Maison Soylent. As patrons arrive and place orders, the player must correctly assemble the requested dishes from ingredients as quickly as possible – people don't come to La Maison Soylent to wait, after all.

6.2.3.2 Gameplay

Play is largely a matter of efficiently shuffling ingredients and tasks so that all food orders are served in a timely manner. The game takes place within a kitchen that is divided into an order board, a delivery window, three to six cooking stations, and a conveyor belt for ingredients. Each order is a set of one or more dishes, and each dish is associated with a recipe. Orders vary in urgency, and service must be served prioritized accordingly.

Play proceeds as follows:

1. Orders arrive at random intervals and are displayed on the order board.
2. The player assigns individual dishes to cooking stations.
3. The player assembles each dish by transferring the ingredients specified by its associated recipe from the conveyor belt to the correct cooking station. If incorrect ingredients are assigned to a station, the dish at that station is spoiled and the player must begin it again.
4. The player performs cooking steps as specified by the recipe. If cooking steps are not initiated soon enough after the first ingredient is transferred, the dish is spoiled.
5. The player transfers completed dishes to the delivery window.

Ingredients arrive on the conveyor belt at random and must be consumed in order, either by using them in a dish or by discarding them. The player may clear the dish assigned to a cooking station by completing it, discarding it if it has become spoiled, or forcing it clear, discarding progress on that dish and returning it to the order board.

Individually, the actions available to the player are simple and can be executed with at most two mouse clicks. Each action takes a short amount of time for the player's avatar to perform, typically one or two seconds, and there is a delay as the avatar moves around the kitchen. Game complexity derives from time pressure and the need to complete multiple dishes in parallel. At any time, the actions available for the player to perform are limited: dish assignment is limited by available stations, ingredient dispatch is limited by conveyor belt contents and the needs of various dishes, cooking steps are limited by which dishes have received all of their ingredients and by delays between steps (e.g. 5 seconds for the oven), and delivery is limited by which dishes have been completed.

Points are awarded for completing dishes, serving orders quickly, correctly assigning ingredients, and meeting abstract goals, such as avoiding waste.

Anatomy enters play in the form of ingredients – apparently, the food at La Maison Soylent is people or, at least, parts of people, grown in cloning vats. Since recipes describe ingredients verbally, players must recognize body parts as they arrive on the conveyor

belt before they can correctly associate them with the names given on the recipe. Since recognition is such an important part of play, the amount of screen space dedicated to the conveyor belt should be large.

This game is inspired by time-management and coordination games such as Dead Hungry Diner [238][128] and Chocolatier [129][130], in which players must execute simple actions at high speed to serve incoming requests. These span the genres of action and strategy in that they require swift decision making and hand-eye coordination while also rewarding wise resource allocation.

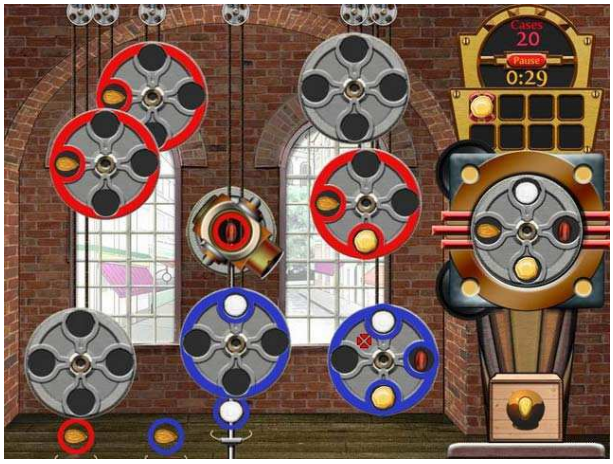


Figure 32 - In-game screenshot from Chocolatier [129][130]

6.2.3.3 Education

This concept takes an object recognition drill and dresses it up inside a more complex game to make it appealing to players. In addition to motivation, the game context provides scaffolding and associations that may make it easier for players to remember and recognize structures. By attempting to recognize structures rapidly and while distracted by many other tasks, players are forced to develop their familiarity to a greater level than might otherwise be necessary.

Since body parts are 3D objects, they ought to be presented as such during play. Players, however, will not have time to rotate and inspect structures from different angles during play. A better option, then, might be to present 3D models that have been pre-rotated

into different key orientations so that players do not lapse into thinking of them as being 2D.

6.2.3.4 Framework integration

This game requires configuration to define which body parts are available as ingredients during play. This might be provided by scene descriptors, the curriculum module, or the current state of the student model.

Much of the game's graphical content is outside the capabilities of the Anatomy Engine, as play takes place in a game-specific non-anatomic environment and structures are rarely, if ever, presented together in the same coordinate space. Assets from the Engine may, however, be useful.

Ingredient dispatch actions can be interpreted as assertions about anatomy and assessed for use by the student model. If the player dispatches a body part to a cooking station where it is required, estimates of their ability to recognize that structure ought to increase. Similarly, if they dispatch a body part to a station where it is not required, estimates ought to decrease.

6.2.4 Neural Invaders

Strategy, Connectivity

6.2.4.1 Premise

A subtle new form of mind control has been developed. Nano-machines penetrate the body at the skin then move to peripheral nerve endings which they follow up to the brain where they influence behavior. A central defense system implanted in the brainstem can combat these invaders, however, using the body's own resources to produce friendly nano-machines and sending these defenders back along the nerves to intercept the invaders.

The player controls a defense system and is charged with detecting and repelling invaders using the resources available to them. This concept is a variation on 2D side-scrolling real time strategy games such as Swords and Soldiers [239].

6.2.4.2 Gameplay

Play revolves around a stylized map of the nervous system. Depending on how the game is configured, players may see the whole body or only a part of it, in more or less detail. Once play begins, hostile signals begin to be detected at peripheral nerve endings and progress towards the central nervous system. The player must build defender nano-machines and send them out through the appropriate nerve root to intercept the invaders.

The game's main resource is energy, which is spent to build defenders and trigger special abilities, and obtained at a slow, constant rate (representing energy harvested from the body) and as a bonus every time an invader is destroyed.

Both invaders and defenders come in several types, with different speed, strength, and health parameters, and different attack modes. Invaders and defenders also have types that convey advantages and disadvantages against different types of opponent; each pair of types skews to one side's favor, forcing the player to make careful tactical decisions in deploying their defenders.

Defenders are dispatched from a nerve root; typically a vertebral spinal nerve, though other nerve roots such as the cranial nerves may be employed. Since there are many nerve roots in the body, play may be limited to some subset during initial play in order to manage difficulty. Both defenders and invaders travel along nerves at a moderate rate; a typical invader might take 15 seconds to travel up the nerves of the arm to the spine. Different nerve lengths will force players to prioritize accordingly.

Since nerves mostly branch outwards, when travelling up a nerve to its root an invader will normally have only one path to follow. Defenders, however, will be constantly faced with branches. Different defenders may behave differently at nerve branches; some may follow whichever nerve has the closest enemy, others may split and spread out, while others might simply stop and wait.

Invaders, when destroyed, yield energy that the player may use. Defenders, when destroyed, are simply removed from play. Invaders that reach nerve roots immediately

cause damage to the player; once the player's health reaches zero, the game ends and the player has lost. Alternatively, once a set number of waves of attackers have been defeated, the game ends and the player is victorious.

6.2.4.3 Education

This game forces players to develop a mental model of the course of different nerves of the body and their connection to the brain via a particular nerve root. Without a good mental model, players will be unable to make quick decisions during play and will not be able to respond sufficiently quickly to invaders.

Given the complexity of the human nervous system, it will most likely be necessary that individual play sessions focus on particular regions of the body, one at a time. The exact region would be specified as a configuration option.

6.2.4.4 Framework integration

This game concept supports configuration and content generation using the Anatomy Engine's scene repository. Each scene would correspond to a nerve map of some subset of nerves within the body, perhaps divided by region or level of complexity. Some additional graphical and path logic would be required to convert scene contents and annotations from the FMA into a working model of the game environment. For example, a graph of nerve connectivity would be required to support the pathing algorithms necessary to determine the behavior of invaders and defenders as they move along the nerves. A dedicated and stylized model set would most likely also be required to produce a nerve map that is visualized appropriately.

Like the other strategy games presented here, the decisions made by the player during play cannot be interpreted as clear assertions about anatomy and thus do not require automated assessment and cannot be incorporated into the student model. This is because the decisions a player makes in a strategy game typically incorporate many considerations that cannot be disentangled. In this game, a decision to send a defender out from a nerve root towards an invader is based on the type of invader detected, the defenders and other

resources available, the location of other invaders and thus the priority of this nerve root over others and finally, whether this nerve root connects to the peripheral nerve the invader is currently on. Of these considerations, the last is an assertion about anatomical knowledge, but is impossible to consider independently.

6.2.5 Frankenstein Wrestling

Strategy, Partition / Location

6.2.5.1 Premise

The Ingolstadt Wrestling Federation is a professional wrestling league with stakes so high that mere human wrestlers can no longer compete. Instead, following the revolutionary inventions of league founder Victor Frankenstein, competition is between monsters made from sewing together the body parts of dead humans.

Wrestlers within the league are organized into "stables" of wrestlers that train together and support each other as they compete for the Carpathian Cup of Champions. As manager of one such stable, the player is responsible for maintaining wrestlers, obtaining new and improved body parts, and setting up show matches to fund their operations.

6.2.5.2 Gameplay

This concept is a variation on sports management games such as the venerable Championship Manager series [240][241], an example of a genre that focuses on the resource allocation decisions involved in running a successful sports team. Funds are received from advertising, sponsorship, and match fees, and allocation decisions include player hiring, training, transportation, facilities, and more. In this variation, players must maintain and enhance the bodies of their wrestling monsters between fights, making decisions about the purchase of body parts[1], the allocation of parts to fighters, and the repair of fighters after a bout.

[1] Don't ask where they get them from.

Though a full game design details are not developed here, players would have as their primary goal the making of money in order to equip, maintain, and train their fighter so that they win the championship.

Play would be organized around a measure of game time that is greatly accelerated from real time. Player actions may have an immediate effect, be delayed, or take effect gradually. For example, purchases may be immediate, but fighter modification and training might take several days of game time.

A player's resources during play would include money, body parts, wrestlers, prestige, and potentially facilities that, for example, limit the wrestlers available and affect training.

At a minimum, the following actions would be available:

- Purchase a new wrestler.
- Replace parts within a wrestler. Given the educational goal of play, parts would be at the granularity of individual muscles and organs. Part replacement could simply involve designating the parts to be replaced or it could involve a surgical mini-game in which the player must remove and replace the parts in a manner similar to the games Dark Cut [124] and Amateur Surgeon [125], mentioned earlier. Parts may be replaced to repair a damaged wrestler or to give them some enhancement described by some conceit (e.g. magic, steampunk electrical powers, and so forth).
- Assign a wrestler to a bout. The effectiveness of wrestlers in a bout would be determined by the parts making up that wrestler, the wrestler's training and experience, and the power of their opponent. Bouts may cost money to enter and award a prize for victory.
- Purchase body parts. Though a more advanced trading system could be employed (see the concept Bone Trader outlined in section 6.2.6), a simple random shop that offers a selection of parts that can be bought for set values would likely be sufficient. Parts might also be received as part of prizes.

6.2.5.3 Education

This game would help players build familiarity with various muscles and organs and their placement within the body. It would not teach players to recognize structures by their appearance as they will generally be labeled, but familiarity will likely increase.

During play, the placement of structures is made apparent to players as they consider what specific parts are needed to fix a larger damaged structure (e.g. an arm), and as they consider the effect of upgraded parts on particular elements of a wrestler's performance (e.g. upgrades to the gastrocnemius might improve kicking strength while upgrades to the lung might improve endurance). This knowledge could be further reinforced using a surgical mini-game.

6.2.5.4 Framework integration

While the anatomical structures used in this game concept could be made configurable, this would likely be detrimental as the interactions between wrestlers and the effect of various body parts and enhancements must be carefully balanced with one another. The framework's tutoring components, too, are difficult to apply as the actions a player might take during regular play are complex and difficult to interpret.

A surgical mini-game wrapped inside the main game, however, could be designed to take better advantage of the framework as the Anatomy Engine could readily support the generation of content for and surgical actions can readily be interpreted as anatomical assertions. For example, when replacing a body muscle, a player must first choose the location of the old muscle, open an incision over it, remove it and not others, and correctly place its replacement. A more detailed variant might instruct players verbally on the steps required, referring to particular features (for example, asking them to cut certain tendons before removing the muscle itself).

This concept presents an example of a game that teaches with educational material in the background, as players in the base game do not need to directly employ anatomical knowledge to make in-game decisions. Furthermore, it demonstrates the use of an outer

game to frame a more directly educational activity. Framing is of benefit as it uses the narrative context of the outer game to motivate play of the inner game, which may be less engaging in itself.

6.2.6 Bone Trader

Strategy, Recognition

6.2.6.1 Premise

Many thousands of years ago, people in certain cultures relied on tribal priests to help them find solutions to their problems. In most cases, the priest would prescribe simple prayers or acts of debasement and atonement to satisfy the gods. Truly dire situations, however, required the sacrifice of sacred charms made from the bodies of the dead.

Bone Trader is a multi-player trading game in which each player is a tribal priest charged with assembling the sacrifices necessary to bring prosperity to their tribe. Since obtaining fresh body parts involves acts most people prefer to avoid, many priests maintain a collection of preserved body parts. The gods, however, are fickle, and often require sacrifices that a priest cannot serve from their own collection, and so a quiet trade has arisen between priests and dubious suppliers. Players compete to best please the gods by being the first to assemble and perform sacrifices.

6.2.6.2 Gameplay

Each player begins with a small amount of money, a set of body parts and a list of the sacrifices their gods demand of them. Each player's sacrifices are unique. The parts, sacrifices, and amount of money held by a player are private by default, but may be revealed to other players if so desired, including as part of a trade.

Play proceeds in rounds during which each player takes a turn. During their turn, a player may propose a single trade to any other player, consisting of either or both money or items. Once the offer is made, the receiving player may accept or reject it. No modification of the offer is allowed once it is made, and the receiving player's rejection or acceptance does not apply until after the offer is made. That is, players may discuss the terms of an

offer freely up to the point where the initiating player decides to make a formal offer; at that point, the receiving player may accept or reject, but may not request further changes in terms.

At the end of their turn, a player may make a sacrifice if they have all of the necessary parts. At the end of the round, players may buy and sell body parts on the black market. The parts available for purchase are unique to each player, and are selected randomly.

Items in a player's collection are represented visually and are not labeled, potentially introducing uncertainty around their anatomical identity. The parts needed for sacrifices, however, are described verbally, and players must make their own judgments about which items match.

Players receive victory points whenever they successfully perform sacrifices, with the amount varying based on the complexity of the sacrifice and whether that sacrifice has been performed before by them or another player. The game finishes at the end of a round when one or more players exceed some target number of victory points. The winner is the player or players with the most victory points.

6.2.6.3 Education

Despite its macabre theme, this game is relatively light-hearted and social. Its educational aim is to create familiarity with different body parts and reinforce students' ability to recognize them. It achieves this by requiring that players actively consider what different parts look like in order to match the descriptions used in sacrifice definitions with the visual representation of parts in their collection.

One particular strength of this game is its social aspect – play that involves direct interaction and trade between players can tap into a rich vein of experience and motivation that single player games are unable to take advantage of. Furthermore, there may be competition or collaboration between players as they apply their skills of recognition to identify parts, in particular when there is some disagreement and uncertainty.

6.2.6.4 Framework integration

As proposed, this concept describes a computer based multiplayer game. As in previous concepts where direct generation of content by the Anatomy Engine is inappropriate, it could still be utilized to configure the body parts used during play.

Similarly, it would be difficult to integrate this game with automated assessment and the student model, as most player actions do not directly make assertions about anatomy. Some information can be gained, however, as failed attempts at a sacrifice based on incorrectly identified body parts may be used to reduce confidence in a player's recognition of that structure, while successful sacrifices might increase confidence in a player's recognition of all structures involved.

6.2.6.5 Variants

Though this concept could be implemented as a computer game, implementation as a face-to-face card game may be more appropriate. Though a card game would be wholly unable to take advantage of the ontology-driven educational framework, it would support much subtler negotiation and richer social interaction that could stimulate learning in a way that other games cannot.

6.2.7 FMA Walls

Puzzle, Connectivity

6.2.7.1 Premise

In the ontology-driven UW Medical School of the future, sections of the now famous Foundational Model of Anatomy are proudly displayed on the walls of corridors and common rooms. FMA entities and relations are represented as interconnected magnetic word strips that are periodically rearranged as needed to show different aspects of the ontology. Unfortunately, every now and then a careless student brushes up against the wall and destroys all or part of one of the displays. This upsets the mad ontologists in the basement, and students have learned that when this happens, it's in their best interests to fix things as quickly and correctly as possible.

In FMA Walls, the player is one such student. Having just knocked the ontology to the floor, all of its pieces are scattered and must be correctly re-assembled as quickly as possible.

6.2.7.2 Gameplay

At the beginning of each game, the player is given a collection of disconnected ontology pieces, where each piece is an entity, a relation, or a fragment containing a small number of entities and relations already linked to one another. They must reconnect all of these pieces correctly to recreate some portion of the FMA.

When all pieces are connected and the player indicates that they are satisfied, their arrangement is assessed and they are informed how many connections are correct and how many incorrect. During play, the player may request some limited number of hints, in which case either a single error in their current arrangement is indicated or they are informed that no errors exist.

It is possible that a given set of relations and entities may be connected in different ways that are all correct. For example, if a single instance of the "articulates with" relation is available in a puzzle with multiple bone entities, it does not matter which pair of bones it is placed between so long as this placement is valid within the FMA.

6.2.7.3 Education

In this concept, players engage directly with the FMA as an abstract representation of anatomical knowledge. Both memory and reason are employed, as players must recall what they know about each entity but also reason about what relations are available and how they might be used.

To make sense of this game, players must first understand the concepts of an entity and a relation within an ontology and how entities and relations may be combined to create subject-predicate-object statements. In addition, they must also know the meaning of each type of relation as defined by the FMA.

This game relies on players having at least some knowledge of how different entities are related. Deduction may be employed during play to limit possible connections, but some initial knowledge is required for players to build from. As a result, this game is best suited for reinforcing knowledge rather than teaching it. Scaffolding could be employed to assist players whose knowledge is limited; for example, they might be shown the network briefly before beginning the puzzle, thus allowing their memory of the overall shape to inform their construction efforts. Alternatively, hints might be shown that limit the connections a player must consider; for example, by stating that a particular entity must be associated with another by a particular relation type.

6.2.7.4 Framework integration

Game sessions may be configured and generated using the query engine and curriculum module. As this game uses only text labels, the graphical capabilities of the Anatomy Engine, however, are irrelevant, though variants might employ images of anatomical entities as well as text.

Integration of this game with the ontology-driven educational framework is straight forward, as players directly assert ontological relations as they build the network. The automated assessment module can be used whenever a relation needs to be tested, and the student model may be updated either at the end of each session when the whole network is evaluated, or individually as the player creates it.

6.2.7.5 Variants

A slightly more complex variation on this game could use blank entities and relations. These might simply be wildcards, or they may be open fields that the player must fill by entering correct entity or relation names.

Another, potentially easier variation would be to present the ontology as a network whose nodes and edges lack labels that players must fill from a set of available entities and relations. Not all nodes and edges need be blank – some may already be filled as clues.

Conceptually, this is very similar to the cloze tests used in reading comprehension, where students are asked to fill in blanks in some piece of text.

6.2.8 Body Assembly

Puzzle, Partition / Location

6.2.8.1 Premise

Assembly puzzles are a type of physical puzzle in which an object must be assembled from a number of smaller parts. A common first encounter with them is as curious looking objects sitting on a mantelpiece or coffee table, just begging to be picked up and inspected. Almost immediately, they fall apart. Courtesy demands re-assembly before re-shelving, but this is usually not as easy as their original form would have suggested, and so begins a protracted distraction, often to the amusement of one's host.

Assembly puzzles can be made from any material but are common projects for hobbyist woodworkers [242]. Most have a simple abstract form; common shapes are pyramids, cubes, spheres, and knots, such as the example shown in Figure 33. More complex forms may be represented, however, as in the case of physical anatomical models, many of which are intended to be dis-assembled and re-assembled by students as part of a learning activity; one such model is shown in Figure 34.

This game concept is a web-based anatomical assembly puzzle for teaching students about the partition of different body parts.

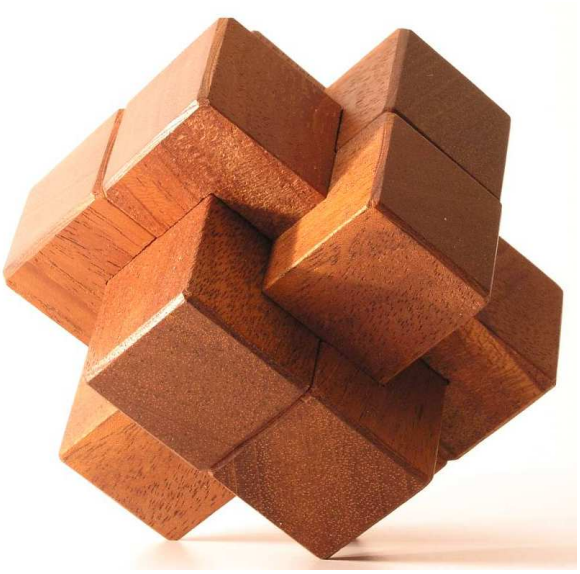


Figure 33 – Chinese Knot Puzzle

Figure 34 – Anatomical Model; Deluxe Dual Sex Torso, 20 part, from 3B Scientific

6.2.8.2 Gameplay

Play is divided into levels, each a separate puzzle. At the beginning of each level, the player is presented with a workspace filled with models representing the parts of one or more anatomical structures. Parts may be picked up by clicking on them, and rotated or moved using the keyboard and mouse. If the player places a part against another part and clicks with the right mouse button, the game attempts to join them. If the two parts are from the same structure, adjoin each other in that structure, and are positioned and oriented approximately correctly with respect to one another, they are joined together into a single larger part. If not, a warning sound effect is played and nothing else happens. Once all possible joins are made, the level ends and the player's performance is evaluated.

The player's score is determined by how quickly they were able to complete all structures, with penalties for failed join attempts. In addition, a star rating is awarded for that puzzle, where one star indicates a puzzle that was completed both slowly and with many errors, two stars indicates a puzzle that was complete either quickly or without errors, and three stars indicates a perfect completion; that is, both quickly and without errors.

Stars are collected across all puzzles attempted by the player, and are used to unlock new puzzles. Mechanisms of this sort are common across casual puzzle games, and are thought to encourage players to attempt puzzles multiple times in order to achieve perfection. Stars also act as motivators by giving players rewards for achievement and, when presented in a summary of past levels, by making it clear to players where they have failed in the past, and where they might want to return [243].

Unfortunately, users find rotation of 3D objects using keyboard and mouse to be particularly challenging [244]. Consequently, players should not be expected to get the position and orientation of parts with respect to one another exactly right. One possibility is to accept as correct any placement that is approximately correct; alternatively, orientation may be limited to a small set of possibilities.

6.2.8.3 Education

This concept forces players to think of anatomical structures as three-dimensional objects rather than as the flat entities they are presented as in textbooks and other print resources. This improves a player's ability to recognize structures from multiple perspectives and encourages them to develop a knowledge of how each structure's features and landmarks are related to one another and how they can be used to determine orientation. When connecting structures to one another, players similarly build up knowledge of the spatial relationships between them as well as how each structure's distinguishing features can be used to indicate where connections exist.

This game does not seek to directly communicate to players how parts are connected; the intent is that players work this out themselves by looking at parts and considering how they might fit together. Scaffolding could, however, be employed to teach players where connections should exist. For example, colored balls might be used to mark connection points on each part such that colors are never used more than once on the same structure but a minimal set of colors is employed overall. This would limit the possible connections between parts and assist players in discovering connections themselves. Other possibilities

might include hints available during play or an animation before play that shows structures being disassembled so that players might have some idea where to start.

Assembly puzzles are thought to generally improve spatial reasoning, which has been shown to correlate well student's performance in learning anatomy [245].

6.2.8.4 Framework integration

Configuration of this game involves specifying what structures and parts should be included. In addition, spatial information is required to specify how structures are connected. If structures are represented as models in a shared coordinate space, this can be used to determine the correct position and rotation of each. The Anatomy Engine supports much of this functionality as is, and could be most likely be adapted with modifications to the Common Graphics Application. Additional graphics may be required, however, to depict structures in various stages of disassembly.

Integration with the framework's tutoring components is relatively straight forward, as each attempt to join two parts together is an assertion that both are parts of some larger structure. This fact is not directly represented in the FMA, but is implied by both parts being related to some larger part by the `fma:regional_part` relation.

6.2.8.5 Variations

A major limitation of this game over physical anatomical models is that it does not engage the kinesthetic abilities of players to the same level. Evidence suggests that spatial skills are learned best through physical interaction with the world. As an alternative to rotation and placement with the keyboard and mouse, players might interact with model parts using a tangible user interface [246] as demonstrated, for example, in a system for teaching molecular biology using physical surrogates in an augmented reality environment [247].

As defined, this concept focuses on partition relationships, specifically regional partitions; that is, how a particular structure is divided into parts by region. Other spatial relationships

could be used instead; for example, blood vessel connectivity, bone articulation, or general spatial relationships.

6.2.9 Hidden Object puzzles

Puzzle, Recognition

6.2.9.1 Premise

Hidden object puzzles are a type of game activity where players are asked to locate and identify objects hidden within a noisy environment. Typically, they are not standalone games in their own right, but are strung together within a framing narrative that gives the overall experience cohesion and meaning, for example by explaining why certain objects must be found. These framing narratives are almost always part of casual adventure games, usually with a mystery theme; for example, solving a murder, discovering lost artifacts, or uncovering some supernatural secret. These fit well, as they blend the gameplay of finding objects with the narrative of finding answers. One example, the games of the Elizabeth Find M.D. Diagnosis Mystery series [248], are hidden object mysteries with a medical theme that unfortunately make no effort to include real medical ideas or anatomy.

This concept describes how hidden object puzzles might be used to teach students to recognize anatomical structures.

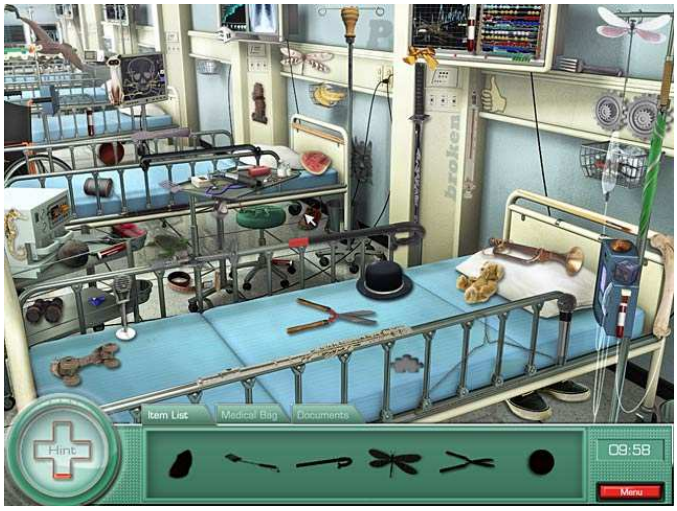


Figure 35 - In-game screenshot from Elizabeth Find M.D. [248]

6.2.9.2 Gameplay

In a hidden object puzzle for teaching body part recognition, the player is presented with a scene rich in background details and other sources of visual complexity. Overlaid on this image are a large number of objects, including some that the player must find. It is normal that neither the objects nor their placement make much sense – in general, the goal is to create a challenging puzzle, not to realistically depict how, say, a storage cupboard might be laid out. In this case, it would particularly difficult to explain why body parts are lying around in a scene.

In most hidden object puzzles, the player is given a list of the objects they must find and may do so in any order. In this game, however, the player is told to find objects one at a time. Objects are listed by either their name or a description. Silhouettes are used in other hidden object puzzles but are perhaps not appropriate here as they would reduce the need for players to remember what objects look like themselves. In addition to the target objects, the scene is filled with distractor objects. These cannot be too visually distinct from target objects as otherwise they would be easy to ignore. Objects may overlay one other slightly but should not do so to such an extent that identification of the concealed object is impossible.

The player indicates that they have found an object by clicking on it. If they click on the correct object, they are awarded points. If they click on the wrong object or on the

background, they lose points. If a player is stuck, they may request a hint. This costs points, but helps them move onto the next object.

6.2.9.3 Education

Hidden object puzzles are slightly more sophisticated than recognition drills, such as flash cards. In them, players must transform names and descriptions into some idea of what body parts look like before attempting to find them in a field full of distractors. In particular, they must recall each part's distinctive features so that they have something specific to look for.

As they are based on recognition, hidden object games require that players have some prior knowledge of the objects they are asked to find. Hinting mechanisms can be employed to give students with limited prior knowledge a clue or, alternatively, players might be provided with a reference guide that shows the desired objects either fully or limited in some way, perhaps as silhouettes or from only one key position, if the puzzle is 3D.

6.2.9.4 Framework integration

A basic hidden object puzzle could easily be built around the Anatomy Engine. Scene descriptors and the curriculum module can be used to define which objects must be found and which can be used as distractors. If descriptions are preferred instead of names to indicate which objects must be found, simple clues may be compiled using information from the FMA, for example, by describing a target object as being a bone found in the arm that articulates with the scapula.

Player actions can be assessed if it is clear which entity they are searching for when they click on an object. This can be achieved by asking them to find objects one at a time thus allowing the tutoring module to assess the correctness of each selection and use this information to update the student model.

6.2.9.5 Variations

Most hidden object puzzles use only two-dimensional images. This makes the angle from which objects are presented very important. An alternative might be 3D objects hidden in a 3D space. For example, objects may be hidden within an overflowing storage cabinet that the player can look through, or they may be scattered around a 3D environment such as a morgue. A 3D hidden object puzzle would need to allow players to pick up and manipulate objects to see them from multiple angles before they can assert that they have found what they are looking for.

Different hint policies can also significantly alter the experience. In an anatomical hidden object puzzle that uses descriptions rather than names, clues can either serve to better describe which entity is being sought, for example by giving additional facts, or they can give information about what that entity looks like by, for example, showing a silhouette.

Depending on how tightly focused the designer wants the game to be on anatomical material, players might be asked to locate other objects in addition to anatomical parts. If embedded within a framing game, the discovery of anatomical parts might be given additional meaning; for example by using the parts discovered by the player across multiple puzzles to, for example, re-construct a cadaver.

6.2.10 Happy Parts

Miscellaneous

6.2.10.1 Premise

Happy Parts is a set collection card game inspired by the classic games of Go Fish [249] and Happy Families [250], targeted primarily at younger players. Players begin with a hand of cards and must create and play sets of cards until the deck is empty. The player who has played the most sets of cards wins.

6.2.10.2 Gameplay

Happy Parts uses a deck of cards, each representing a body part and each a member of one or more categories; for example, the "biceps brachii" card might belong to the categories "muscle" and "part of upper arm".

To set up the game, each player is dealt 7 cards. Players then take turns asking players for cards and playing sets if they can. The game ends when a player is required to pick up a card and no cards remain available in the deck.

On their turn, a player may do one of the following:

- Ask one other player for a particular card: if the asked player has that card, they must turn it over and the asking player gets another turn. If the asked player does not have that card, the asking player must draw a card from the deck.
- Ask one other player for a card of some category: if the asked player has a card of that set, they must turn it over. If they have more than one, the asked player chooses which card to hand over. The asking player does not get another turn.
- Play a set of cards. A set is made up from one of each unique card in some category; for example, all muscles, or all parts of the upper arm.

Note that playing a set is an action that requires a turn to perform. A player who has received an extra turn by requesting and receiving a particular card may immediately use their extra turn to play a set, but otherwise, playing a set takes a whole turn.

The contents of each player's hand of cards is private, but all card transfers are public. The deck is kept face-down in the middle of the table. All sets are played publicly and may be examined at any time by any player.

6.2.10.3 Education

The goal of Happy Parts is to teach younger students about the names and organization of various body parts. Though players must associate body parts into categories during play, the game does not force them to reason with this information, and so primarily acts

to create familiarity. As it is intended for use by younger students, cards should correspond to relatively macroscopic structures; for example, each region of the body might be made up from a nerve, a blood vessel, a muscle, and a bone found in that part of the body.

Like Bone Trader, Happy Parts is a social game that seeks to create familiarity and fluency with body parts rather than attempting to teach more advanced knowledge or reasoning skills. Happy Parts thus requires no prior knowledge on the part of players.

6.2.10.4 Framework integration

As conceived, Happy Parts is a card game and thus cannot be integrated with the ontology-driven educational framework. While it could be implemented on mobile devices so that both framework integration and face to face play is possible, the usability of mobile games as a substitute for physical card games has not yet been established, and cards may remain the optimal format for this game concept.

Happy Parts is included in this selection of game concepts despite its inability to interact with the framework as it demonstrates an approach to educational play that is much more casual and play-oriented than most other activities. Furthermore, as its rules are near identical to Go Fish, it provides an example of near-seamless integration of educational content into a game that children already play for fun.

6.2.11 Bone Finds

Miscellaneous

6.2.11.1 Premise

An important part of archaeology is the identification and curation of finds. When bones are found, they must be identified and interpreted based on their individual shape and properties, the context they are found in, and their relationship to other bones. Researchers use this identification to determine how many individuals a given collection of bones constitutes and, ideally, to learn something about them; for example, by interpreting bone damage and growth patterns.

In Bone Finds, the player plays the role of a bone specialist at an archaeological dig. They are presented a collection of unlabeled bones and must identify each by assigning a name to it.

6.2.11.2 Gameplay

During each game session, a player is presented with a collection of bones that they must identify. Bones are organized into a grid and are thus free of any archaeological context – this is because the game focuses on identifying bones, not on interpreting archaeological sites. Players identify bones one at a time, in any order.

To identify a bone, a player selects it from the grid. All other bones disappear and the selected bone is shown at full size. The player may rotate or examine the bone using normal camera controls. They may also consult a reference guide containing both a whole skeleton and individual bones. Once they are ready to make their identification, they type the name of the bone into a field. Assistance from text completion is provided to ensure that the standardized name for each bone is used.

Points are awarded at the end of an identification session – players do not receive feedback from identification until they submit all identifications. Correct identifications on the first try win full points. Incorrect identifications are not corrected, but marked for the player to review and re-submit. Points are awarded for correct identifications in subsequent rounds, but in lower amounts depending on how many attempts the player has made.

6.2.11.3 Education

By only providing feedback once all bones are identified, the game separates feedback slightly from player actions. This prevents players from finding answers by quickly trying all possibilities and is unlikely to cause significant problems, as feedback is given for each identification individually, preventing any possibility of confusion about where mistakes were made.

Bone Finds focuses primarily on teaching students to recognize bones. It may be extended to ask them to identify bone fragments, thus also teaching bone partition. A further

extension might to require that students place identified bones correctly on a skeleton, thus teaching location. This could be an alternative to asking them to name structures or both could be required.

Bone Finds can be played by players with or without prior knowledge, with the only difference being in terms of difficulty. Players who have little experience with bones should be able to play effectively as all necessary information is available via the reference skeleton.

6.2.11.4 Framework integration

Bone Finds may be limited to certain sets of bones, allowing play to meet the needs of a particular student or curriculum. Configuration may be achieved using scene descriptors or, more likely, using the curriculum module.

The graphical needs of Bone Finds are sufficiently modest that it can be fully implemented as application built on the Anatomy Engine's Common Graphics Application.

Bone Finds also integrates well with the tutoring features of the Anatomy Engine. Names assigned to bones can be directly verified using automated assessment and the outcomes of this can feed readily into the student model. Similarly, the activity may be configured to focus on different collections of bones, allowing it to adapt to different curricula. If the archaeological premise is not taken too seriously, it can even be extended to structures that would not normally survive to be found during a dig.

6.2.11.5 Variations

Several variations on this game concept are possible:

- As mentioned, partial bones and anatomical structures that are not bones could be used, though, in the latter case, this would make the archaeological premise less believable.
- As described, Bone Finds is played as a series of standalone activities. Alternatively, it could be woven into a larger game in order to create a more well-rounded experience. This might include the addition of more archaeological content; for

example, presenting bones in context and expecting players to use this to interpret their finds.

- Multiple sets of models could be used, with different levels of damage and normal anatomic variation. For example, while the reference skeleton might contain "perfect" models that depict the well cared for bones of an anatomical collection, bones found in the field might use models that depict bones that are degraded or damaged.

6.3 Conclusion

This chapter was dedicated to answering research question C, which asks how educational games can be designed to take advantage of an ontology-driven learning framework.

It achieves this by describing an application of General Morphological Analysis along with notes on free-form design outside of that structured process. Eleven game concepts were presented that together form a library of possibilities for the framework. These concepts propose approaches to interesting educational gameplay, illustrate the types of problem that the ontology-driven framework is capable of addressing, and provide a detailed examination of the concerns that must be considered in implementing games to take advantage of it as well as many of its limitations and strengths.

The remainder of this chapter presents reflections on the process and some observations about the games proposed.

6.3.1 The effectiveness of GMA

GMA was an effective method of exploring possible designs for educational games based on my framework. My early attempts to forge ahead with a more informal process of my own devising was fraught with difficulties, but once I adopted the method proper, these problems fell away and I was able to relatively swiftly work through the process and generate the games presented in this chapter. My experience here is instructive,

illustrating the necessity of clearly specifying one's goals when embarking on a design activity such as this.

The strength of GMA, it seems, comes in large part from the way it isolates the various activities within design. One is encouraged to specify one's problem separately from thinking about how it is solved, affording clarity that would otherwise not exist. Similarly, rather than roiling around in a sea of possibilities as one might during a brainstorming session, I found that GMA facilitated clear thinking about possible solutions one by one.

The structured nature of GMA may not, however, be suitable all types of design problem. Relatively simple design problems would probably not benefit from it, and the overhead it imposes would most likely be harmful. Similarly, its rigorous approach and aim of choosing solutions at least somewhat objectively makes it inappropriate in highly subjective domains such as artistic design. Its strength becomes apparent, however, when it is applied to solving problems that are complex, both in terms of possible solutions, problem definition, and solution outcomes.

In contrast to the many hundreds or thousands of cells in some of the large scale applications of GMA documented in the literature, I chose to use a relatively small morphological box with only nine cells. These parameters were few yet broad, covering three major genres of game and the three dominant categories of declarative anatomical knowledge. This enabled me to cover a large part of the design space and forced me to think about several types of game that I would not otherwise have considered. As a result, I have been able to propose games that vary widely in the educational approaches and styles of play they employ. A consequence of taking such a wide aim with so small a box, however, was that each cell was defined quite broadly, a fact made worse by the rather flexible nature of genre definitions. Consequently, it is not possible for me to claim that the ideas I finally selected for each cell are fully representative of the possibilities within. Rather, I am limited to claiming that my games are strong examples that cover a broad array of possible designs. Luckily, this meets my objective of demonstrating the variety of ways games could be designed to make use of the ontology-driven framework.

Though I can be no means claim that my application of GMA was perfect, it was certainly effective, and I would recommend it for use in similar situations.

6.3.2 Uses of the framework

The uses made of the ontology-driven framework by the games presented in this chapter address four general classes of problem:

- **Content generation** – Whole scenes are not generally employed in the game concepts proposed, but assets, style sheets, and the results of queries are used to generate content in several. FMA Walls, for example, generates a puzzle from query results, while assets and stylesheets are used to create content in La Maison Soylent, Bone Trader, and others.
- **Content configuration** – Most of the games presented are configurable in that they allow an educator to select which particular anatomical structures are to be taught and used during play. Both scene descriptors and curriculum definitions are suggested by different games. For example, in Blood Racer, the vasculature used during play is defined by the blood vessels contained within a scene, while in FMA Walls, puzzles are assembled around the entities declared in a curriculum. In this context, the Anatomy Engine is a tool for defining how a game template creates its own content, as opposed to using content already created using the scene builder.
- **Assessment** – Many of the games proposed involve player actions that can be interpreted as assertions about anatomy. The framework may assess these assertions against the FMA and the results can be used to update the student model and inform in game events, such as the summaries given at the end of a session of FMA Walls or Body Assembly. Several issues in the assessment of gameplay are discussed below.

6.3.3 Macabre game themes.

Several of the game concepts presented in this chapter are rather morbid. This may be due to my sense of humor, but is also a limitation of the content. Anatomy is inherently

about human body parts, and outside medicine, there are few narrative premises available involving body parts that do not tend toward the macabre.

6.3.4 Assessing gameplay

In educational games, most of the actions taken by players are at least partly based on their knowledge of the subject being taught. When the criteria normally used to choose an action is mostly based on domain knowledge, a player's choice of that action can reasonably be interpreted as an assertion about that knowledge.

For example, in FMA Walls, when a player connects two entities by a relation, the only criteria they would normally be expected to have considered is whether those entities are actually related. Of course, the action may have been a random guess or due to some irrelevant belief, in which case it is incorrect to interpret it as an assertion. Unfortunately, it is impossible to tell these situations apart. Despite these limitations, it seems reasonable to use the interpretation of player actions to update estimates in the student model, as discussed in section 5.2. Furthermore, this problem exists in other testing schemes; for example, in multiple choice tests it is impossible to determine if a student selected an option because they knew the right answer, because they were guessing, or because they were unsure and that option had not been used recently.

Interpretation of actions is impossible in games where player decisions are affected by factors other than domain knowledge. For example, in a strategy game such as Frankenstein Wrestling it is impossible to know that a player chose to upgrade a wrestler's biceps because they knew it would improve their arm strength or because they thought the cost of that upgrade was affordable given the resources they had available. Similarly, in an action game such as Zombie Hunt, it is impossible to determine if a player shot a zombie just below the ribs because they knew the spleen was located there or because the target moved unexpectedly while they were aiming for the leg. In these cases, it is not possible to update the student model as a result of play.

One approach to this limitation is to constrain player actions in certain parts of the game so that anatomical knowledge is the only relevant consideration. For example, in Frankenstein Wrestling, a surgery mini-game is proposed that requires a player to take several anatomy focused actions in order to correctly install new body parts in their wrestlers. The actions within this mini-game can then be interpreted as assertions and used to update the model.

If no way can be found to interpret a player's actions, their educational progress must be assessed in some other way. Normal methods such as exams, presentations, and interaction with a human tutor are all possible. Alternatively, games that do not support assessment may be interleaved with games that do – if both cover the same material, learning from both will be captured in the student model by assessment in the one that does. This is analogous to classroom activities that are not themselves directly assessed but that are thought to help students learn in order that they will perform well in exams.

6.3.5 Framing games

Learning activities vary widely in the motivational scaffolding they provide players. Many worthwhile activities, particularly puzzles, are themselves not particularly engaging over the long term yet are valuable from an educational standpoint. One strategy for addressing this problem is the use of a framing game that wraps the learning activity and adds additional motivational structures to engage students.

Framing games enhance engagement in two broad ways. Narrative framing involves wrapping learning activities within a story that engages students emotionally through their concern for characters in that story and a desire to find out what happens next. Gameplay framing wraps the learning activity in mechanics that confer additional meaning to actions taken within the activity. Examples of gameplay framing include achievement awards such as star-based progress goals, and meta-puzzles, where events in the learning activity contribute towards larger goals, such as bonus objects found across several activities that unlock access to a special weapon.

Hidden object puzzles, as described in section 6.2.9, provide an excellent example of this strategy. Hidden object puzzles are themselves not particularly engaging, and are thus normally framed within casual adventure games that provide both narrative and gameplay framing. Basic point and click adventure controls are used, allowing players to move around the game world by clicking on doors, containers, and other objects in images of rooms and other environments. This creates an illusion of control as the player may explore the world as they choose, but progress within the game's main plot is usually close to linear and constrained by puzzles such as, for example, finding a key to unlock a door. Games of this variety typically employ a number of types of puzzle of which hidden objects are only one prominent example; others may include solving riddles, decoding messages, logic puzzles, and spatial puzzles such as tile arrangement and jigsaw puzzles. A similar strategy could be employed to create medical adventure games that wrap a series of anatomical learning activities.

Chapter 7. Ontology-driven learning activities

Chapters 4 and 5 present answers to research questions A and B by presenting a series of software components that illustrate the ways an ontology such as the FMA can be used to support content generation and tutoring. Chapter 6 then addresses question C by offering a variety of educational game concepts that have been designed to take advantage of the features offered by an ontology-driven educational system.

In this chapter, I describe how such a system is formed by the various components described throughout this dissertation. I begin by describing two additional software components, the first responsible for managing learning activities, and the second for making recommendations to students about which activities they should attempt and in what order. Next, I describe the overall system, dubbed the Anatomy Learning System. Finally, I present a self-generating quiz activity as an example of how a learning activity might be constructed within this system.

This chapter is divided into four parts, followed by a conclusion:

- Section 7.1 describes the Activity Manager, where learning activity templates are configured into instances and registered with particular curricula to form exercises, with notes on usage and implementation.
- Section 7.2 presents the Recommendation Engine, a tutoring component that assesses exercises against a student's current knowledge, as estimated within a student model, to guide them on to future learning activities.
- Section 7.3 summarizes how the components presented in this dissertation fit together to form the Anatomy Learning System. Also included is a description of

the user experience of both educators and students as well as a high-level overview of the system's design.

- Finally, section 7.4 presents a self-generating quiz activity as an example of how a learning activity might be built using the System.

7.1 Activity Manager

The purpose of the Activity Manager is to provide a means for activity authors and educators to configure learning activities and make them available to students enrolled in particular curricula.

Before proceeding, several terms must be defined:

- **Activity Template** – a learning activity in its most general, content agnostic form. Each template is a software application that can be configured to teach particular content and consists primarily of game and user interface logic. An example activity template is the concept of a multi-choice quiz; that is, the list of rules and conventions used to pose a particular quiz.
- **Activity Instance** – a learning activity that has been configured with content or instructions for how content should be derived, reifying it from an activity template into an actual, concrete activity that is ready for a student to attempt. The relationship between an activity template and an activity instance is analogous to that between the concept of a quiz and an actual quiz with questions defined and ready to answer.
- **Exercise** – an activity instance that is associated with a particular curriculum. Though the configuration of an activity instance may specify curricula for use in content generation, it is not formally registered by the Activity Manager as being available for use by students enrolled in a curriculum until an exercise is created. It is this binding that specifies to which student model student actions will be applied as exercises are completed.

An activity template may be an educational game, a quiz, or any other form of educational content that can be invoked via a URL. Static activities such as reading may be created using a pre-defined "static content" template that can be configured into an instance by providing a URL where that content can be found. Each of the game concepts described in chapter 6 could be implemented as an activity template for inclusion in the system.

The Activity Manager provides users with a series of administrative web pages where activity templates, activity instances, and exercises can be created and managed.

7.1.1 Usage

Using the Activity Manager's administrative web pages, users may:

- Create, update, and delete activity templates.
- Create, update, and delete activity instances.
- List the exercises associated with a curriculum.
- Register an activity instance with a curriculum to create an exercise.
- Delete an existing exercise.

7.1.2 Implementation

The Activity Manager is implemented as a web application written in Java with Spring MVC, running on Tomcat, with storage provided by the PostgreSQL relational database. It integrates with the Curriculum Manager, the Scene Builder, and the Student Model using web services and Java APIs. The Activity Manager provides a Java API through which it is accessed by the Recommendation Engine and the Anatomy Learning System's frontend.

An activity template is described by:

- A name.
- A description.
- A URL where instances based on this template may be invoked.

- A URL where the facts taught by a particular instance based on this template can be found.
- A list of parameters, each defined by:
    - A label.
    - A parameter name.
    - A type, being one of number, string, scene ID, curriculum ID, and student ID.

An activity instance is a particular configuration of an activity template. An activity template may be used by multiple instances. An activity instance consists of:

- A name
- A description
- An association with a particular activity template
- A list of parameter values. Each parameter value is defined by:
    - A name, which must be one of the names assigned to parameters in the associated template.
    - A value, which must match the type of the associated parameter.
- A list of facts that will be taught by this instance. This list is generated by the activity template when the relevant URL is invoked with the parameter values defined by this instance passed in as HTTP query parameters.

An activity instance may be registered with a curriculum to create an exercise. An exercise is defined by a reference to an activity instance and a curriculum. When a student comes to attempt an exercise, this curriculum is used along with that student's ID to determine which student model should be updated during play.

The stages an activity template passes through to become an instance, an exercise, and eventually a game to be played are shown in Figure 36.

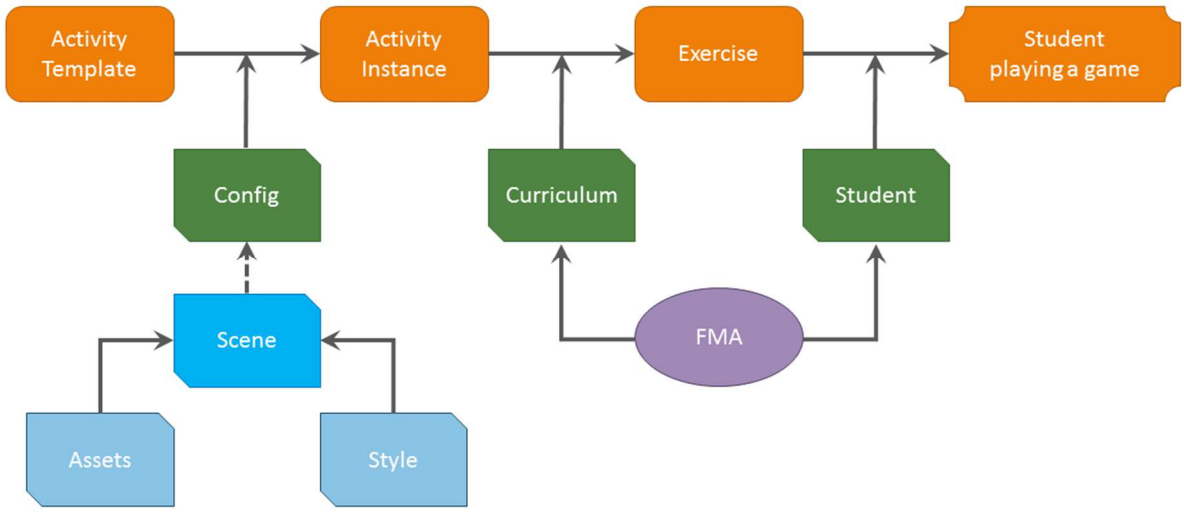


Figure 36 - Phases of an activity

7.1.3 Further Development

The Activity Manager satisfactorily meets its requirements as currently defined. No further development is suggested.

7.2 Recommendation Engine

Aside from evaluating student actions and giving direct feedback, a major role of any human tutor is to provide students with guidance and suggestions for where they should focus their effort.

Computerized tutoring systems can offer guidance by assessing the various activities a student might perform in order to determine which is likely to have the most beneficial effect on their education. A variety of algorithms might be employed to do this; in the relatively simple implementation presented here, recommendations are produced by comparing a student's current knowledge with information about the material covered in each activity to calculate a rating that is then used to rank activities in a suggested order of priority. More sophisticated approaches might weight activities so that new material is introduced in some order according to pedagogical principles such as mastery learning.

The Recommendation Engine operates on exercises, as described in section 7.1, each being an activity instance bound with a curriculum. Recommendations are calculated on a

per-student, per-exercise basis, with exercise curriculum and current student ID used to select a student model for use in calculations.

The recommendation is an essential part of the Anatomy Learning System's student and educator overview pages, described in section 7.3.1.

While the Recommendation Engine provides students with guidance on a per-exercise basis, guidance might also be provided at a more granular level; for example, a quiz activity might provide fine level guidance by selecting quiz questions according to the student's current knowledge, with the aim of drilling them on material where they are weak. Such fine-grained guidance is specific to the design of a particular activity, and is therefore not discussed here, though similar principles might be employed.

7.2.1 Usage

The Recommendation Engine presents no user interface, as it is intended solely as a service for use by other software components.

7.2.2 Implementation

The Recommendation Engine is implemented as a web application written in Java with Spring MVC, running on Tomcat with a PostgreSQL database for storage. It is made accessible to the Activity Manager and Anatomy Learning System via a Java API.

Recommendation ratings are calculated as numeric values between 0 and 1, where a high value indicates an exercise that is highly recommended. A recommendation rating should be interpreted as an indication of how much a particular student will be exposed to material they do not know by completing a given exercise. Ratings may be requested individually, for a particular exercise and student, or as a group, for all exercises associated with a given curriculum and student.

Ratings are calculated as the mean of the probability values in some student model of all facts that the activity instance associated with a particular exercise claims to teach, subtracted from 1. The student model used in this calculation is determined by the

curriculum associated with an exercise and the student for whom the recommendation is being calculated. For example, an exercise that claims to teach the origin and insertion of muscles of the leg will be rated by averaging all relation facts in a student's model that concern muscle attachment then subtracting this result from one. If the system has low confidence in a student's knowledge of these facts, that exercise will be highly rated, and vice versa.

7.2.3 Further Development

As implemented, the Recommendation Engine is relatively simple, providing recommendations based on how well a student knows the subject matter taught by a given exercise. Improvements are possible, as discussed below.

7.2.3.1 All exercises are treated equally

The current rating algorithm treats all claims about the facts taught by a given exercise equally, despite the fact that different types of activity are likely to vary in how well they teach different types of anatomical knowledge. Such a simple approach is an inevitable default, as more sophisticated algorithms require concrete information about the efficacy of each type of activity.

Gathering this information is both difficult and expensive in terms of time and effort. Two general families of approach are possible:

- **Data driven** – Students are tested before and after an exercise in order to understand how well that exercise has improved their knowledge. Studies of this nature can be time-consuming and expensive to conduct well, and may present significant methodological difficulties. Simplistic approaches such as observing the effect on the student model of a student's completion of an exercise are misguided, as there is no guarantee that the activity that is being tested will update the model wisely, making the its estimates untrustworthy.
- **Expert driven** – Different types of activity may be assessed by experts following some formal method. This approach is inherently subjective, but may be

sufficiently rigorous to be worthwhile, particularly considering how difficult and expensive it can be to obtain accurate results from data driven evaluation.

Once a method is devised for calculating how effective an activity instance is at teaching particular facts or groups of facts, more advanced recommendation algorithms may be implemented in which exercises are recommended based on how effectively they teach the facts a student currently needs to know.

7.2.3.2 Support for mastery learning

The algorithm described here recommends exercises based on which facts a student does not yet know, irrespective of any ordering an educator might want to impose. In particular, this prevents the use of the system within a so-called mastery learning approach, where students are led through a series of educational units one-by-one, with progress to each new module contingent on the student first mastering previous material.

A more sophisticated recommendation system should offer support for educators to specify the order in which exercises become available. One approach might be for each exercise to specify a set of threshold constraints that must be met before it is made available to students. For example, given exercises A, B, and C, constraints on exercise C could specify that it would not become available until students had exceeded 0.8 on exercise A and 0.6 on exercise B.

7.2.3.3 Recommendation rating precision

It could be misleading to present recommendations to students and educators with much precision. On the one hand, error is inherent in any rating calculated from estimates such as those contained within the student model, and thus recommendation ratings are inherently imprecise. On the other hand, however, much of this error exists not within the calculation but in the way that evidence is interpreted by individual learning activities and fed into the student model. Since students are already able to inspect the raw events and fact estimates from which a recommendation is calculated, it may be inappropriate to hide the details of recommendations based on these.

For the purposes of this prototype, ratings are presented at two significant figures. Further simplification might be achieved by casting ratings into buckets, labeled "highly recommended", "not recommended", and so forth. User input would be required to determine the trade-offs involved in this decision.

7.2.3.4 Handling repeats

It is unclear whether a student who has just completed a learning activity should have that same activity recommended to them again immediately. This situation could easily occur if a particular activity's effect on the student model is not large enough to significantly change recommendation ratings, as might be the case if they perform approximately the same on a quiz that they have previously taken, or if the activity concerned is a static activity, such as a reading.

A variety of modifications are possible, including the incorporation of a timing threshold in recommendation ratings and rules expressed on exercises to require that they not be made available again until some time has passed or some follow-up activity is completed.

7.3 Anatomy Learning System

The Anatomy Learning System is an ontology-driven educational system formed from the components described throughout this dissertation. As most of its functionality inheres within these components and has thus already been described in the preceding chapters, this section focuses primarily on the overall user interface for educators and students, as well as providing details on how all the parts fit together.

The Anatomy Learning System supports two types of users: educators and students. Educators are administrative users, responsible for defining curricula, configuring learning activities and managing students. Students, the system's primary users, pursue learning by completing exercises based on recommendations from the system and, if so desired, inspecting their own progress using the student model and associated event log. Users of both types interact with the system primarily through special front-end pages tailored to their needs.

A third group of users, not discussed here, are the developers and content creators who create learning activity templates, write queries, and manage both assets and style sheets. These users interact with the system using the component specific administrative user interfaces described in previous chapters.

7.3.1 Usage

7.3.1.1 Educators

Educators work from a console page that provides them with links to pages where they can:

- Define curricula for use in creating student models and configuring activity templates.
- Create scenes using the Scene Builder, Anatomy Explorer, and Data Visualizer, for use in configuring activity templates.
- Configure activity templates to create activity instances.
- Register activity instances with curricula to create exercises.
- Register activity instances to create exercises.
- Enroll students.

Tasks can be performed in any order, provided all prerequisite objects and configurations exist. It is assumed that educators will not define activity templates themselves, as this process is somewhat technical. Instead, they will select from activity templates that have already been defined and configure these to create instances.

In addition to providing links to the necessary administrative pages, an educator's console page lists all curricula and the students enrolled in them, with summary information from each student's model alongside links to detailed information.

An example educator console page is shown in Figure 37.

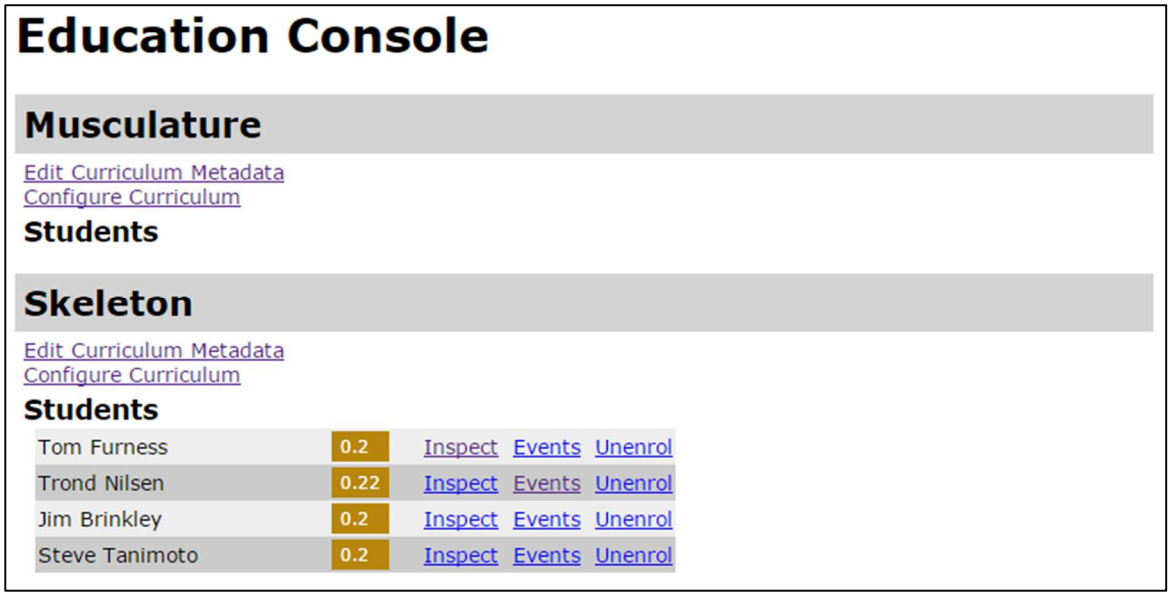


Figure 37 – Educator Console

7.3.1.2 Students

Students interact with the Anatomy Learning System using a frontend console page, divided into sections according to the curricula they are enrolled in. Each curriculum is listed with its name, description, and a summary of student progress provided along with a link to the student model for more detailed inspection. Underneath each curriculum are listed the exercises available for it, in order of recommendation, with exact ratings highlighted in red, yellow, or green, according to whether that exercise is urgent, recommended, or should be performed for revision only.

Using this console page, students may:

- Review the curricula they are enrolled in.
- Evaluate their progress in a particular curriculum by inspecting the associated student model and event log.
- Select and complete an exercise.

When students select an exercise to work on, the corresponding activity template is loaded in their browser, with all relevant parameters including the student's ID, that of the curriculum they are working on, and a callback URL for the console page, to be loaded when they are done.

An example student console page is shown in Figure 38.

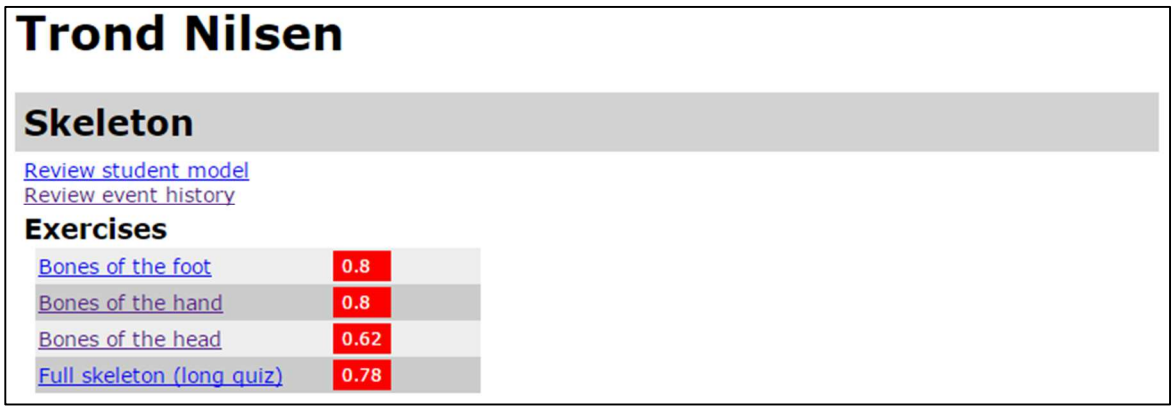


Figure 38 – Student Console

7.3.2 Implementation

The Anatomy Learning System is implemented as a web application written in Java with Spring MVC, running on Tomcat with a PostgreSQL database for storage. It interacts with a number of other components through their web services and Java API. Two user interfaces are provided, one for educators, and the other for students. Most functionality is performed within individual components, and is documented in the appropriate section within one of the preceding chapters.

The Anatomy Learning System interacts directly with the following components:

- The Scene Builder, as a source of content that can be used to configure activity templates to create activity instances.
- The Curriculum Manager, to retrieve curricula for listing on console pages, as a source of content information, and as a hook for registering activity instances to create exercises.
- The Activity Manager, to perform the work of managing activity templates, activity instances, and exercises.
- The Student Model, as a source of information about student progress and as a destination for students wanting to learn more by inspecting their model and event log.

- The Recommendation Engine, as a means of calculating recommendations that determine how exercises should be listed for each curriculum.

Other components interact with the Anatomy Learning System indirectly by providing support to components on the list above or through use by particular learning activities.

7.3.2.1 Further development

Extensions to the various components that make up the Anatomy Learning System have been discussed in previous sections; most of these will have a beneficial effect on the system. Two additional system-level considerations are discussed below.

7.3.2.2 User management

At present, the system does not differentiate between different types of users, despite conceptual differences between administrators, educators, and students, and the availability of a role-base access control system within the Anatomy Engine. This is largely because such differentiation does not further any research goals and, furthermore, is a hindrance to testing and development. It is, however, not acceptable for deployment.

Access to various parts of the system is currently controlled using Spring Security, which allows authorization to be specified using URL patterns. This is adequate to prevent users from accessing administrative pages if they do not possess the requisite permissions, but does not allow finer-grained access control; for example, to control which users may manage particular curricula and so forth. This situation could be improved by extending the RBAC component to support access control lists, as suggested in section 4.2.4.2, but many other components would also require modification. Another option involves the development of specialized plugins for Spring Security, but this has not yet been investigated.

7.3.2.3 Exercises without model updates

Almost all of the discussion about learning activities so far has assumed that an activity is an interactive experience in which users both learn and express knowledge that is used to update the student model. This need not be so. Instead, they may be passive activities

such as readings, offline activities such as group discussions, or games in which player actions cannot easily be interpreted as assertions about anatomy.

In all of these cases, no information is generated to update the student model. This has two implications: firstly, the model becomes less accurate, as presumably the student has learned something as result of the exercise that has not been captured; and secondly, their recommendations will not change, leading the Recommendation Engine to recommend repeating the exercise, as discussed in section 7.2.3.4.

One strategy for addressing this issue is to include within a curriculum activities whose purpose is the teaching of knowledge mixed in with those that seek to test it. Other activities might be included to help students build associations and skills using knowledge in order to improve retention. In this way, the system can be broadened as a platform from just being a tool to support interactive games and learning activities to being a general-purpose learning management system. Such extension of the system would be an ambitious undertaking, and might best be achieved by integrating with an existing learning management framework such as Moodle [251] or Blackboard [252].

7.3.2.4 Dependency between different objects

One drawback of the modular approach that has been taken in developing the software components described throughout this dissertation is that it is difficult for individual components to manage how their resources are used by others. For example, there is presently no way for the Scene Builder to know how scenes in its repository are used in activity templates such as the quiz described below. Consequently, there is no way to assure an educator that the scenes and other elements they might depend on will remain constant unless they develop them all themselves. This poses a major obstacle to the philosophy of an open platform such as the Anatomy Learning System, where users are encouraged to share and share alike their creations.

7.4 Example – Self Generating Quiz

Quizzes are a common form of learning activity used primarily to test knowledge and reinforce it in memory through repetition. Quizzes take many forms, including multi-choice, short answer, and object selection. The quiz presented in this section asks students to identify anatomical structures either by selecting the name of a highlighted structure from a list or by indicating a structure that has been specified by name.

Though its efficacy as an educational tool is modest, the main purpose of this quiz is to illustrate how learning activities can be built and integrated with the Anatomy Learning System, and how the system's various features work in practice. The quiz employs the following elements of the system:

- The 3D scene used by the quiz is loaded from the Scene Builder's scene repository, and may have been created using any of the tools that use it to store content.
- Potential questions in a particular quiz exercise are selected by intersecting the entities shown in the scene with entities contained within the curriculum associated with that exercise. That is, entities must exist within both to be valid for use in questions; the scene so that they can be indicated with highlighting and so that students can select them, and the curriculum so that only relevant questions that can be used to update the student model are asked.
- Actual questions are selected randomly from the pool of potential questions, weighted according to estimates of student knowledge derived from the student model. Questions for which the system estimates a lower degree of student knowledge are asked more frequently than those that they are thought to know well.
- The Assessor is used by the quiz to assess student responses and dispatch evidence to the server so that the student model can be updated.

The quiz exists as a single activity template and has been configured to produce three activity instances, one concerning the bones of the skull, another concerning the muscles

of the face and neck, and a third concerning the structures of the brain. A single curriculum has been defined that incorporates knowledge about these entities, and each of these instances has been registered to create three exercises that are considered by the recommendation engine and presented on the student's overview page.

7.4.1 Gameplay

The student experience of the quiz and learning system proceeds as follows:

- The student navigates to the Anatomy Learning System, logging in with credentials that have been given to them by a course administrator.
- The system directs them to the console page, which shows information relevant to them.
    - If they are enrolled within the curriculum containing the quiz, it will be shown along with the three quiz exercises. If the student has not yet performed any exercises, all three will be listed in red to indicate that they are "highly recommended". Otherwise, recommendations will vary according to what progress they have made.
- The student selects a quiz, which is loaded in their browser. They see a 3D scene showing the relevant anatomy, with a small panel describing the quiz to them. Figure 39 shows a screenshot of the quiz just after it has started. The student clicks a button to begin play.
- The student answers questions one at a time. These questions are generated based on that student's knowledge so that less known entities are asked about more often than known entities. Questions take two forms:
    - The quiz indicates a structure by highlighting it within the scene and the student is asked to identify it by selecting its name from a list.
    - The quiz gives a structure name, and the student is asked to identify that structure by selecting it within the scene and pressing a button.

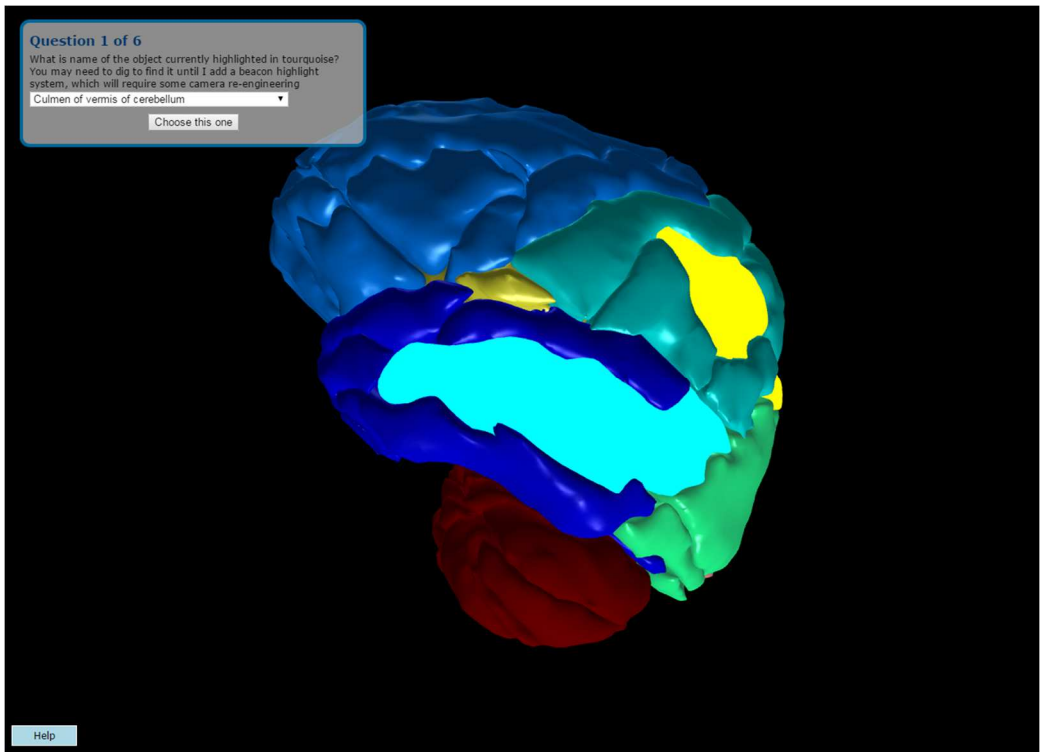


Figure 39 - User interface; Quiz

- While answering questions, the student has access to the full suite of camera controls provided by AVA, as well as the ability to hide structures in order to better see those underneath.
- If a student answers correctly, the quiz congratulates them and then moves on to the next question. If they answer incorrectly, the quiz informs them and offers to show them the correct answer before they proceed to the next question.
- Once all questions have been attempted, the quiz summarizes their performance with a score presented as the number of questions answered correctly out of the number of questions answered.
- Finally, the quiz returns the player to the student console page, where updated progress estimates and recommendations are shown.

Conceptually, this quiz is similar to flash card software such as Mnemosyne [253] and Supermemo [254], as well as the physical flash cards used by many students.

7.4.2 Implementation

The quiz template is implemented in JavaScript using three.js, the CGA, and the Assessor's JavaScript support class. Its architecture is based around a state machine pattern, with states for each step of initialization, for the various stages of each type of question, and for the summary page, making it fairly easy to extend to support new types of question. The quiz takes advantage of services provided by the Scene Builder, Student Model, and Curriculum Manager via their web service interfaces.

The quiz template must be configured with four parameters in order to create an actual quiz exercise that a student can play:

- The number of questions to be asked in each session, defined during activity instance configuration
- A scene ID, used to obtain a scene content descriptor for display within the quiz, defined during activity instance configuration
- A curriculum ID, used in conjunction with the Scene ID to select potential questions, defined during exercise registration
- A student ID, used in conjunction with the curriculum ID to select a student model for updating and actual question selection, defined by which student is invoking the exercise.

Questions are created according to the following procedure:

- A potential question is created from each entity that is present in the scene, is represented in the curriculum, and is associated with a LabelFact.
- Each potential question is assigned a weight w based on student knowledge retrieved from the student model using the formula $w = 1 - p$, where p is the probability value attached to the LabelFact associated with the entity represented by that question.

- Actual questions are selected randomly from the list of potential questions, accounting for the weights just calculated. Once a question is selected, it is cast randomly into one of two types according to whether it asks the student to identify an entity by name or by shape and location.

Answers are assessed using the Assessor's JavaScript support class, then used as evidence to update the student model. All facts are pre-cached as described in section 5.3.1, obviating the need for web service requests to perform assessment, though requests are still required to dispatch updates.

7.4.3 Further development

The quiz could be extended or improved in a variety of ways:

- In complex scenes, some assets may be concealed behind others, making it difficult for students to find them. This is particularly problematic when a question asks the student to name an entity that cannot be seen. Students may dig to find assets, but this is tiresome if they do not know where to look. One solution is to provide some kind of beacon effect that indicates the location of a required asset; for example, by radiating lines outward and through other assets.
- Assets are currently highlighted by assigning to them a bright emissive color, with the precise hue dependent on whether they have been highlighted because they are selected, hovered, or required by a question. Hue-based differentiation, however, imposes difficulties for students who are color blind. One approach to addressing this problem is simply to modify the selection of colors to address the most common forms of color blindness. Unfortunately, however, no scheme can address all forms of color blindness [255], and many schemes may serve one form while making things worse for another. Consequently, alternatives such as texturing or pulsating highlights may be required.
- Students are currently asked to identify entity names from a list, which provides them with scaffolding, which may make that task too easy. One alternative is to

ask them to enter names using the keyboard, but this may be cumbersome. Another approach, used by many flash card tools [253], [254], do not require students to actually enter a name, but instead asks them indicate when they've remembered and then to indicate whether they were correct or not. This ought to be harder, but has a drawback in that it allows dishonest students to lie and thereby inflate the student model's estimates of their knowledge.

- A variety of other types of fact could be supported within the quiz without much work. For example, in a quiz based on bone articulation, a student could be presented with a particular bone and asked to select another bone from a list that articulates with it.
- A more sophisticated quiz might allow educators to explicitly define questions with supporting scenes and information on what correct and incorrect answers imply for that student's knowledge. Such a quiz would require substantially more effort to implement, but would be relatively straightforward using the services provided by the Anatomy Engine. As an example, such a quiz could be used to create questions similar in form to those used in medical board exams.

7.5 Conclusion

This chapter presented the Anatomy Learning System, an educational platform built from the components presented in chapters 4 and 5. Two additional software components were presented to support the system, along with a sample activity template in the form of a self-generating quiz.

7.5.1 Future work

Several possible avenues for future work were presented throughout this chapter, including:

- Improvements to the recommendation algorithm to support exercise efficacy ratings derived from expert or data driven analysis, as discussed in section 7.2.3.1.

- More advanced approaches to recommendation, including threshold constraints to support the needs of mastery learning, as discussed in section 7.2.3.2, and mechanisms to support the consideration of time and repetition, as discussed in section 7.2.3.4
- Support for access control over the many types of object within the system, allowing content authors to control the use of the materials they create and to prevent students from directly modifying curricula and student models, as discussed in section 7.3.2.2 and, previously, in section 4.2.4.2.
- Extensions to the quiz template including support for more fact types or, alternatively, a more advanced quiz that allows educators to create complex questions similar to those used in medical board exams.

7.5.1.1 Additional activities

Aside from improving the functionality of the Anatomy Learning System and its various components, a major area of future development should be the implementation of additional activity templates in order to expand the capabilities of the platform.

Of the concepts presented in chapter 6, several could be implemented relatively easily using the components already available (FMA Walls, Body Assembly, Hidden Object, Happy Parts, and Bone Finds). Others are conceptually more sophisticated and would require additional design work (c.f. La Maison Soylent, Neural Invaders, Frankenstein Wrestling, Bone Trader), while the remainder are conceptually straight forward but could not be developed without the use of a more sophisticated game engine (c.f. Blood Racer, Zombie Hunt).

Chapter 8. Conclusion

In the preceding chapters, I have presented a series of software components and design ideas for how an ontology-driven educational system might be constructed. These contributions span a range of areas of functionality, including 3D visualization tools for the web, content authoring, ontology exploration, tutoring, student modeling, and educational activity management, as well educational play in the form of action, strategy, and puzzle games. These contributions have explored, illustrated, and justified the potential of large-scale general-purpose ontologies as a resource on which educational tools can be built.

My explorations have been structured around three research questions, re-stated here:

A. How can an ontology be used to support the creation of content for learning tools and activities?

B. How can an ontology be used to support tutoring?

C. How can games and learning activities be designed to take advantage of an ontology-driven educational platform?

Each of these questions was addressed separately, in chapters 4 through 6, respectively, with the contributions of each integrated into an overall learning system described in chapter 7.

Chapter 4 presented a content creation framework for the 3D web along with a series of end-user applications that supported different approaches to authoring content as well as offering environments that students might use to work with anatomy in project-based learning. This framework demonstrated that an ontology can be used to support knowledge-based content selection and content labeling, as well as providing standardized and controlled terms for its integration with other sources of information. Furthermore,

one of the applications, the Anatomy Explorer, demonstrated that the ontology itself can be a valuable teaching tool.

Chapter 5 presented an ontology-driven tutoring framework based around the classic trinity model of computer-based tutoring. Three components were described: the first an example of using an ontology as a domain model, the second an example of how it can be used as the basis for a student model, and the third showing one way it can be used to support tutoring.

Chapter 6 presented a range of concepts for teaching human anatomy using an ontology-based learning system, with ideas ranging from classic anatomy reconstruction using models to strategy games that require players to operate directly on anatomical knowledge in order to plan a patch to victory in the game.

Finally, chapter 7 brought these disparate ideas together and presented the Anatomy Learning System, a learning platform built from many of the components presented in previous chapters that supports an educational feedback loop of play, assessment, and recommendation leading to further play, as shown in Figure 40. To demonstrate this platform, a self-generating quiz was presented along with details of its implementation that illustrate how other activities, such as those presented in chapter 6, might be incorporated into the system.

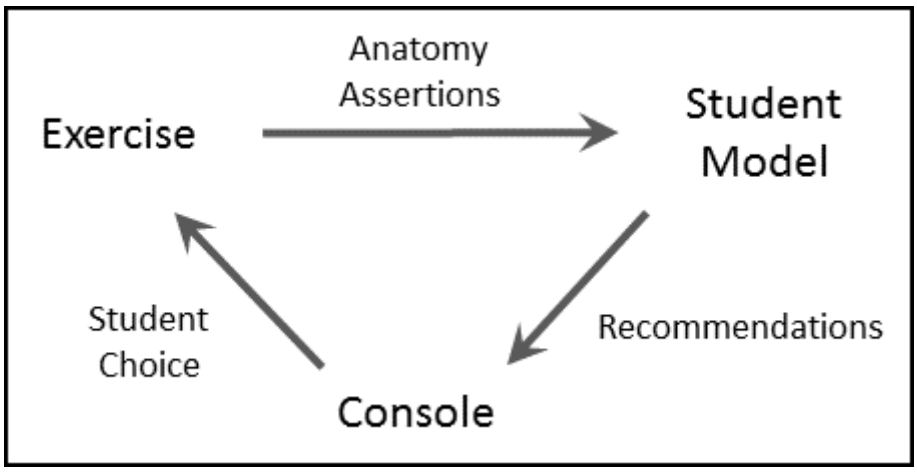


Figure 40 - Educational feedback loop

In all, five completed end-user software applications were presented along with eleven software components, eleven game concepts, and a functional prototype learning platform.

In the remainder of this chapter, I present reflections and personal notes on the experience of working on this project and the overall direction it could take in the future, followed by a summary of the suggestions for future work presented in the conclusions of each of chapters 4 through 7.

8.1 Personal Notes

In all, this project took three years to complete, with work starting in earnest in summer 2012 and proceeding through May 2015. In addition to the design and technical challenges described elsewhere in this dissertation, several challenges of a non-technical nature presented themselves, particularly in the formative stages and later, as I came to pull things together and write them up.

8.1.1 Cross disciplinary research and its challenges

A significant early challenge involved coming to grips with the demands of cross-disciplinary research, in particular reconciling the different conceptions of what constitutes a valid research goal and what methods are appropriate.

I approached this challenge with the broad strategy of working to construct a research project that had validity and appeal within the several disciplines of my advisors. The alternative, choosing one approach and pursuing it to the exclusion of all others, would have amounted to shying away from this problem, and would not have suited my particular disposition.

This approach required me to thread the needle by carefully discussing my goals and work with collaborators and advisors to ensure that I understood their perspectives and was able to take account of their expectations. From their feedback, I chose an exploratory path that allowed me to address three relatively different types of question within a larger

vision, that of ontology-driven education. This approach was, on the whole, successful. In retrospect, this approach has been personally rewarding, as it has exposed me to a variety of methodologies and perspectives.

8.1.2 "How" questions

The questions posed in this project are "how" questions whose answers are inherently complex and cannot be directly addressed using quantitative tools and hypothesis testing. Instead, these questions must be answered through qualitative evaluation, expert review, and reasoned discussion.

The choice to pose such "how" questions was taken because more straight forward "yes / no" hypotheses would simply not have allowed me to approach my overall research goal of exploring the potential of ontology-driven education.

8.1.3 Exploratory research

In exploratory research, one begins with the conjecture that some technology or method will have merit or will at least yield interesting results when applied to some problem area. The results of exploratory research are not definite statements backed up by empirical results, but are rather more like the results of trail-blazing, in that they provide a first map of some possible area of endeavor and an account that may point the way towards productive hypotheses and experiments for future research. This project, then, has explored the potential of ontology-driven education and has opened up a number of possibilities for future research and development.

8.2 On the End Product

This project has resulted in several practical contributions that could be adopted by practitioners and other researchers without much further refinement. The Scene Builder, Anatomy Explorer, Anatomy Learning System, and Quiz are all ready for immediate use by students and educators, while the APIs and libraries of all components are ready and fit for use in developing further applications such as more advanced educational activities.

The following notes address the potential and limitations of these products in the near term.

8.2.1 Model Availability

By far, the biggest limitation to uptake of the Scene Builder and Anatomy Explorer is a lack of high quality 3D models. Though models can be purchased or constructed from volumetric data to some extent, further effort would still be required to segment these models into pieces at a sufficiently fine-grained level of detail for use with the FMA. I have not yet seen any model set that even approaches representation of most of the fine-grained features found in the FMA such as muscle attachment points, tendons, anatomical spaces, bone landmarks, and so forth. Consequently, the probable only option for solving this problem is simply to hire 3D artists to make detailed models, either starting from scratch or using existing models as a starting point.

8.2.2 Ontology Availability

All of the material presented herein is oriented towards teaching human anatomy. There is no a priori reason, however, that this must be so. For example, an ontology-based rules system can be combined with 3D models within the same software components to create educational games for teaching engine design, architecture, or any number of other areas of human endeavor that involve the understanding of complex physical structures.

Human anatomy was chosen for several reasons, the first and foremost being interest and the availability of funding, but also because the availability and high quality of the FMA makes for an excellent test bed. A major limitation to the application of these tools and this approach to other domains, then, would be the lack of such a high quality ontology. Whether the construction of an ontology is worth the effort is unknown, and would likely depend on whether other applications for it could be found and on the extent to which online education is necessary and effective using the methods proposed here.

8.2.3 Extensibility

As discussed throughout chapter 4, there are many ways in which the Anatomy Engine could be improved or extended. Furthermore, many visualization needs within anatomy and medicine more generally could be served by adapting these tools. Understanding this, much of the design philosophy behind the components presented here has been to employ modularity and support extension wherever possible. To what extent this was worthwhile within the context of meeting the research goals and requirements for this dissertation is unknown, but it is hoped that this will prove helpful from a practical standpoint.

8.3 Future Work

Directions for further development and future research have been presented throughout this dissertation in concluding sections within each chapter. The list here summarizes those suggestions:

For the Anatomy Engine:

- Continuing development and completion of the FMA, as discussed in 4.7.2.3.1.
- Improved tools for query authoring, as well as query search and re-use.
- The completion of a number of extensions to these applications, as documented in the subsections labeled "Further Development" throughout this chapter.
- The implementation and evaluation of a number of potential applications based on the framework, as documented throughout section 4.7.2 and in Chapter 6.
- The acquisition of additional models so that the whole body is represented at a high level of detail.
- The development of methods for handling model sets that represent anatomy at multiple granularities.
- Extension of the Engine to other ontologies, both in medicine and other domains.

- Extension of the Engine to support image slices and volumetric 3D data, as discussed in sections 4.7.2.4.2 and 0, respectively.

For the tutoring framework:

- More sophisticated methods of updating fact nodes in the student model, including algorithms that take advantage of time, that apply more sophisticated Bayesian reasoning, and that allow updates of variable strength, as discussed in sections 5.2.3.4 and 5.2.3.5.
- Modifying the student model to account for dependencies between different types of fact, as discussed in section 5.2.3.1.
- Improvements to the way student models are initialized, as discussed in section 5.2.3.2.
- More usable visualizations of the domain and student models, as described in section 5.2.3.6.

For the Anatomy Learning System:

- Improvements to the recommendation algorithm to support exercise efficacy ratings derived from expert or data driven analysis, as discussed in section 7.2.3.1.
- More advanced approaches to recommendation, including threshold constraints to support the needs of mastery learning, as discussed in section 7.2.3.2, and mechanisms to support the consideration of time and repetition, as discussed in section 7.2.3.4
- Support for access control over many types of object within the system, allowing content authors to control the use of the materials they create and to prevent students from directly modifying curricula and student models, as discussed in section 7.3.2.2 and, previously, in section 4.2.4.2.

- Extensions to the quiz template, including support for more fact types or, alternatively, a more advanced quiz that allows educators to create complex questions similar to those used in medical board exams.

The applications, software components, and design concepts presented in this dissertation illustrate both the variety and the scope of the ways that ontologies can be employed as a knowledge resource in educational software for anatomy. Though additional development and polish is of course required, it is my hope that the ideas presented here might either be integrated into existing anatomy courses or used to create standalone educational software as the basis of massively open online courses in anatomy.

Appendix A Validation Study Materials

A.1. Activity 1 – Scene Builder

The Anatomy Engine's Scene Builder is a tool for creating 3D scenes of anatomical structures that can be viewed on the web either as standalone pages, or within existing web resources (such as Wikipedia).

As well as letting you create scenes, the Scene Builder acts as a repository that other, more advanced anatomy visualization tools can build on, as you will see in activities 2 and 3.

The Anatomy Engine and its various components differ from other online tools you may have seen for human anatomy in that it is intelligent. By building on the Foundational Model of Anatomy, a knowledge base that understands how the human body fits together and provides standard terms for use in research and clinical data, the Anatomy Engine allows the creation of sophisticated applications for education, scientific visualization, and research that are not otherwise possible.

In this first activity, you will learn about the basic features of the Anatomy Engine by using the Scene Builder to create a scene showing the skull, the muscles of the neck, and part of the brain.

The instructions below cover all required tasks. Feel free to ignore these instructions at any time to repeat a step, try something different, or just explore the interface. You may ask questions or stop the activity at any time.

A.1.1. Preliminary notes

In the instructions, you will see a few terms that you may not be familiar with. They are defined here, for your reference.

- **Asset Set** - an asset set is a collection of 3D models that can be used together in a scene. You will be using the "Body Parts 3D", a collection of 1000 models derived from MRI data and kindly provided by a group at the University of Tokyo.
- **Stylesheet** – a stylesheet is a list of instructions telling the system how to color different anatomical structures. You will be using the "Default Styles" stylesheet, which colors structures similarly to most anatomical diagrams you will be familiar with.
- **Query** – a request, with parameters, for a list of anatomical structures. One request, for example, might be for all structures that articulate with the left femur, while another might be for the chain of nerves responsible for innervating the right biceps brachii. You will use several pre-authored queries, but more advanced users might write their own. Queries may simply concern anatomy, or they may tie into other data sources; for example, to select structures based on clinical or research data.

A.1.2. Instructions

- Start the Anatomy Engine
    - Go to http://ethmoid2.biostr.washington.edu:8080/SIG-SceneGen
    - Log in with bootstrap / bootstrap
    - You should see a white web page with a number of black buttons along the top of the screen. This is the main console of the Anatomy Engine.
    - *This first step may already have been done for you.*

- Create a new scene
    - Choose "Scene Builder", 5th from the left in the line of buttons at the top of the screen. Gray buttons will appear.
    - Choose "Create new Scene" from the gray buttons.

- Enter "Task 1 – Head and Neck – (your name)" in the box labeled "Name".
  - Make sure you include your name so that you can tell which scenes are yours.
- Click "Submit".

- Open the scene for editing
  - Choose "List Scenes" from the gray buttons at the top of the screen.
  - A list of already created scenes will appear to the right of the screen.
  - Find the scene you just created in the list.
  - Click the link next to it labeled "Build".
  - A new browser tab will open. Move to this tab and examine the user interface presented.
    - The bulk of the screen is black. This is the background on which anatomical structures will appear as you work.
    - In the lower left are two pale blue buttons, the **Help Button** labeled "Help", and the **Link Button** labeled "Link to this Scene".
    - In the top right hand corner is the **Builder Panel**.
  - Click on the **Help Button**.
    - A dialog will appear listing the controls you can use to interact with the Scene Builder. You don't need to remember these now, but you may find it useful to consult this help dialog later.
    - Press the "?" key to close the Help box

- Edit your scene's name
    - Click the button labeled "Edit" at the top of the **Builder Panel**.
    - Change the name of your scene to "Activity 1 – Head and Neck – (your name)".
    - Click "Save Changes".
        - Observe that the name of your scene has updated at the top of the **Builder Panel**.

- Create a scene fragment called "Skull" using a query.
    - Press the "Add Query Fragment" button at the top of the **Builder Panel**.
    - In the dialog box that pops up, configure the query that will select which anatomical structures will be added to your scene. Choose the following values:
        - Name: "Skull"
        - Asset Set: "Body Parts 3D"
        - Stylesheet: "Default Styles"
        - Query: "Members of a set (FMA Name)"
        - FMA Name "Skeleton of head"
            - Note that as you type a list of allowed structures will appear.
    - Click "Save Changes".
        - In a few seconds, a skull should appear in the screen, and your new scene fragment will be described in the **Builder Panel**.
    - Examine the user interface describing your new scene fragment

        - Note that structures listed in green are shown in the scene and have a button labeled "Hide" next to them.
        - Note the message at the bottom of the fragment stating "Some entities in this fragment do not have models". This indicates that the query you have just run returned entities for which 3D models are not available.
    - Click the link "Hide entities" in the top right of the list of structures.
        - This hides the list of structures, and is useful when working with large scenes.
        - Click "Show entities" to show the list of structures again.
    - Play with the "Hide" and "Show" buttons next to structure names.
        - Note that they change whether certain bones are shown in the scene.
        - Note that when an entity is not shown in the scene it is listed in red.
    - Click on the missing model message to see the list of entities that cannot be shown in the scene.
        - This list contains small bones: (the incus, malleus, and stapes), and a "group" entity for which no model exists: the "Skeleton of head".
- Create a scene fragment called "Face Muscles" by selecting structures from a list.
    - Press the "Add List Fragment" button at the top of the **Builder Panel**.
    - In the dialog box that pops up, choose the following parameters for your new fragment:
        - Name: "Neck muscles"
        - Stylesheet: "Default Styles"
        - Asset Set: "Body Parts 3D"

- From the list that pops up, check the box next to the following structures:
    - Anterior Belly of Left Digastric
    - Left Geniohyoid
    - Left Mylohyoid
    - Left Omohyoid
    - Left Scalenus Anterior
    - Left Scalenus Medius
    - Left Scalenus Posterior
    - Left Sternocleidomastoid
    - Left Sternohyoid
    - Left Sternothyroid
    - Left Stylohyoid
    - Left Thyrohyoid
    - Posterior Belly of Left Digastric

- Click "Save Changes" and wait for your scene fragment to load.
- Examine the user interface for your new fragment.
    - It is similar to that for the query fragment you created earlier, except there are no entities with missing models. This is because you can only select entities for which 3D models are available when creating your fragment

- Now that you've created a simple scene, explore it a little.
    - Rotate the scene.
        - Hold the right mouse button and drag to the left or right to rotate around an axis running vertically through the scene.
        - Hold the right mouse button and drag up or down to rotate around an axis running horizontally through the scene.
        - Press the left or right arrows on the keyboard to rotate around an axis running straight out from the screen.
    - Zoom in or out
        - Use the mouse wheel to zoom in or out.
    - Select a structure. There are two ways to do this:
        - Either
            - Move the mouse over any structure, and left click.
            - Click on the name of a structure in **Builder Panel**.
        - Once selected, a structure will be highlighted in the list in the **Builder Panel** and a panel will appear at the top of the screen showing its name.
            - Note the colored boxes and slide bars under the structure's name. Ignore these for now.
        - To de-select a structure, either click on a new structure, or click on an empty part of the scene.
    - Change your point of focus
        - Hold "Shift" and press the left mouse button somewhere on the scene to focus the camera on that point. This allows you to zoom in on particular features.

- Keep playing with these controls until you feel you have mastered them.

- Modify the appearance of a structure.
  - Select any structure so that its name appears in the panel at the top of the screen.
  - Examine the color controls.
    - You do not need to understand the lighting model in any depth, but briefly:
      - The scene has a single light source that you cannot directly see but whose light you can see reflected on objects in the scene
      - Ambient and Emissive control a structure's base color.
      - Diffuse, specular, and shininess control the way the surface reflects light. Diffuse controls the color of the broad, "matte" reflection of the surface, while specular controls the color of the glossy highlights. Shininess controls how "sharp" those highlights are.
      - Alpha controls how transparent a structure is.
  - Choose a structure, and change its color.
    - Use the following color settings (the exact shade doesn't matter):
      - Ambient – light blue
      - Diffuse – light blue
      - Emissive – black
      - Specular – white

        - By adjusting the appearance of a structure, you can highlight it for reference or to get a viewer's attention.
        - On PC, you may need to click "OK" in the color selection box to apply the color you have selected.
        - On a Mac, you may need to close the color selection box like you close any other window before the color is applied.

- Save a viewpoint
    - Rotate, zoom, and re-focus the scene to give a good view on the hyoid bone
    - Click the "Save Viewpoint" button at the top of the **Builder Panel**.
        - A saved viewpoint is used by default when a scene is shown to a user.
        - A saved viewpoint can also be returned to at any time by pressing "R".
            - Verify this by rotating the scene in any arbitrary direction to move it away from the hyoid bone, then pressing "R".

- Complete the scene
    - Add the teeth using a query fragment.
        - Hint – remember to use the correct asset set and stylesheet
        - Hint – the FMA groups the teeth into the set "Secondary Dentition". Use a query that looks for the members of a set.
    - Add the following additional structures using a list fragment
        - Set of Nasal Cartilages
        - Thyroid Cartilage

- Add part of the brain using a query fragment.
  - This time, use the brain parts stylesheet
  - Hint – remember to use the body parts 3D asset set
  - Hint - The easiest way to add a good chunk of the brain is by adding it as "regional parts" of the "Left cerebral hemisphere".
  - This query might take a little longer to run, as the brain is much more complex than the bones of the skull.
- Hide part of the skull so that the brain inside can be seen.
- Highlight the left digastric muscle by changing its emissive color to pink. Note that the digastric muscle has two parts!
- Choose a good viewpoint for the scene, and save it.

- Publish the scene.
  - Click the button labeled "Link to this scene" in the lower left corner of the screen.
    - A panel appears with instructions on how you can use the scene you have created.
    - To hide this panel, click the button again.
  - Copy the URL from the first of the links shown, under "On our server"
  - Open a new browser tab and visit http://www.meme-hazard.org/sig/demo_wikipedia
  - Paste the URL you just copied into the input field label "Paste URL here", and press "Apply"
  - The page will update, and the scene you have just authored will appear in line with the Wikipedia content.

The page you see is not the real Wikipedia. Instead, it is a mockup page built from a Wikipedia page. Content created using the Anatomy Engine can be incorporated into any web page that you have authority to edit, such as your own educational materials.

A.2. Activity 2 – Data Visualizer

The Data Visualizer lets you create data visualizations based on scenes that have been stored in the Scene Builder's scene repository. To be visualized in this way, data sets must either be annotated with the Foundational Model of Anatomy (FMA) or with some other ontology or controlled vocabulary that can be linked to the FMA.

In this demonstration, you will learn how to use the Data Visualizer to create visualizations, how to create your own stylesheet for use in visualizations, and how to save visualizations for use in other web resources.

The instructions below cover all required tasks. Feel free to ignore these instructions at any time to repeat a step, try something different, or just explore the interface. You may ask questions or stop the activity at any time.

A.2.1. Preliminary notes

Visualizations are created by selecting a combination of a scene, a data query (with optional parameters), and a stylesheet:

- The structures shown in a visualization are defined by the contents of a **scene** from the scene repository. Viewpoint and structures are preserved from the original scene, but the appearance of structures is overridden.
- A data **query**, similar to those used in the Scene Builder, provides data values for each structure in the scene for which data exists. This data can be viewed directly by selecting structures in the visualization, or used to select a style for that structure.
- The appearance of each structure is determined by a **stylesheet** based on data loaded from a query. Stylesheets consist of a series of rules that are matched against a structure's data to select a **style** that defines the appearance of that structure.

A.2.2. Instructions

- Start the Anatomy Engine
    - Go to http://ethmoid2.biostr.washington.edu:8080/SIG-SceneGen
    - Log in with bootstrap / bootstrap
    - You should see a white web page with a number of black buttons along the top of the screen. This is the main console of the Anatomy Engine.
    - *This first step may already have been done for you.*

- Open the Visualizer
    - Choose "Applications", 2nd from the left in the lower line of buttons at the top of the screen.
    - Two gray buttons will appear. Click the one labeled "Data Visualizer".
    - A new browser tab will open. Move to this tab, and examine the user interface.
        - You will see a black background space. As with the Scene Builder, anatomical structures will appear here when loaded.
        - In the lower left are two buttons. The **Help Button**, as previously shown, gives instructions you may wish to consult. The **Save Scene Button** allows you to save the contents of the Visualizer as a scene.
            - Clicking on the both buttons again will hide their associated panel.
        - In the upper left is the **Legend Panel**. It is currently empty, but, when a scene is loaded, it will list the meaning of the different styles used in the scene.
        - To the right is the **Configuration Panel**. It is divided into three sub-panels, the **Scene Sub-panel**, the **Data Sub-panel**, and the

Stylesheet Sub-panel. These sub-panels control your selection of the three elements of a visualization.

- Each sub-panel contains a field showing the current selection, a drop down box, and a button.
- The **Data Sub-panel** and **Stylesheet Sub-panel** also contain a grey description panel where information about the current selected query or stylesheet is shown.

- Below the configuration panel is the **Info Panel**. It currently contains the text "Nothing Selected". When data has been loaded, values for the selected structure will be shown here.

- Load a scene
  - In the **Scene Sub-panel**, use the drop down box to select your Head and Neck scene from Activity 1. It should be called "*Activity 1 – Head and Neck – (your name)*"
  - Click "Load Scene"
    - Once your scene has loaded, it will appear on the black background. All of its structures will be grey, as styles will be assigned to them later.
    - Your scene will be listed in the current scene field.

- Apply a basic stylesheet
  - In the **Stylesheet Sub-panel**, examine the stylesheets listed in the drop down box.
    - You may recognize the "Default Styles", "Brain Parts", and "Bone Types" stylesheets from the Scene Builder. These three stylesheets can be used outside the Data Visualizer because they do not rely on

structures having data values. They can also be applied within the Visualizer before data is loaded.

- Select "Default Styles" from the drop down box.
  - An explanation of this stylesheet will appear in the description panel below the drop down box.
- Click "Apply Stylesheet".
  - The structures in your scene will be styled similarly to a traditional anatomical diagram. That is, bones will be bone-colored, muscles crimson, and so forth.
  - "Default Styles" will appear in the current stylesheet field.
  - Note that the Legend Panel will update to show the meaning of each of the styles used.
- Apply the "Bone types" stylesheet.
  - Bones will be styled according to type (long, short, irregular, cartilage etc.).
- Apply the "Brain parts" stylesheet
  - Brain structures will be styled according to the lobe or region they are a part of.

- Create a simple data visualization.
  - Load the scene called "Brain".
    - This scene shows the brain in more detail, and is thus better suited for visualizing data.

- Assuming you still have the "Brain Parts" stylesheet applied, the newly loaded structures should automatically be styled using that stylesheet.

- Load some data.
  - In the **Data Sub-panel**, use the drop-down box to select the query "Query 188 (AAL Cached)".
    - This query returns data summarizing the results of a series of studies in which activation sites in different regions of the brain were recorded using MRI for both schizophrenic and healthy patients while performing a simple motor tasks
  - Click "Load Data"
    - The "Data loaded" will update to show the newly loaded query.
    - An explanation of the query will appear in the description panel below the drop down box.
- View data for a structure
  - Select any brain structure by clicking on it
  - Examine the **Info Panel**.
    - It now contains data values for your structure.
      - The value `healthy` counts how many healthy patients in the study had activations in this region of the brain.
      - The value `schizo` counts how many schizophrenic patients in the study had activations in this region of the brain.
- Apply the "Brain – 188 – Healthy patients" stylesheet.

        - Each structure will be styled according to the data value `healthy`.
        - Click on several structures and verify that data values correspond with the meaning of each style listed in the **Legend Panel**.
    - Apply the "Brain – 188 – Schizophrenic patients" stylesheet.
        - Review structure styles and data values as before.

- Look at a basic stylesheet
    - Return to the browser tab labeled "Anatomy Engine – Admin".
    - In the navigation, choose the black button 4th from the top left, labeled "Styles".
    - In the navigation, click the grey button labeled "List Stylesheets"
        - A list of stylesheets will appear to the right of the screen.
    - Click on the stylesheet named "Brain Parts".
        - A configuration panel will appear, consisting of a series of rows of colored boxes under the title "Configure Stylesheet – Brain Parts".
    - Examine the configuration panel.
        - Each row corresponds to a single style.
        - Styles are defined using the same controls as in the Scene Builder.
        - Text in the field labeled "Rule" is used to determine how styles are applied
            - This stylesheet uses simple tags corresponding to FMA structure types.
                - For example, a rule with the tag "Gyrus of frontal lobe" will be applied to any structure that the FMA indicates is a ***Gyrus of frontal lobe***.

- Look at an expression-based stylesheet
    - Click on the stylesheet called "Brain – 188 – Healthy patients".
    - Examine the configuration panel.
        - In this stylesheet, each rule is a simple expression that is applied to a structure's data values.
            - For example, the rule "healthy > 1" matches all structures with a value for `healthy` of greater than 1.
        - Note that more than one rule can apply to a given structure. When a stylesheet is applied, the first matching rule, working from the top, is selected.
    - Change the stylesheet so that structures with a value for `healthy` of greater than 32 turn blue.
        - Change the color swatches under Diffuse and Emissive for the rules for **healthy > 64** and **healthy > 32** to blue.
    - Return to the Data Visualizer, and check the results of your change. You will need to re-apply the stylesheet for your change to have effect.

- Create a parameterized data visualization
    - Return to the Data Visualizer if you haven't already.
    - Apply the stylesheet "Brain - 137 - VBM value"
    - In the **Data Sub-panel**, use the drop down box to select the query "Data - Query 137 (by URSI)", but do not click "Load Data".
        - The description panel will update to show an explanation of the query along with a list of URSI values. These will be used below.

        - A box will appear beneath the description panel labeled "Parameters". It contains a field labeled "URSI".
            - Parameters allow the use of queries that require additional information to execute. For example, a query might require a patient ID to load data concerning that individual.
    - Enter "M02116465" into the URSI parameter field, then click "Load Data".
        - The brain will be styled to show values from this new data set.
    - Experiment with loading data using other URSI values from the query description.

- Save your scene
    - Click on the **Save Scene Button** in the lower left of the screen.
        - A panel will appear with options for saving a scene.
    - Click on the button labeled "Save as new scene"
        - Once the scene has saved, the name of your newly saved scene will appear. Note that its name is "Data Visualizer", followed by a timestamp. This will make it possible to find later.
    - Switch to the browser tab containing the Anatomy Engine main console.
    - List scenes as you did while working on the Scene Builder.
        - Click "Scene Builder", the "List Scenes"
    - Find your newly created scene.
        - It will be the last of the scenes with names beginning "Data Visualizer"
    - Open your scene in the Scene Builder, and change its name to "(your name)'s data visualization".

- Publish your scene
    - Return to the Anatomy Engine Main console and refresh the list of scenes by clicking on the grey box labeled "List Scenes".
    - Find your scene in the list.
        - Right click on the link next to your scene labeled "View". Choose "Copy Link Address"
    - Open a new browser tab and visit http://www.meme-hazard.org/sig/demo_wikipedia
    - Paste the URL you just copied into the input field label "Paste URL here", and press "Apply"
    - The page will update, and the scene you have just authored will appear in line with the Wikipedia content.
    - Return to the Anatomy Engine Main console
    - In the list of scenes, find the link labeled "Zip". If you click on this link, a zip file will download. It contains the files necessary to deploy your scene on your own server if you do not want to rely on ours.

A.3. Activity 3 – Anatomy Explorer

While the Scene Builder uses the Foundational Model of Anatomy to help users create content and the Data Visualizer relies on it to organize and interpret data sets, the Anatomy Explorer lets you explore the FMA itself.

You can use the explorer to:

1. **Find information** – the explorer is easier to browse and navigate than, say, a text book, and presents information in a consistent format that is easy to peruse.
2. **Learn** – the explorer supports learning activities such as research and problem solving that force students to actively process anatomical information, thereby improving retention and integration with existing knowledge.
3. **Create content** – the explorer offers an intuitive and interactive means of constructing content. Scenes can be constructed directly in the explorer or created in the Scene Builder based on information from the Explorer.
4. **Find and fix problems** – the explorer can help ontology authors, content creators, and query writers debug their work either by correcting their understanding of how structures are related or by finding errors and omissions in the FMA itself.
5. **Understand how the FMA works** – the explorer is a reference tool that can help anyone who wants to work with the FMA in a technical capacity understand its structure and function.

In this activity, you will learn how to use the Anatomy Explorer and then use it to answer questions about various anatomical structures.

The instructions below cover all required tasks. Feel free to ignore these instructions at any time to repeat a step, try something different, or just explore the interface. You may ask questions or stop the activity at any time.

A.3.1. Preliminary notes

In the instructions, you will see a few terms that you may not be familiar with. They are defined here, for your reference.

- **Entity** – Every named structure, space, region, and other part of the body is modeled within the Foundational Model of Anatomy as an entity. They are the nouns that make up the knowledge represented in the FMA. Entities are listed in blue; for example ***Left temporal bone***, ***Biceps brachii***, and ***Heart***.
- **Relation** – Entities in the FMA are connected to one another by relations. They are the verbs of the FMA. For example, the femur *articulates with* the hip bone, the left carotid *branches from* the arch of the aorta, and so forth. Relations are assertions about entities. Relations are listed in red; for example **Has regional part**, **Articulates with**, and **Member of**.
- **Relation Type** – Each relation has a type. The above examples have the relation types *articulates with* and *branches from*, respectively. The Anatomy Explorer groups relations by type.

A.3.2. Instructions

- Start the Anatomy Engine
  - Go to http://ethmoid2.biostr.washington.edu:8080/SIG-SceneGen
  - Log in with bootstrap / bootstrap
  - You should see a white web page with a number of black buttons along the top of the screen. This is the main console of the Anatomy Engine.
  - *This first step may already have been done for you. If you still have the anatomy engine open from a previous activity, start from there.*

- Open the Explorer
  - Choose "Applications", 2nd from the left in the lower line of buttons at the top of the screen.

- Two gray buttons will appear. Click the one labeled "Anatomy Explorer".
- In the form that appears, choose the following values:
  - **Asset Set:** Body Parts 3D
  - **Starting Asset:** Diaphragm
  - **Stylesheet:** Default styles
- Click "OK".
- A new browser tab will open. Move to this tab and examine the user interface.
  - You will see the diaphragm, a large muscle, shown in red in the center of the screen.
  - In the lower left are two buttons. The **Help Button**, as previously shown, gives instructions you may wish to consult. The **Load / Save Button** allows you to save the contents of the Explorer as a scene or, alternatively, load an existing scene to explore from. The **Clear Button** removes all structures from the scene.
  - In the upper right are two panels.
    - The upper panel, titled "Change Exploration", is the **Search Panel**. It lets you search for anatomical structures about which to look up information.
    - The lower panel, currently titled "No current exploration", is the **Exploration Panel**. It will show information about anatomical structures as you explore.

- Explore the diaphragm.
  - Click on the diaphragm.

        - It will become highlighted in yellow.

    - The **Exploration Panel** will fill with information about the ***Diaphragm***.

    - Find the title "Diaphragm (ID: 13295)" at the top of the panel.

        - This indicates which entity information is provided for. It is referred to as the "currently explored entity".

    - Observe that the remainder of the **Exploration Panel** contains a series of sub-panels, each with its own title and containing pairs of boxes connected by arrows.

        - Sub-panels show information about the relations involving the currently explored entity, in this case, the ***Diaphragm***, and are discussed more below.

        - An example sub-panel is shown in Figure 41 - Example relation type sub-panel.

    - Find the scroll-bar to the right of the **Exploration Panel.**

        - Use the scrolling function of your mouse wheel or track pad to scroll the **Exploration Panel** up and down to view information that flows off the bottom of the screen.

- Toggle the visibility of the diaphragm

    - Find the blue box labeled "In Scene".

        - This indicates whether the currently explored entity is shown as a 3D model.

    - Click on the neighboring button, labeled "Remove from Scene".

        - The diaphragm will disappear.

        - The blue box will become red and say "Not in Scene".

- The button will change to say "Add to Scene"

o Click on the button labeled "Add to Scene".

- The diaphragm will reappear.
- The red box will become blue once more and say "In Scene".
- The button will change back to say "Remove from Scene".

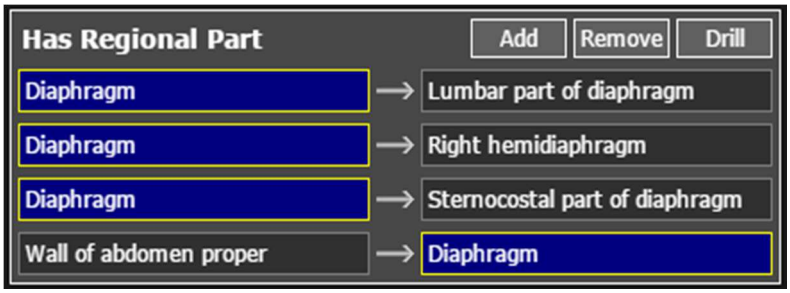


Figure 41 - Example relation type sub-panel

- Examine a sub-panel

o In your browser, find the sub-panel labeled **Has regional part**, as shown in Figure 41 - Example relation type sub-panel.

- You may need to scroll the panel downwards to find it.

o Look at the first pair of boxes, linked by an arrow.

- A pair of boxes represents a relation between two entities.
- The first pair of boxes shows that the ***Diaphragm*** **Has regional part** ***Lumbar part of diaphragm***.
  - Note that the arrow represents a relation of a particular type, as specified by the title of the sub-panel.

o Look at the individual boxes

- Each box represents a single entity, and contains the name of that entity.
- The color of a box has meaning:
  - **Blue** – this entity is present in the scene.

- **Red** – a model exists for this entity, and it can be added to the scene
- **Grey** – no model exists for this entity. It can be explored "virtually" in the **Exploration Panel**, but cannot be added to the scene.

- Notice that boxes for the currently explored entity are bordered in yellow.

- Find the relation ***Wall of abdomen proper*** **Has regional part** ***Diaphragm***
  - The ordering of boxes is important. This relation indicates that that the ***Diaphragm*** is a regional part of the ***Wall of Abdomen Proper***, not the reverse.
- Find the sub-panel labeled **Regional part of**.
  - Observe that this sub-panel contains the same relations as **Has regional part**, but backwards.
  - Several other types of relation also possess an inverse form.

- Navigate to a new entity
  - Find the sub-panel for the relation **Is Directly inferior to**.
    - This relation describes a spatial relationships between structures.
  - Click on the box corresponding to the ***Liver***.
    - The **Exploration Panel** will reload and show information concerning the liver.
  - Make the ***Liver*** visible.
    - Hint – use the "Add to Scene" button, mentioned above.
  - From the ***Liver***, navigate to the ***Common bile duct***.

        - Hint – the ***Liver*** is directly superior to the ***Common bile duct***.
        - Hint – the ***Common bile duct*** is in a grey box, as no 3D model is available for it.

- Explore a new entity using the **Search Panel**.
    - In the **Search Panel** (at the top right of the screen), type "eighth th", and wait briefly.
        - A list of structures will appear in a drop down from the input box.
        - Select the ***Eighth thoracic vertebra*** from the list, and click the "Explore" button.
    - Add the ***Eighth thoracic vertebra*** to the scene.
        - Hint – use the "Add to Scene" button.
    - Rotate the scene so that you can see the ***Eighth thoracic vertebra***.

- Add a group of entities to the scene.
    - Using the **Exploration Panel**, find the relation indicating that ***Eighth thoracic vertebra*** is a member of the ***Set of thoracic vertebra***.
    - Explore the ***Set of thoracic vertebra***.
    - Find the sub-panel labeled for the relation **Has member**.
    - Click the button labeled "Add" in this sub-panel.
        - Clicking "Add" in a sub-panel will add every entity in that sub-panel for which a model is available. That is, all entities labeled in red within the sub-panel will be added – those in blue boxes are already in the scene, and those in grey boxes cannot be added as no model is available for them.

- Click the button labeled "Remove" in this sub-panel
  - Clicking "Remove" in a sub-panel will remove every entity in that sub-panel, except for the currently explored entity. That is, all entities labeled in blue within the sub-panel will be removed.
- Add the thoracic vertebrae back to the scene.

- Review the exploration history
  - Click the **History Button** in the lower left hand corner of the screen.
    - The **History Panel** will appear, listing the entities you have already explored.
    - The currently explored entity is highlighted with a green bar.
    - If you are following these instructions precisely, the list should contain ***Set of thoracic vertebra***, ***Eighth thoracic vertebra***, ***Common bile duct***, ***Liver***, and ***Diaphragm***. Notice that these are the entities you have thus far explored, in reverse order.
  - Click on ***Liver*** in the **History Panel**.
    - The **Exploration Panel** will change to show information about the ***Liver***.
  - Press the up button on your keyboard.
    - The highlight bar in the history panel will move up one step. Information about the Common bile duct should now be shown in the **Exploration Panel**.
    - The down button works similarly, instead moving the highlight bar down one step.

        - Both buttons will function in this way regardless of whether the **History Panel** is visible.

    - Close the **History Panel**.
        - Press the 'H' button on your keyboard.
        - Alternatively, click the **Help Button** in the lower left of the screen.

- Highlight entities associated by a particular relation type.
    - Add the constitutional parts of the ***Left side of rib cage*** to the scene.
        - Use the **Search Panel** to explore ***Left side of rib cage***.
        - Find the sub-panel for **Has constitutional part**.
        - Click the "Add" button in this sub-panel
    - Explore the ***Left third rib*** by clicking on it in the scene.
    - Find the sub-panel for **Articulates with**.
    - Click on the title of that sub-panel
        - The sub-panel will turn purple.
        - The ***Left third rib*** in the scene will turn pink
        - The ***Left third costal cartilage*** and ***Third thoracic vertebra*** in the scene will turn purple.

- Drill down into a relation
    - Use the **Search Panel** to explore the ***Musculature of left free upper limb***.
    - In the sub-panel for the relation **Has member**, click the button labeled "Drill".
        - The **Drill Panel** will appear in the center of the screen, containing a list of relations, indented at different levels.
            - Observe that only with no indentation begin with the currently explored entity. Relations at all other levels of indentation begin with the entity that ends the next relation above them at the previous level of indentation.
            - This panel shows the entities connected to the currently explored entity by chains of one or more steps of a particular relation. Levels of indentation make it easier to see how relations carry on from previous relations in a chain.
            - In this case, we see that ***Musculature of left free upper limb*** **Has member** ***Musculature of left arm***, which in turn **Has member** ***Left anconeus***, and so forth.
    - Click the "Add" button in the **Drill Panel**.
        - Progress bars will appear as a series of muscles load into the scene. Click outside the **Drill Panel** to hide it and see these entities.

- Use the explorer to answer some questions
    - What bones does the muscle ***Biceps brachii*** originate and insert from?

- Hint – Use the relations **Has muscle origin** and **Has muscle insertion** to find the exact origin and insertion point of the muscle.
- Hint – once you have found the regions of origin and insertion, use the **Regional part of** relation to go from specific part entities such as ***Radial tuberosity*** and ***Supraglenoid tubercle*** up to the more overall bone entity that these are a part of.
    - You will need to do this several times to get to the actual bone.
- Hint – the biceps brachii has two insertions and two origins on two bones
    - You can ignore the ***Antebrachial fascia*** – it is not part of a bone.

o What muscles are served by nerves originating from the ***C6*** spinal nerve?

- Hint – Try finding ***C6*** by starting with ***Biceps brachii*** and finding it in the sub-panel for relation **Is segmental supply of**.
- Hint – Once at C6, the **Is segmental supply of** and **Is secondary segmental supply of** relations will provide your answer.

- Load and explore an existing scene.

o Click on the **Load / Save Button** (in the lower left of the screen) to bring up the **Load / Save Panel**.

- Note – clicking on the **Load / Save Button** again will hide the associated panel.

o In the first drop box, labeled "Load scene contents into explorer", select your head and neck scene from activity #1, then click "Load scene"

- Your existing scene will be automatically cleared before the new scene is loaded.
- The entities from your scene will load, though custom styles will be overridden by the stylesheet you selected when opening the Explorer.

- Create and save a scene
  - Click on the **Clear Button** (in the lower left of the screen).
  - Create a scene showing the bones and muscles of the left arm.
    - Hint – You loaded these muscles earlier when working with the **Drill Panel**.
      - You may notice that some muscles are not added when you do this. The ***Left biceps brachii***, for example, is not shown.
      - Look at the list in the **Drill Panel**, and observe that several muscles are shown in grey, indicating that a model is not available for them.
      - Explore the ***Left biceps brachii*** by clicking on it in the **Drill Panel** (you will find it near the top of the list). Notice that there are two **Has regional part** relations connecting it to entities for which models do exist, the ***Long head of left biceps brachii*** and the ***Short head of left biceps brachii***.
      - Add these structures, then go back to the **Drill Panel** to look for more examples where this occurs and add those too.
    - Hint – To add bones, try following the same procedure using ***Skeleton of left free upper limb***. You will not need to go looking for parts as you did with the muscles.

- Hint – you do not need to exhaustively create a perfect scene. Stop when you have created a scene that you are happy with.

- When you are done, click the **Load / Save Button** to bring up the **Load / Save Panel**.

- Click the button labeled "Save as new scene"

  - This will save your scene using in the scene repository. As with scenes saved from the Data Visualizer and Scene Builder, you can view and embed scenes created this way in a separate browser window or embedded in an existing web resource.

A.4. Discussion Questions

After participant completed all activities, they participated in a semi-structured interview. The following topics were raised, though discussions were open-ended.

- Is the software functional?
- Does the structure and layout of the software make sense?
- Can you see how you might use a tool like this to generate visualizations for your work?
- Do you think students would find visualizations derived from tools like this useful?
    - Where can tools like this fit into educational activities?
    - Are educational games / structure learning activities useful?
- What do you think about the use of surface models over clinical images such as CTs, MRIs, and 3D voxel volumes derived from either?
    - Do you think students can benefit from scenes constructed with surface models?
    - Are there particular contexts in which scenes constructed with surface models can fit into your existing work?
- Model detail is obviously a challenge. The models in the tool you have just seen are relatively macroscopic, but models that are more detailed are available for purchase.
    - Did the scenes you were able to construct contain adequate detail?
    - What level of detail do you think is necessary?
    - What fine-grained structures would you like to be able to visualize?
- Computer generated 3D scenes are rarely realistic. What elements of realism do you think are important?

    - Shape and location
    - Color
    - Texture
    - Anatomical variation
    - Background pathology (e.g., scar tissue)
    - Layers, compartments, and fascia
    - "Messiness"
- Most anatomy scenes show the body presented in the standard anatomical position.
    - Is this useful?
    - When are other positions useful?

Bibliography

[1] D. Sturges and T. Maurer, "Allied Health Students' Perceptions of Class Difficulty: The Case of Undergraduate Human Anatomy and Physiology Classes," *Internet J. Allied Health Sci. Pract.*, vol. 11, no. 4, 2013.
[2] J. Levey, "Understanding Baccalaureate Nursing Students' Perceptions of Anatomy and Physiology," *HAPS Educator*, vol. Fall 2009, Sep-2009.
[3] W. W. Cottam, "Adequacy of medical school gross anatomy education as perceived by certain postgraduate residency programs and anatomy course directors," *Clin. Anat.*, vol. 12, no. 1, pp. 55–65, Jan. 1999.
[4] M. Grimley, R. Green, T. Nilsen, and D. Thompson, "Computer Game or Traditional Lecture: The Effect of Delivery Mode on Experience Ratings for High and Low Achieving Students," 2010.
[5] A. King, "From Sage on the Stage to Guide on the Side.," *Coll. Teach.*, vol. 41, no. 1, pp. 30–30–35, 1993.
[6] P. J. O'Byrne, A. Patry, and J. A. Carnegie, "The development of interactive online learning tools for the study of Anatomy..," *Med. Teach.*, vol. 30, no. 8, pp. 260–271, 2008.
[7] J. Skidmore, "The case for prosection: Comment on R. L. M. newell's paper," *Clin. Anat.*, vol. 8, no. 2, pp. 128–130, 1995.
[8] D. G. Jones and M. I. Whitaker, "Anatomy's use of unclaimed bodies: reasons against continued dependence on an ethically dubious practice," *Clin. Anat. N. Y. N*, vol. 25, no. 2, pp. 246–254, Mar. 2012.
[9] J. C. McLachlan and D. Patten, "Anatomy teaching: ghosts of the past, present and future," *Med. Educ.*, vol. 40, no. 3, pp. 243–253, Mar. 2006.
[10] L. J. Rizzolo and W. B. Stewart, "Should we continue teaching anatomy by dissection when …?," *Anat. Rec. B. New Anat.*, vol. 289B, no. 6, pp. 215–218, Nov. 2006.
[11] A. Winkelmann, "Anatomical dissection as a teaching method in medical school: a review of the evidence," *Med. Educ.*, vol. 41, no. 1, pp. 15–22, Jan. 2007.
[12] A. Anderson, D. Huttenlocher, J. Kleinberg, and J. Leskovec, "Engaging with Massive Online Courses," in *Proceedings of the 23rd International Conference on World Wide Web*, Republic and Canton of Geneva, Switzerland, 2014, pp. 687–698.
[13] R. Mayer, "Applying the science of learning to medical education.," *Med. Educ.*, vol. 44, no. 6, p. 543, 2010.
[14] K. Salen and E. Zimmerman, *Rules of Play: Game Design Fundamentals*. MIT Press, 2004.
[15] R. Mayer, *The Cambridge handbook of multimedia learning*. Cambridge U.K. ;;New York: Cambridge University Press, 2005.
[16] E. Klopfer, S. Osterweil, and K. Salen, "Moving Learning Games Forward," Massachusetts Institute of Technology, 2009.
[17] L. Uhr, "Teaching machine programs that generate problems as a function of interaction with students," in *Proceedings of the 1969 24th national conference*, New York, NY, USA, 1969, pp. 125–134.
[18] "Roguelike," *Wikipedia*. 13-Jan-2012.
[19] *Strange Adventures in Infinite Space*. Digital Eel, 2003.
[20] "Device Ownership Over Time," *Pew Research Internet Project*. [Online]. Available: http://www.pewinternet.org/data-trend/mobile/device-ownership/.
[21] J. P. Gee, "Deep Learning Properties of Good Digital Games How Far Can They Go?," in *Serious Games: Mechanisms and Effects*, 1st ed., Routledge, 2009, pp. 65–80.
[22] P. Cohen, "Educational Outcomes of Tutoring: A Meta-Analysis of Findings," *Am. Educ. Res. J.*, vol. 19, no. 2, p. 237, 1982.
[23] B. S. Bloom, "The 2 Sigma Problem: The Search for Methods of Group Instruction as Effective as One-to-One Tutoring," *Educ. Res.*, vol. 13, no. 6, pp. 4–16, Jun. 1984.

[24] D. C. Merrill, B. J. Reiser, Shannon K. Merrill, and S. Landes, "Tutoring: Guided Learning by Doing," *Cogn. Instr.*, vol. 13, no. 3, pp. 315–372, Jan. 1995.
[25] D. C. Merrill, B. J. Reiser, M. Ranney, and J. G. Trafton, "Effective Tutoring Techniques: A Comparison of Human Tutors and Intelligent Tutoring Systems," *J. Learn. Sci.*, vol. 2, pp. 277–305, Jul. 1992.
[26] B. Fox, "Cognitive and interactional aspects of correction in tutoring," in *Teaching knowledge and intelligent tutoring*, P. Goodyear, Ed. Norwood, NJ: Ablex, 1991, pp. 149–172.
[27] M. Lepper, L. Aspinwall, D. Mumme, and R. Chabay, "Self-perception and social-perception processes in tutoring: Subtle social control strategies of expert tutors," in *Self-inference processes: The Sixth Ontario Symposium in Social Psychology*, 1990, pp. 217–237.
[28] L. Sarasin, *Learning style perspectives : impact in the classroom*. Madison WI: Atwood Pub., 1999.
[29] T. L. Hein and D. D. Budny, "Teaching to students' learning styles: approaches that work," in *Frontiers in Education Conference, 1999. FIE '99. 29th Annual*, 1999, vol. 2, pp. 12C1/7–12C114 vol.2.
[30] M. T. H. Chi, M. Roy, and R. G. M. Hausmann, "Observing Tutorial Dialogues Collaboratively: Insights About Human Tutoring Effectiveness From Vicarious Learning," *Cogn. Sci.*, vol. 32, no. 2, pp. 301–341, Mar. 2008.
[31] A. Collins and A. L. Stevens, "A Cognitive Theory of Inquiry Teaching," in *Teaching knowledge and intelligent tutoring*, P. Goodyear, Ed. Norwood, NJ: Ablex, 1991, pp. 149–172.
[32] C. Linehan, B. Kirman, S. Lawson, and G. Chan, "Practical, appropriate, empirically-validated guidelines for designing educational games," in *Proceedings of the 2011 annual conference on Human factors in computing systems*, New York, NY, USA, 2011, pp. 1979–1988.
[33] C. Rosse and J. Mejino, "The Foundational Model of Anatomy Ontology," in *Anatomy ontologies for bioinformatics principles and practice*, [London?] : Springer, 2007.
[34] D. Kachlik, V. Baca, I. Bozdechova, P. Cech, and V. Musil, "Anatomical terminology and nomenclature: past, present and highlights," *Surg. Radiol. Anat.*, vol. 30, no. 6, pp. 459–466, Aug. 2008.
[35] Federative Committee on Anatomical Terminology, *Terminologia Anatomica International Anatomical Terminology*, Bilingual edition. Thieme Medical Publishers, 2000.
[36] "Terminologia Anatomica," *Wikipedia, the free encyclopedia*. 05-Jul-2014.
[37] D. B. Lenat, "CYC: A Large-scale Investment in Knowledge Infrastructure," *Commun ACM*, vol. 38, no. 11, pp. 33–38, Nov. 1995.
[38] "Definition - gaming, n.," *OED Online*. Oxford University Press.
[39] R. Caillois, *Man, Play and Games*. University of Illinois Press, 2001.
[40] J. Huizinga, *Homo Ludens: A study of the play element in culture*. Routledge, 1949.
[41] "Lego," *Wikipedia, the free encyclopedia*. 01-Sep-2014.
[42] "SimCity," *Wikipedia, the free encyclopedia*. 29-Aug-2014.
[43] D. Dutton, *The art instinct: beauty, pleasure, & human evolution*. New York: Bloomsbury Press, 2009.
[44] M. Grimley, "Using computer games for instruction: The student experience," *Act. Learn. High. Educ.*, vol. 12, no. 1, pp. 45–56, 2011.
[45] "Definition - anatomy, n.," *OED Online*. Oxford University Press.
[46] A. P. Dhawan, *Medical image analysis*. 2011.
[47] A. T. Raftery, "Anatomy teaching in the UK," *Surg. - Oxf. Int. Ed.*, vol. 25, no. 1, pp. 1–2, Jan. 2007.
[48] C. Rosse and J. L. V. Mejino, "A reference ontology for biomedical informatics: the Foundational Model of Anatomy.," *J. Biomed. Inform.*, vol. 36, no. 6, pp. 478–500, Dec. 2003.
[49] J. F. Brinkley and C. Rosse, "Requirements for an on-line knowledge-based anatomy information system.," *Proc. AMIA Symp.*, pp. 892–896, 1998.

[50] K. Sugand, P. Abrahams, and A. Khurana, "The anatomy of anatomy: A review for its modernization," *Anat. Sci. Educ.*, vol. 3, no. 2, pp. 83–93, Mar. 2010.
[51] J. Older, "Anatomy: A must for teaching the next generation," *The Surgeon*, vol. 2, no. 2, pp. 79–90, Apr. 2004.
[52] "Learning Goals for Students in Anatomy and Physiology," *Human Anatomy & Physiology Society*. [Online]. Available: http://www.hapsweb.org/?page=LearningGoals. [Accessed: 22-Jul-2014].
[53] S. C. Marks, S. L. Bertman, and J. C. Penney, "Human anatomy: A foundation for education about death and dying in medicine," *Clin. Anat.*, vol. 10, no. 2, pp. 118–122, Jan. 1997.
[54] H.-W. Korf, H. Wicht, R. L. Snipes, J.-P. Timmermans, F. Paulsen, G. Rune, and E. Baumgart-Vogt, "The dissection course - necessary and indispensable for teaching anatomy to medical students," *Ann. Anat. Anat. Anz. Off. Organ Anat. Ges.*, vol. 190, no. 1, pp. 16–22, 2008.
[55] L. T. Detwiler, J. L. V. Mejino, and J. F. Brinkley, "Converting the Foundational Model of Anatomy to OWL2," in *Proceedings, Fall Symposium of the American Medical Informatics Association*, San Francisco, 2015.
[56] J. F. Brinkley and L. T. Detwiler, "A Query Integrator and Manager for the Query Web," *J. Biomed. Inform.*, vol. 45, no. 5, Oct. 2012.
[57] World Wide Web Consortium, "XQuery 1.0: An XML Query Language," *XQuery 1.0: An XML Query Language*, 14-Dec-2010. [Online]. Available: http://www.w3.org/TR/xquery/. [Accessed: 05-Jul-2012].
[58] World Wide Web Consortium, "SPARQL Query Language for RDF," *SPARQL Query Language for RDF*, 15-Jan-2008. [Online]. Available: http://www.w3.org/TR/rdf-sparql-query/. [Accessed: 05-Jul-2012].
[59] World Wide Web Consortium, "Resource Description Framework (RDF): Concepts and Abstract Syntax," *Resource Description Framework*, 10-Feb-2004. [Online]. Available: http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/. [Accessed: 05-Jul-2012].
[60] World Wide Web Consortium, "OWL Web Ontology Language - Overview," *OWL Web Ontology Language - Overview*, 10-Feb-2004. [Online]. Available: http://www.w3.org/TR/owl-features/. [Accessed: 05-Jul-2012].
[61] M. Urban-Lurain, "Intelligent tutoring systems: An historic review in the context of the development of artificial intelligence and educational psychology," 1996. .
[62] D. Sleeman and J. S. Brown, *Intelligent tutoring systems*. Academic Press, 1982.
[63] E. Wenger, *Artificial intelligence and tutoring systems : computational and cognitive approaches to the communication of knowledge*. Los Altos Calif.: Morgan Kaufmann Publishers, 1987.
[64] J. Self, "Theoretical foundations for intelligent tutoring systems," *J. Artif. Intell. Educ.*, vol. 1, no. 4, pp. 3–14, 1990.
[65] M. Merrill, "Instructional Transaction Theory: An Introduction.," *Educ. Technol.*, vol. XXXI, no. 6, p. 7, 1991.
[66] B. Bloom, *Taxonomy of educational objectives : the classification of educational goals*, 1st ed. New York: D. McKay, 1956.
[67] R. Gagné, *The conditions of learning*, 2d ed. New York: Holt Rinehart and Winston, 1970.
[68] P. Kyllonen and V. Shute, *Taxonomy of learning skills*. Brooks Air Force Base Tex.: Air Force Human Resources Laboratory Air Force Systems Command, 1988.
[69] R. Nkambou, "Modeling the Domain: An Introduction to the Expert Module," in *Advances in Intelligent Tutoring Systems*, 1st Edition., Berlin, Heidelberg: Springer-Verlag, 2010, pp. 15–32.
[70] J. S. Brown and R. R. Burton, "SOPHIE: a pragmatic use of artificial intelligence in CAI," in *Proceedings of the 1974 annual ACM conference - Volume 2*, New York, NY, USA, 1974, pp. 571–579.
[71] W. J. Clancey, "Acquiring, representing, and evaluating a competence model of diagnostic strategy," Stanford University, Technical Report, Stanford University Department of Computer Science CS-TR-85-1067, Aug. 1985.

[72] M. Singley, J. Anderson, J. Gevins, and D. Hoffman, "The algebra word problem tutor," in *Proceedings of the 4th International Conference on Artificial Intelligence and Education*, 1989, pp. 267–275.
[73] K. R. Koedinger and J. R. Anderson, "Abstract planning and perceptual chunks: Elements of expertise in geometry," *Cogn. Sci.*, vol. 14, no. 4, pp. 511–550, 1990.
[74] A. T. Corbett and J. R. Anderson, "LISP intelligent tutoring system: Research in skill acquisition," in *Computer-assisted instruction and intelligent tutoring systems: shared goals and complementary approaches*, Hillsdale, New Jersey: Erlbaum, 1992, pp. 73–109.
[75] B. P. Woolf, "Student Modeling," in *Advances in Intelligent Tutoring Systems*, 1st Edition., Berlin, Heidelberg: Springer-Verlag, 2010, pp. 267–279.
[76] J. A. Self, "Formal Approaches to Student Modeling," in *Student Models: The Key to Individual Educational Systems*, New York: Springer-Verlag, 1994.
[77] M. Mayo and A. Mitrovic, "Optimising ITS behaviour with Bayesian networks and decision theory," *Int. J. Artif. Intell. Educ.*, vol. 12, pp. 124–153, 2001.
[78] R. J. Mislevy and D. H. Gitomer, "The role of probability-based inference in an intelligent tutoring system," *User Model. User-Adapt. Interact.*, vol. 5, no. 3–4, pp. 253–282, 1996.
[79] H. T. Everson, "Modeling the student in intelligent tutoring sytems: The promise of a new psychometrics," *Instr. Sci.*, vol. 23, no. 5, pp. 433–452, 1995.
[80] W. Murray, "An easily implemented, linear-time algorithm for bayesian student modeling in multi-level trees," in *Proc. of 9th World Conference of Artificial Intelligence and Education AIED*, 1999, vol. 99, pp. 413–420.
[81] C. Conati, "Bayesian Student Modeling," in *Advances in Intelligent Tutoring Systems*, 1st Edition., Berlin, Heidelberg: Springer-Verlag, 2010, pp. 281–299.
[82] C. Conati, "Using Bayesian Networks to Manage Uncertainty in Student Modeling," *J. User Model. User-Adapt. Interact.*, vol. 12, no. 4, pp. 371–417, Nov. 2002.
[83] M. Mayo and A. Mitrovic, "Using a probabilistic student model to control problem difficulty," in *Intelligent Tutoring Systems*, Montréal, Canada, 2000, vol. 1839, pp. 524–533.
[84] S. Bull and J. Kay, "Student Models that Invite the Learner In: The SMILI:() Open Learner Modelling Framework," *Int. J. Artif. Intell. Educ.*, vol. 17, no. 2, pp. 89–120, 2007.
[85] B. du Boulay and R. Luckin, "Modelling Human Teaching Tactics and Strategies for Tutoring Systems," *Int. J. Artif. Intell. Educ.*, vol. 12, no. 3, pp. 235–256, 2001.
[86] J. Bourdeau and M. Grandbastien, "Modeling Tutoring Knowledge," in *Advances in Intelligent Tutoring Systems*, 1st Edition., Berlin, Heidelberg: Springer-Verlag, 2010, pp. 123–143.
[87] K. VanLehn, "The Interaction Plateau: Answer-Based Tutoring < Step-Based Tutoring = Natural Tutoring," in *Proceedings of ITS 2008, the 9th International Conference on Intelligent Tutoring Systems*, Montreal, QC, Canada, 2008.
[88] B. Woolf, *Building intelligent interactive tutors : student-centered strategies for revolutionizing e-learning*. Amsterdam ;;Boston: Morgan Kaufmann Publishers/Elsevier, 2009.
[89] "Web3D Consortium," *Web3D Consortium*, 26-Aug-2014. [Online]. Available: http://www.web3d.org/.
[90] "Three.js," *Wikipedia, the free encyclopedia*. 31-Aug-2014.
[91] "jQuery," *Wikipedia, the free encyclopedia*. 31-Aug-2014.
[92] S. Kim, J. F. Brinkley, and C. Rosse, "Profile of on-line anatomy information resources: design and instructional implications.," *Clin. Anat. N. Y. N*, vol. 16, no. 1, pp. 55–71, 2003.
[93] E. A. Akl, R. W. Pretorius, K. Sackett, W. S. Erdley, P. S. Bhoopathi, Z. Alfarah, and H. J. Sch&a#x00FC;nemann, "The effect of educational games on medical students' learning outcomes: A systematic review: BEME Guide No 14," *Med. Teach.*, vol. 32, no. 1, pp. 16–27, Jan. 2010.

[94] W. C. McGaghie, S. B. Issenberg, E. R. Petrusa, and R. J. Scalese, "A critical review of simulation-based medical education research: 2003-2009," *Med. Educ.*, vol. 44, no. 1, pp. 50–63, Jan. 2010.
[95] M. Csikszentmihalyi and I. S. Csikszentmihalyi, *Optimal Experience: psychological studies of flow in consciousness*. Cambridge University Press, 1988.
[96] S. S. Boocock and E. O. Schild, "Simulation Games in Learning.," Jan. 1968.
[97] A. K. Gordon, "Games For Growth; Educational Games in the Classroom.," Jan. 1970.
[98] "Trivial Pursuit," *Wikipedia, the free encyclopedia*. 29-Aug-2014.
[99] C. McIntire, "Corpus Morphus: The Human Anatomy Board Game," *Am. Biol. Teach.*, vol. 57, no. 8, p. 538, 1995.
[100] "StudyJams - The Human Body," *StudyJams - The Human Body*, 04-Sep-2014. [Online]. Available: http://studyjams.scholastic.com/studyjams/jams/science/human-body/human-body.htm.
[101] "Health Games - Body Parts - Labeling | Learning Games For Kids." [Online]. Available: http://www.learninggamesforkids.com/health_games/body_parts/labeling.html. [Accessed: 05-Sep-2014].
[102] B. Crossett, *Anatomy Arcade*. 2008.
[103] "Science Net Links - All Body systems," *Science Net Links - All Body systems*. [Online]. Available: http://sciencenetlinks.com/media/filer/2011/10/13/allsystems.swf.
[104] "Purpose Games - 'Anatomy,'" *Purpose Games - "Anatomy."* [Online]. Available: http://www.purposegames.com/games?t=974&so=mp.
[105] "Five Senses - Human Senses Game and Lesson Plan for Preschool Kids." [Online]. Available: http://www.turtlediary.com/preschool-games/science-games/the-five-senses.html. [Accessed: 05-Sep-2014].
[106] "StudyStack - Anatomy Flash Cards," *StudyStack - Anatomy Flash Cards*, 04-Sep-2014. [Online]. Available: http://www.studystack.com/Anatomy.
[107] M. Jensen, "Promoting Higher Order thinking in Freshman-level Anatomy and Physiology," presented at the Annual Meeting of the National Science Teachers Association / Society of College Science Teachers, New Orleans, Louisiana, 20-Mar-2009.
[108] M. Jensen, *Web Anatomy*. University of Minnesota.
[109] "Funtrivia - Human Body," *Funtrivia - Human Body*, 04-Sep-2014. [Online]. Available: http://www.funtrivia.com/quizzes/sci__tech/health_and_human_biology/human_body.html.
[110] E. Akl, R. Mustafa, T. Slomka, A. Alawneh, A. Vedavalli, and H. Schunemann, "An educational game for teaching clinical practice guidelines to Internal Medicine residents: development, feasibility and acceptability," *BMC Med. Educ.*, vol. 8, no. 1, p. 50, 2008.
[111] Duckie deck, "Duckie Deck." [Online]. Available: http://duckiedeck.com/play/stethoscope.
[112] "Quia - Anatomy & Physiology," *Quia - Anatomy & Physiology*, 04-Sep-2014. [Online]. Available: http://www.quia.com/shared/search?category=16116&adv_search=true.
[113] "Health Games - Body Parts - Sliding Puzzle | Learning Games For Kids." .
[114] *Med School*. Scrub Games, University College Cork School of Medicine, 2009.
[115] S. G. Fukuchi, L. A. Offutt, J. Sacks, and B. D. Mann, "Teaching a multidisciplinary approach to cancer treatment during surgical clerkship via an interactive board game," *Am. J. Surg.*, vol. 179, no. 4, pp. 337–340, Apr. 2000.
[116] "Operation," *Wikipedia*. .
[117] "Surgeon Simulator 2013," *Wikipedia, the free encyclopedia*. 24-Aug-2014.
[118] "Trauma Center (series)," *Wikipedia, the free encyclopedia*. 28-Aug-2014.

[119] "Virtual Hip Re-surfacing," *EdHeads*, 05-Sep-2014. [Online]. Available: http://www.edheads.org/activities/hip2/index.shtml.
[120] "Virtual Knee Surgery," *EdHeads*. [Online]. Available: http://www.edheads.org/activities/knee/index.shtml.
[121] "Operation: Heart Transplant," *PBS*. [Online]. Available: http://www.pbs.org/wgbh/nova/eheart/transplantwave.html.
[122] "Life Changing Science - Virtual Open Heart Surgery," *Australian Broadcasting Corporation*, 05-Sep-2014. [Online]. Available: http://www.abc.net.au/science/lcs/heart.htm.
[123] "Oncology Game," *GameUp*, 05-Sep-2014. [Online]. Available: http://www.brainpop.com/games/oncology/.
[124] "Dark Cut 2 | Action Games | Play Free Games Online at," *Armor Games*. [Online]. Available: http://armorgames.com/play/353/dark-cut-2. [Accessed: 05-Sep-2014].
[125] "Amateur Surgeon," *Adult Swim Games*. [Online]. Available: http://games.adultswim.com/amateur-surgeon-twitchy-online-game.html. [Accessed: 05-Sep-2014].
[126] "Hospital Haste," *GameHouse*, 05-Sep-2014. [Online]. Available: http://www.gamehouse.com/download-games/hospital-haste.
[127] "Doctor Life: Be a Doctor!," *Big Fish Games :: Safe & Secure Game Downloads*. [Online]. Available: http://www.bigfishgames.com/download-games/26379/doctor-life-be-a-doctor/index.html. [Accessed: 05-Sep-2014].
[128] "Dead Hungry Diner," *Wikipedia, the free encyclopedia*. 21-Aug-2014.
[129] *Chocolatier*. Big Splash Games LLC, 2007.
[130] "Chocolatier (video game)," *Wikipedia, the free encyclopedia*. 26-Aug-2014.
[131] "The Magic School Bus (book series)," *Wikipedia, the free encyclopedia*. 01-Sep-2014.
[132] C. Corbitt, "The Nervous System Game," *Sci. Child.*, vol. 43, no. 6, pp. 26–29, Mar. 2006.
[133] "Neverwinter Nights," *Wikipedia, the free encyclopedia*. 29-Aug-2014.
[134] S. Marks, J. Windsor, and B. Wünsche, "Evaluation of game engines for simulated surgical training," in *Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia*, Perth, Australia, 2007, pp. 273–280.
[135] S. Marks, "A Virtual Environment for Medical Teamwork Training with Support for Non-Verbal Communication using Consumer-Level Hardware and Software," Doctoral, University of Auckland, Auckland, New Zealand, 2011.
[136] A. Maciel, T. Halic, Z. Lu, L. P. Nedel, S. De, A. Maciel, T. Halic, Z. Lu, L. P. Nedel, and S. De, "Using the PhysX engine for physics-based virtual surgery with force feedback," *Int. J. Med. Robot. Comput. Assist. Surg. Int. J. Med. Robot. Comput. Assist. Surg.*, vol. 5, 5, no. 3, 3, pp. 341, 341–353, 353, Sep. 2009.
[137] J. Mackenzie, G. Baily, M. Nitsche, and J. Rashbass, "Gaming technologies for anatomy education," in *unpublished conference presentation) 7th International Conference on Information Visualisation IV*, 2003, vol. 3, pp. 16–18.
[138] A. Richardson-Hatcher, M. Hazzard, and G. Ramirez-Yanez, "The cranial nerve skywalk: A 3D tutorial of cranial nerves in a virtual platform," *Anat. Sci. Educ.*, p. n/a–n/a, Mar. 2014.
[139] "Second Life," *Wikipedia, the free encyclopedia*. 05-Sep-2014.
[140] "Oculus Rift," *Wikipedia, the free encyclopedia*. 05-Sep-2014.
[141] "Razer Hydra," *Wikipedia, the free encyclopedia*. 04-Sep-2014.
[142] W. Winn, F. Stahr, C. Sarason, R. Fruland, P. Oppenheimer, and Y. L. Lee, "Learning about Puget Sound using virtual reality technology," in *Proceedings of the 17th Biennial Conference of the Estuarine Research Federation*, Seattle, Washington, 2003.
[143] A. Kancherla, "A Novel Virtual Reality Tool for Teaching Dynamic 3D Anatomy," *Lect. Notes Comput. Sci.*, no. 905, p. 163, 1995.
[144] A. State, M. A. Livingston, W. F. Garrett, G. Hirota, M. C. Whitton, E. D. Pisano, and H. Fuchs, "Technologies for augmented reality systems: realizing ultrasound-guided

needle biopsies," in *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, New York, NY, USA, 1996, pp. 439–446.
[145] J. P. Rolland, D. L. Wright, and A. R. Ranchería, "Towards a Novel Augmented-Reality Tool to Visualize Dynamic 3-D Anatomy," in *Studies in Health Technology and Informatics*, 1997, vol. 39, pp. 337–348.
[146] R. G. Thomas, N. William John, and J. M. Delieu, "Augmented Reality for Anatomical Education," *J. Vis. Commun. Med.*, vol. 33, no. 1, pp. 6–15, Mar. 2010.
[147] D. Patten, "What lies beneath: the use of three-dimensional projection in living anatomy teaching," *Clin. Teach.*, vol. 4, no. 1, pp. 10–14, 2007.
[148] "Cellcraft," *Carolina.com*, 05-Sep-2014. [Online]. Available: http://www.carolina.com/teacher-resources/Interactive/online-game-cell-structure-cellcraft-biology/tr11062.tr.
[149] "Game Review - Cellcraft," *[meme - hazard] - Dangerously random ravings..* .
[150] J. Schell, *The art of game design: a book of lenses*. Amsterdam; Boston: Elsevier/Morgan Kaufmann, 2008.
[151] S. Bjork and J. Holopainen, *Patterns in Game Design (Game Development Series)*, 1st ed. Charles River Media, 2004.
[152] C. B. Stapleton, C. E. Hughes, and J. M. Moshell, "MIXED FANTASY: Exhibition of Entertainment Research for Mixed Reality.," in *Mixed and Augmented Reality, 2003. Proceedings. The Second IEEE and ACM International Symposium on*, 2003, pp. 354–355.
[153] T. Nilsen, J. Looser, and S. Linton, "Motivations for Augmented Reality Gaming," in *Proceedings of Fuse "04, New Zealand Game Developer"s Conference*, 2004, pp. 86–93.
[154] H. Gray, *Gray's Anatomy*. J.W. Parker, 1858.
[155] J. C. McLachlan, J. Bligh, P. Bradley, and J. Searle, "Teaching anatomy without cadavers," *Med. Educ.*, vol. 38, no. 4, pp. 418–424, Apr. 2004.
[156] M. J. Ackerman, "The Visible Human Project," *Proc. IEEE*, vol. 86, no. 3, pp. 504–511, Mar. 1998.
[157] Mi. P. D'Alessandro and R. A. Bergman, "Home - Anatomy Atlases," *Anatomy Atlases*, 08-Sep-2014. [Online]. Available: http://www.anatomyatlases.org/.
[158] D. Summers, "Harvard Whole Brain Atlas: www.med.harvard.edu/AANLIB/home.html," *J. Neurol. Neurosurg. Psychiatry*, vol. 74, no. 3, pp. 288–288, Mar. 2003.
[159] M. Schuenke, E. Schulte, U. Schumacher, L. M. Ross, E. D. Lamperti, and M. Voll, *Head and Neuroanatomy*, 1st edition edition. Stuttgart; New York: Thieme, 2010.
[160] C. Rosse, "The potential of computerized representations of anatomy in the training of health care providers.," *Acad. Med. J. Assoc. Am. Med. Coll.*, vol. 70, no. 6, pp. 499–505, Jun. 1995.
[161] "Interact Elsevier - Home," *Interact Elsevier*, 08-Sep-2014. [Online]. Available: http://www.interactelsevier.com/.
[162] "BioDigital Human: Anatomy and Health Conditions in Interactive 3D," *BioDigital Human*, 08-Sep-2014. [Online]. Available: https://www.biodigital.com/.
[163] A. Garg, "How medical students learn spatial anatomy.," *Lancet*, vol. 357, no. 9253, p. 363, 2001.
[164] A. Garg, "Is there any real virtue of virtual reality?: the minor role of multiple orientations in learning anatomy from computers.," *Acad. Med. J. Assoc. Am. Med. Coll.*, vol. 77, no. 10, pp. S97–9, Oct. 2002.
[165] "Anatomy and Physiology," *Primal Online Learning*, 08-Sep-2014. [Online]. Available: http://www.primalonlinelearning.com/.
[166] *Anatomy and Physiology*. OpenStax.
[167] W. Norman, "The Anatomy Lesson," 08-Sep-2014. [Online]. Available: http://www.wesnorman.com/.
[168] R. Whitaker, "Home," *Instant Anatomy*, 08-Sep-2014. [Online]. Available: http://www.instantanatomy.net/index.html.
[169] "Coursera," *Wikipedia, the free encyclopedia*. 03-Sep-2014.

[170] "MIT OpenCourseWare," *Wikipedia, the free encyclopedia*. 24-Aug-2014.
[171] "FutureLearn," *Wikipedia, the free encyclopedia*. 24-Aug-2014.
[172] "Khan Academy," *Wikipedia, the free encyclopedia*. 07-Sep-2014.
[173] "Body Worlds," *Wikipedia, the free encyclopedia*. 24-Aug-2014.
[174] "The Human Body (TV series)," *Wikipedia, the free encyclopedia*. 24-Aug-2014.
[175] "Inside the Human Body - BBC One," *BBC*. [Online]. Available: http://www.bbc.co.uk/programmes/b0110f51. [Accessed: 08-Sep-2014].
[176] J. Kappelman, T. Ryan, and M. Zylstra, "e-Skeletons®: The Digital Library as a Platform for Studying Anatomical Form and Function," *-Lib Mag.*, vol. 5, no. 9, Sep. 1999.
[177] "Nova," *PBS*, 08-Sep-2014. [Online]. Available: http://www.pbs.org/wgbh/nova/.
[178] "Science and Nature - Human Body and Mind," *BBC*, 08-Sep-2014. [Online]. Available: http://www.bbc.co.uk/science/humanbody/.
[179] "3D Yoga Anatomy," *App Store*. [Online]. Available: https://itunes.apple.com/us/app/3d-yoga-anatomy/id558683564?mt=8. [Accessed: 08-Sep-2014].
[180] "iDance!," *App Store*. [Online]. Available: https://itunes.apple.com/us/app/idance!/id786101092?mt=8. [Accessed: 08-Sep-2014].
[181] "Bodyweight Training Anatomy," *App Store*. [Online]. Available: https://itunes.apple.com/us/app/bodyweight-training-anatomy/id681285239?mt=8. [Accessed: 08-Sep-2014].
[182] "Miniatlas Sports Injuries," *App Store*. [Online]. Available: https://itunes.apple.com/us/app/miniatlas-sports-injuries/id401945574?mt=8. [Accessed: 08-Sep-2014].
[183] "The Liberated Voice." [Online]. Available: http://www.claudiafriedlander.com/the-liberated-voice/. [Accessed: 09-Sep-2014].
[184] D. C. Bentley and S. C. Pang, "Yoga asanas as an effective form of experiential learning when teaching musculoskeletal anatomy of the lower limb," *Anat. Sci. Educ.*, vol. 5, no. 5, pp. 281–286, Sep. 2012.
[185] P. Sparks Stein, A. D. Richardson, and S. D. Challman, "The anatomy of self-defense," *Anat. Sci. Educ.*, vol. 1, no. 3, pp. 130–132, May 2008.
[186] "WebMD - Home," *WebMD*, 08-Sep-2014. [Online]. Available: http://www.webmd.com/.
[187] "Anatomy," *About.com*, 08-Sep-2014. [Online]. Available: http://biology.about.com/od/anatomy/.
[188] "Diseases and Conditions," *Mayo Clinic*, 08-Sep-2014. [Online]. Available: http://www.mayoclinic.org/diseases-conditions.
[189] "Osteoporosis Disease V1," *App Store*. [Online]. Available: https://itunes.apple.com/us/app/osteoporosis-disease-v1/id583127509?mt=8. [Accessed: 09-Sep-2014].
[190] "Miniatlas Joint Diseases," *App Store*. [Online]. Available: https://itunes.apple.com/us/app/miniatlas-joint-diseases/id422273756?mt=8. [Accessed: 09-Sep-2014].
[191] D. A. Cook, "The failure of e-learning research to inform educational practice, and what we can do about it..," *Med. Teach.*, vol. 31, no. 2, pp. 158–162, Feb. 2009.
[192] M. Prensky, *Digital Game-Based Learning*, 1st ed. McGraw-Hill Companies, 2000.
[193] C. Perrotta, G. Featherstone, H. Aston, and E. Houghton, *Game-based learning: latest evidence and future directions*. NFER Slough, 2013.
[194] K. L. McClarty, A. Orr, P. M. Frey, R. P. Dolan, V. Vassileva, and A. McVay, "A literature review of gaming in education," *Gaming Educ.*, 2012.
[195] J. Kirriemuir, A. McFarlane, and others, "Literature review in games and learning," 2004.
[196] J. F. Brinkley, B. A. Wong, K. P. Hinshaw, and C. Rosse, "Design of an anatomy information system," *IEEE Comput. Graph. Appl.*, vol. 19, no. 3, pp. 38–48, 1999.

[197] W. Warren and J. F. Brinkley, "Knowledge-based, interactive, custom anatomical scene creation for medical education: the Biolucida system," *AMIA Annu. Symp. Proc. AMIA Symp. AMIA Symp.*, pp. 789–793, 2005.
[198] W. Warren, A. Agoncillo, J. Franklin, and J. Brinkley, "Intelligent Web-based Whole Body Visualization for Anatomy Education," *AMIA. Annu. Symp. Proc.*, vol. 2006, p. 1136, 2006.
[199] N. Mitsuhashi, K. Fujieda, T. Tamura, S. Kawamoto, T. Takagi, and K. Okubo, "BodyParts3D: 3D structure database for anatomical concepts," *Nucleic Acids Res.*, vol. 37, no. suppl 1, pp. D782–D785, Jan. 2009.
[200] A. M. Winkler, "Brain for Blender," *Brainder*, 11-Mar-2013. .
[201] D. M. Smith, A. Oliker, C. R. Carter, M. Kirov, J. G. McCarthy, and C. B. Cutting, "A Virtual Reality Atlas of Craniofacial Anatomy," *Plast. Reconstr. Surg.*, vol. 120, pp. 1641–1646, Nov. 2007.
[202] J. F. Brinkley and C. Rosse, "The Digital Anatomist distributed framework and its applications to knowledge-based medical imaging," *J. Am. Med. Inform. Assoc. JAMIA*, vol. 4, no. 3, pp. 165–183, Jun. 1997.
[203] J.-X. Dai, M. S. Chung, R.-M. Qu, L. Yuan, S.-W. Liu, and D. S. Shin, "The Visible Human Projects in Korea and China with improved images and diverse applications," *Surg. Radiol. Anat.*, vol. 34, no. 6, pp. 527–534, Mar. 2012.
[204] W. B. Lober and J. F. Brinkley, "A Portable Image Annotation Tool for Web-based Anatomy Atlases," *Proc. AMIA Symp.*, p. 1108, 1999.
[205] T. Nilsen and J. Gray, "Semi-automatic segmentation of anatomy from tomographic images for construction of detailed 3D models." 08-Dec-2011.
[206] World Wide Web Consortium, "Cascading Style Sheets (CSS) Snapshot 2010," 12-May-2011. [Online]. Available: http://www.w3.org/TR/css-2010/.
[207] "Phong reflection model," *Wikipedia, the free encyclopedia*. 15-Oct-2014.
[208] B. T. Phong, "Illumination for Computer Generated Pictures," *Commun ACM*, vol. 18, no. 6, pp. 311–317, Jun. 1975.
[209] S. G. Potkin, J. A. Turner, G. G. Brown, G. McCarthy, D. N. Greve, G. H. Glover, D. S. Manoach, A. Belger, M. Diaz, C. G. Wible, J. M. Ford, D. H. Mathalon, R. Gollub, J. Lauriello, D. O'Leary, T. G. M. van Erp, A. W. Toga, A. Preda, and K. O. Lim, "Working memory and DLPFC inefficiency in schizophrenia: The FBIRN study," *Schizophr. Bull.*, vol. 35, no. 1, pp. 19–31, Jan. 2009.
[210] Charles P. Friedman, *Evaluation methods in biomedical informatics*, 2nd ed. New York: Springer, 2006.
[211] D. S. Shin, M. S. Chung, H. S. Park, J. S. Park, and S. B. Hwang, "Browsing software of the Visible Korean data used for teaching sectional anatomy," *Anat. Sci. Educ.*, vol. 4, no. 6, pp. 327–332, Nov. 2011.
[212] R. Nkambou, R. Mizoguchi, and J. Bourdeau, *Advances in Intelligent Tutoring Systems*, 1st Edition. Springer, 2010.
[213] I. Robinson, J. Webber, and E. Eifrem, *Graph Databases*, 1 edition. Beijing ; Sebastopol, CA: O'Reilly Media, 2013.
[214] K. B. Korb and A. E. Nicholson, *Bayesian Artificial Intelligence, Second Edition*, 2 edition. Boca Raton, FL: CRC Press, 2010.
[215] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3 edition. Upper Saddle River: Pearson, 2009.
[216] A. T. Corbett and J. R. Anderson, "Knowledge tracing: Modeling the acquisition of procedural knowledge," *User Model. User-Adapt. Interact.*, vol. 4, no. 4, pp. 253–278, 1994.
[217] K. R. Koedinger, P. I. Pavlik, J. Stamper, T. Nixon, and S. Ritter, "Avoiding Problem Selection Thrashing with Conjunctive Knowledge Tracing," in *Proceedings of the 3 rd International Conference on Educational Data Mining*, 2010, pp. 91–100.
[218] M. C. Desmarais and R. S. J. d Baker, "A review of recent advances in learner and skill modeling in intelligent learning environments," *User Model. User-Adapt. Interact.*, vol. 22, no. 1–2, pp. 9–38, Apr. 2012.

[219] C. Plaisant, J. Grosjean, and B. B. Bederson, "SpaceTree: Supporting Exploration in Large Node Link Tree, Design Evolution and Empirical Evaluation," in *Information Visualization, IEEE Symposium on*, Los Alamitos, CA, USA, 2002, vol. 0, p. 57.
[220] F. Zwicky, *Discovery, invention, research through the morphological approach.* New York: Macmillan, 1969.
[221] T. Ritchey, *Wicked Problems – Social Messes*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011.
[222] T. Ritchey, "Problem structuring using computer-aided morphological analysis," *J. Oper. Res. Soc.*, vol. 57, no. 7, pp. 792–801, 2006.
[223] T. Ritchey, "Fritz Zwicky, morphologie and policy analysis," in *16th EURO conference on operational analysis, Brussels*, 1998.
[224] F. Zwicky, *Morphological Astronomy*, Softcover reprint of the original 1st ed. 1957 edition. Berlin, Heidelberg: Springer, 1957.
[225] C. R. O'Neal, "New approaches to technological forecasting-- Morphological analysis: An integrative approach," *Bus. Horiz.*, vol. 13, no. 6, pp. 47–58, 1970.
[226] R. J. Youmans and T. Arciszewski, "Design Fixation: A Cloak of Many Colors," in *Design Computing and Cognition '12*, J. S. Gero, Ed. Springer Netherlands, 2014, pp. 115–129.
[227] H. Rittel and M. Webber, "Dilemmas in a General Theory of Planning," *Policy Sci.*, vol. 4, pp. 155–169, 1973.
[228] E. de Bono, *Six Thinking Hats*, 2 edition. Boston: Back Bay Books, 1999.
[229] "Wacky Wheels," *Wikipedia, the free encyclopedia*. 25-May-2015.
[230] "Grand Theft Auto (series)," *Wikipedia, the free encyclopedia*. 28-May-2015.
[231] "TrackMania," *Wikipedia, the free encyclopedia*. 08-May-2015.
[232] *Operation Wolf*. Taito, 1987.
[233] "Operation Wolf," *Wikipedia, the free encyclopedia*. 15-Nov-2014.
[234] *The Typing of the Dead*. Sega, 1999.
[235] "The Typing of the Dead," *Wikipedia, the free encyclopedia*. 31-Mar-2015.
[236] *The House of the Dead 2*. Sega, 1998.
[237] "The House of the Dead 2," *Wikipedia, the free encyclopedia*. 12-Jan-2015.
[238] *Dead Hungry Diner*. Black Market Games, 2012.
[239] "Swords and Soldiers," 07-Apr-2015. [Online]. Available: http://www.swordsandsoldiers.com/.
[240] P. Collyer and O. Collyer, *Championship Manager*. 1992.
[241] "Championship Manager series," *Wikipedia, the free encyclopedia*. 24-Apr-2015.
[242] C. Self and T. Lensch, *Crafting Wood Logic Puzzles: 18 Three-dimensional Games for the Hands and Mind*, Fourth Printing edition. Chanhassen, MN: Cool Springs Press, 2006.
[243] M. Herger, *Enterprise Gamification*. .
[244] D. A. Bowman, E. Kruijff, J. J. LaViola, and I. Poupyrev, *3D User Interfaces: Theory and Practice*. Addison-Wesley Professional, 2004.
[245] A. Garg, "Do virtual computer models hinder anatomy learning?," *Acad. Med. J. Assoc. Am. Med. Coll.*, vol. 74, no. 10, pp. S87–9, Oct. 1999.
[246] H. Ishii and B. Ullmer, "Tangible bits: towards seamless interfaces between people, bits and atoms," in *CHI '97: Proceedings of the SIGCHI conference on Human factors in computing systems*, New York, NY, USA, 1997, pp. 234–241.
[247] S. Weghorst, "Augmenting Tangible Molecular Models," in *Proceedings of the 13th International Conference on Artificial Reality and Telexistence*, Tokyo, Japan, 2003.
[248] *Elizabeth Find M.D. - Diagnosis Mystery*. Gunnar Games, Inc., 2009.
[249] "Go Fish," *Wikipedia, the free encyclopedia*. 09-Apr-2015.
[250] "Happy Families," *Wikipedia, the free encyclopedia*. 05-Feb-2015.
[251] M. Dougiamas, *Moodle*. Open Source.
[252] *Blackboard Learning Management System*. Blackboard Inc.
[253] P. Bienstman, *Mnemosyne*. 2006.
[254] P. A. Wozniak, *SuperMemo*. 1987.

[255] L. R. Rubin, W. L. Lackey, F. A. Kennedy, and R. B. Stephenson, "Using color and grayscale images to teach histology to color-deficient medical students," *Anat. Sci. Educ.*, vol. 2, no. 2, pp. 84–88, Mar. 2009.