Vrije Universiteit Brussel
Faculty of Science
Department of Computer Science

SPECIFYING SPATIAL AND PART-WHOLE RELATIONS FOR VIRTUAL REALITY: A LANGUAGE PERSPECTIVE

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF APPLIED COMPUTER SCIENCE
TO ACHIEVE THE DEGREE OF LICENTIATE

Bram Pellens
Academic Year 2002-2003

Promotor: Prof. Dr. Olga De Troyer

Vrije Universiteit Brussel
Faculteit Wetenschappen
Departement Informatica

SPECIFICEREN VAN RUIMTELIJKE EN DEELGEHEEL RELATIES VOOR VIRTUELE REALITEIT: EEN TAAL PERSPECTIEF

EEN VERHANDELING
VOORGEDRAGEN AAN HET DEPARTEMENT TOEGEPASTE INFORMATICA
VOOR HET BEHALEN VAN DE GRAAD VAN LICENTIAAT

Bram Pellens
Academiejaar 2002-2003

Promotor: Prof. Dr. Olga De Troyer

Abstract

Virtual Reality (VR) has become extremely popular in the current decade. The very first idea of having real, almost realistic, participation in the human-computer interaction was presented by Ivan Sutherland in 1965: "make that (virtual) world in the window look real, sound real, feel real, and respond realistically to the viewers actions". With Virtual Reality technology, we attempt to make human vision and interaction with a computer generated world similar to that of the real (physical) world. In this way we enable the user to use his/her experience in interacting with the real world instead of requiring him/her to learn to perceive and interact with a totally different kind of world. Without any doubt VR has attracted a lot of interest of people in the last few years. Being a new paradigm of user interface it offers great benefits in many application areas.

Several approaches have been developed to build Virtual Environments, the worlds created with the Virtual Reality technology. Most approaches require skilled people, making it almost impossible for the common user to use Virtual Reality technology. The new approach, developed at the WISE (Web and Information System Engineering) lab of the Vrije Universiteit Brussel (VUB), is different from the other approaches in such a way that here there is no need for a Virtual Reality technology background to specify a Virtual Environment. However, the approach is still in a premature phase: all the things in the Virtual World that somebody wants to create could only be specified independently of each other. Relations between these things, the objects in the Virtual World, could not be expressed. Therefore, there was a need to be able to specify explicit relations between the objects. The principle behind the WISE approach is to make use of natural language constructs to specify the Virtual World instead of VR technology. Therefore, the challenge of this thesis was to find suitable natural language constructs that are powerful enough to specify different types of relations between objects.

In this thesis, we will introduce two different kinds of relations, the part-whole relations and the spatial relations, into the WISE approach. The part-whole relations are used to specify composed or complex objects where objects are parts of other objects, which can be composed again. The spatial relations are used to describe the spatial layout of the Virtual World. The position of an object is specified relative to

other objects. We will also keep the ability to place an object at an exact location, as was the case before. These relations are found to be very important when describing more advanced Virtual Worlds. They will help the designer in specifying the Virtual World in an intuitive manner.

Samenvatting

Virtuele Realiteit is gedurende het laatste decennium enorm populair geworden. Het idee van een echte, een bijna realistische, participatie in de interactie tussen mens en computer werd voor het eerst beschreven door Ivan Sutherland in 1965: "laat deze (virtuele) wereld in het venster er echt uitzien, echt doen klinken, echt laten voelen, en realistisch laten reageren op de acties van de waarnemer". Met de technologie van Virtuele Realiteit proberen we het uitzicht van, en de interactie met een computer gegenereerde wereld te evenaren aan dat van de echte (fysische) wereld. Op deze manier kunnen we ervoor zorgen dat de gebruiker zijn/haar ervaringen in de interactie met de echte wereld kan gebruiken in plaats van dat hij/zij genoodzaakt is om het waarnemen, en de interactie met een totaal verschillende wereld te moeten aanleren. Zonder enige twijfel heeft VR veel aandacht van de mensen naar zich toe getrokken in de laatste jaren. Dit nieuw paradigma van gebruikers interactie geeft ons grote voordelen in vele toepassingsgebieden.

Er bestaan verschillende benaderingen voor het ontwikkelen van Virtuele Omgevingen, werelden die gecreëerd zijn met behulp van de technologie van Virtuele Realiteit. Voor de meeste benaderingen zijn ervaren en geschoolde mensen nodig waardoor het bijna onmogelijk wordt om als gemiddelde informaticus of analist een Virtuele Wereld te bouwen. De nieuwe aanpak, ontwikkeld door het WISE (Web and Information System Engineering) lab van de Vrije Universiteit Brussel (VUB), is verschillend van elke andere methode in die zin dat het hier geen vereiste is om een Virtuele Realiteit achtergrond te hebben om een Virtuele Wereld te ontwerpen. De methode staat echter nog altijd in zijn kinderschoenen. Bijvoorbeeld moeten alle voorwerpen in de Virtuele Wereld, die iemand wil maken, onafhankelijk van elkaar gespecificeerd worden. Relaties tussen deze voorwerpen, the objecten in the Virtuele Wereld, konden niet uitgedrukt worden. Er was dus behoefte om expliciete relaties tussen de objecten te kunnen specificeren. Het principe achter de WISE benadering is om gebruik te maken van natuurlijke taal constructies om een Virtuele Wereld te specificeren, in plaats van direct VR technologie te gebruiken. De uitdaging van deze thesis was dan ook om geschikte natuurlijke taal constructies te vinden die krachtig genoeg zijn om verschillende soorten van relaties tussen objecten uit te drukken.

In deze thesis zullen we twee verschillende soorten van relaties in de WISE benadering introduceren, de deelgeheel relaties en ruimtelijke relaties. The deelgeheel relaties worden gebruikt om complexe of samengestelde objecten te specificeren waar een object opgebouwd kan zijn uit andere objecten die dan weer samengesteld kunnen zijn. De ruimtelijke relaties worden gebruikt om de ruimtelijke lay-out van de Virtuele Wereld te beschrijven. De positie van een object wordt gespecificeerd door middel van de positie een ander object. Deze relaties zijn zeer belangrijk voor het beschrijven van meer complexe Virtuele Werelden. Ze helpen de designer in het specificeren van de Virtuele Wereld op een meer intuïtieve manier.

Acknowledgements

I would like to sincerely thank Prof. O. De Troyer for making it possible to realize this thesis and for her suggestions and help during the development of this document. Also thanks go to Peter Stuer, Raul Romero and Wesley Bille for their contributions during the various gatherings. Special thanks go to Wesley Bille in particular for his comments and his all-round support in this work. Next, I would like to thank all the professors and their assistants I had during my period here at the university for giving me a good educational foundation.

Apart from all these persons I would like to thank my parents for making it possible for me to do this kind of studies, and who supported me mentally. I am also very thankful to my friends, especially my girlfriend Els Klerkx, and Peter Plessers, for getting me through some hard times and supporting me in every ways they could.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Introduction

Nowadays computer graphics is used in many domains of our life. Entering the 21th century, it becomes difficult to imagine an architect, engineer, interior designer, or game developer working without a graphical toolkit. Computer graphics has drawn much attention in the last few years and extensive media coverage causes this interest to grow. Also the production of cheap computers with better and faster graphics allows the average user to move into the world of computer graphics. We can create and manipulate three-dimensional worlds but this is not enough: people always want more. They want to be able to step into this world, be part of this world and interact with it. This technology, which became popular in the current decade, is called Virtual Reality (VR) (see section 1.2 for an introduction on Virtual Reality). The worlds created with this technology are called Virtual Environments (VE). Section 1.3 will discuss Virtual Environments in more detail.

As the technology became more and more popular, many ways have been developed to create Virtual Environments. The approach to build Virtual Environments that is developed at the WISE (Web and Information System Engineering) lab of the Vrije Universiteit Brussel (VUB) is different from all other methods as it is using ontologies as a description formalism for the world to be generated (see section 1.4). Earlier readings on the WISE approach have proved that this approach works but still there has to be done a lot of work. In this thesis we will make some contributions to this approach. The argumentation for these contributions is given in section 1.5. Section 1.6 will give us an outline of the remainder of this work.

1.2 Virtual Reality

The very first idea of Virtual Reality was presented by Ivan Sutherland in 1965 as described in [22]: "make that (virtual) world in the window look real, sound real, feel real, and respond realistically to the viewers actions".

The terms Virtual Reality (VR) and Virtual Environments (VE) are used in computer community interchangeably. These terms are the most popular and most often used, but there are many others. Just to mention a few important ones: Synthetic Experience, Virtual Worlds, Artificial Worlds or Artificial Reality. All these terms refer to immersive, interactive, three-dimensional computer generated environments and to the technologies required to build these environments. Most of the definitions in literature correspond with each other, but sometimes we can find some small differences. In the rest of this work we will use the terms Virtual Environments and Virtual Worlds.

Virtual Reality offers great benefits in many applications [22]. We give some examples. People have been gathering a lot of data for a long time. The management of all this data is not an easy task. In the area of architectural visualization, a huge amount of data can be easily navigated in three-dimensional spaces by virtually walking through the data. In the field of modelling and designing, Virtual Reality offers the designer the possibility to watch how the modelled object will look like. The designer can see and feel the product before it is built. When testing the products after the modelling phase, Virtual Environments are used as test environments. Testing products this way will result in a much cheaper production life cycle then when the products are being built, physically tested and adapted when necessary. In the training and education area Virtual Reality is used in simulators e.g. flight simulators for developing and testing the skills of a future pilot. And finally, in the entertainment area, various multi-player games make use of Virtual Reality techniques. These games enable the user, or teams of users, to meet others in virtual rooms for playing against each other.

1.3 Virtual Environments

The term Virtual Environments is defined by Ellis using the following definition: "Virtual Environments can be defined as interactive, virtual image displays enhanced by special processing and by non-visual display modalities, such as auditory and haptics, to convince users that they are immersed in a synthetic space" [13]. So, Virtual Environments (VEs) are applications that let users navigate and interact with a three-dimensional, computer-generated (and computer-maintained) environment in real time. Users can move around, look at, and manipulate graphical objects using input devices ranging from a keyboard and mouse to HMDs (Head Mounted Display) and cyber gloves. In this work we will focus on the Desktop VR. This refers to the use of a conventional computer monitor as the output device onto which the 3D environment is rendered in contrast to the immersive VR where some kind of immersive display and tracking equipment is used in order to create the illusion of being inside the computer-generated environment, rather than viewing it from the outside through a screen.

1.4 Approaches to build Virtual Environments

Several approaches for building Virtual Environments have been developed. The two major approaches in creating Virtual Reality are the development toolkits and authoring systems. With the development toolkits, the user has to program the Virtual Environments by hand. An authoring system allows the user to interactively construct the Virtual World by means of a user interface and some input devices. The new approach for building Virtual Environments using knowledge representations (such as in ontologies), developed at the WISE lab, is different from the other methods. See figure 1.1 for the general approach. The ontology is used here as a mechanism for the user to describe the Virtual World. In chapter 2, we will discuss ontologies in more detail. Generating Virtual Environments from ontologies does not require the user to have a good knowledge of how to write programming code in a specific language. It also does not require the user to know how to work with complicated tools to design Virtual Environments through a user interface. In other words, using this new approach you do not need to be skilled in the VR technologies to be able to create Virtual Environments. The user just builds a domain ontology containing all the information that is needed to build the Virtual Environment. From this ontology the actual Virtual World is being generated. The work presented in this thesis builds on previous research in this topic as is described in [5].

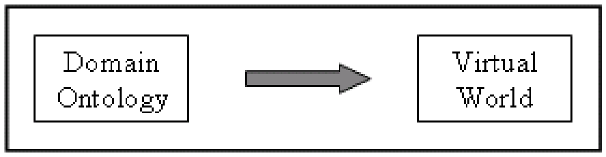


Figure 1.1: From a domain ontology to Virtual World

All the areas, which benefit from Virtual Reality, described in section 1.2, require rather complex Virtual Worlds to be built. Until now, the approach WISE has followed to generate Virtual Environments from domain ontologies only allowed to build very simple worlds. It was possible to create simple structures by describing simple domain objects. All the objects we described in the domain ontology were directly mapped to primitive objects provided by a toolkit (e.g. a box, a cylinder, a sphere). These objects were then placed at a specific location in the Virtual Environment. Afterwards, the code for the Virtual Environment was generated.

1.5 Building Virtual Environments in an Intuitive Way

Building Virtual Reality applications is in general complex and non-intuitive. Creating a Virtual World from scratch, or modifying an existing one, is a very complex task. Most tools available in the areas of 3D modelling have the capability of producing large Virtual Worlds but are difficult to use, require many hours of training, and are expensive. So, most of the times, they are unavailable for the inexperienced user. Therefore, the main goal of the WISE approach is to make the development of Virtual Worlds more accessible, also to non-VR specialists.

The principle behind the WISE approach is to make use of natural language terms to specify a Virtual World. More people could be involved in the development of Virtual Environments if it wasn't a complicated mathematical problem where all kinds of difficult VR terms are used. Using natural language to build a Virtual Environment is very intuitive for everyone because we all use it in our every day life. People already have a good notion of how to express, to someone else, by means of natural language, what objects or scenes (e.g. a room) look like. They make descriptions by means of relationships between objects, in the world or part of the world they want to describe, to do this.

In this thesis we will contribute to the WISE approach by proposing a way to model relationships between objects in a Virtual World that is as intuitive as possible. As was said before, the WISE approach is not yet fully elaborated. At the start of this thesis work, it was only possible to place objects at exact locations in a Virtual Environment and also composed objects were not yet supported. It was not possible to express explicit relations between objects [4]. However, in reality, the positions of most objects are related to each other and objects are composed of other objects. So relations between the objects are needed. In this context, two aspects are very important: composed objects and spatial layout. These are the two aspects on which we will concentrate in this thesis. We shortly introduce them in the following subsections. For each of these aspects we will introduce some relations later in this work.

1.5.1 Composed Objects

To adequately model Virtual Environments there is a need to have composed or complex objects. In VR applications, simple objects are used as building blocks to create complex objects, which in turn serve as building blocks for even more complex objects, and so on. Object hierarchies are very common because most objects are composed of several other objects and only a few objects are indivisible. If we decompose an object into a collection of parts, we already have created a two-level hierarchy. The greatest benefit that the designer will encounter by creating a hierarchy is the update propagation. A change in the definition of one building-block object is automatically propagated to the other levels of the hierarchy that use the object. If we would for example re-locate a composite object, all the sub-components belonging to this composite object will be re-located as well [14]. But knowing that an object is composed of a collection of other objects is not sufficient. Much more information is needed to be able to visualize a composed object and to allow interaction with a composed object. The ways in which an object is composed of several other objects can for example depend on various characteristics. So, being able to describe a Virtual World using composed objects will be a great contribution to the WISE approach.

1.5.2 Spatial Layout

From the moment we enter a place, we are engaged in spatial cognition. We perceive the place from the viewpoint of our own eyes. We interact with the world around us and thereby automatically construct mental representations of that world and our own position in it. Yet, we do not conceive the world as a geometric space, a space with three extrinsic orthogonal coordinates that specify the locations of points, objects, or regions [34].

The WISE approach was lacking appropriate concepts to model the spatial layout of a Virtual Environment in an intuitive way. An object had to be placed inside the Virtual Environment at a specific location described by three coordinates. However, in our mental representation of the world we incorporate different elements in constructing mental spaces for the real spaces important in our lives. Elements, such as objects, landmarks, cities, or continents, are remembered relative to other elements. For example, when people are asked where they live, they often describe their home position in terms relative to the nearest landmark likely to be known to the questioner, e.g. 500 meters north of the church. Therefore, we find it important in describing a Virtual Environment to give the possibility to define the layout of the Virtual Environment without the use of specific coordinates, but by specifying the location of objects relative to the position of other objects. This way is closer to the mental representation of space used by humans and therefore fits better in the intuitive modelling approach used by WISE.

1.6 Summary and outline

Virtual Reality is a technology that is gaining much popularity and there are many application areas for which it could be useful. However, almost all techniques available for creating Virtual Environments are far too difficult for the average designer to use. A new method to build Virtual Worlds is developed at the WISE lab. In this approach, Virtual Environments are generated from domain ontologies. The advantage is that a method based on this approach, can be used by a broader range of people, who do not necessarily have to be skilled in Virtual Reality technology. The method is still in an early phase but the results so far are very promising.

As already pointed out, the goal of this thesis is to contribute to the approach by investigating the possibility to build more advanced Virtual Worlds. We will do this by providing intuitive modelling concepts for composed objects and spatial layout.

To achieve this goal, we will introduce two kinds of relations between objects that will allow the designer to specify more complex Virtual Environments in a more intuitive manner. The first kind of relations are the *part-whole* relations. These relations allow defining composed (or complex) objects inside the Virtual Environment. The second kind of relations, the *spatial* relations, will provide the designer the possibility to specify a spatial layout for the objects. To incorporate these new concepts in the WISE approach, the concept of an upper ontology is used. The background information for this thesis is given in the chapters 2 to 5. An introduction to upper ontologies is given in chapter 2. Chapter 3 describes the current WISE approach to use ontologies to generate a Virtual Environment. In chapter 4, background knowledge about part-whole relations is given and in chapter 5 this is done for the spatial relations. Chapter 6 gives the formal definitions of the new concepts and presents the upper ontologies that are built. In chapter 8, we describe how part-whole relations and spatial relations will be incorporated into the current WISE approach. This chapter also discusses the extensions needed to the current prototype tool, supporting the WISE approach, to include the new concepts. The use of the new concepts are illustrated with an example in chapter 7. Chapter 9 indicates further work and finally, in chapter 10 general conclusions are made.

Chapter 2

Upper Ontologies

2.1 Introduction

Building systems based on some kind of knowledge representation involves the creation of a model of the particular domain under consideration. Such a model is often an abstraction of the domain itself and this is what makes it very powerful and useful too. It allows to abstract from irrelevant details and hereby allows us to focus only on the concepts we are interested in.

In this chapter we will give a detailed description of ontologies, the formalism for knowledge representation used. Ontologies are used for describing the Virtual Worlds we want to visualize. In the current system that we will discuss in the next chapter, we are only able to describe simple Virtual Worlds. Simple Virtual Worlds result from simple ontologies. To describe more complex objects, we need to define some high-level structures that can be used to do this. The designer will then be able to use these structures to describe the Virtual World. An upper ontology will provide us the place in which we can define those high-level structures. What we need is some kind of hierarchy of ontologies dependent on each other, and each specializing the concepts of the ontologies higher up in the hierarchy.

This chapter first includes a general definition of an ontology in section 2.2. Then we will discuss the different uses for ontologies in section 2.3. Afterwards, in section 2.4, we will see that ontologies can be classified into different kinds, namely upper-level ontologies and lower-level ontologies of which the upper-level ontologies are the most important ones to define the structures to be used for describing complex Virtual Worlds. A summary and an evaluation is given in section 2.5.

2.2 What are ontologies?

Ontologies have gained much popularity over the last few years, especially in the knowledge engineering area. However, the definition of ontology remains a bit vague because of the fact that the term is used in many different ways. A first interpretation that is radically different from all the other definitions (described in [19]) is the one with a capital "O" referring to a philosophical discipline. The other ones (with a lowercase "o") all relate somehow to knowledge bases or logical theories.

In the first sense Ontology is the science or study of being. Specifically, this area of philosophy investigating the nature and the relations of reality, a particular system according to which problems of the nature of being are investigated. Here Ontology tries to give answers to questions like: What is being? Or in other words: What are the features common to all beings? [19].

In the field of artificial intelligence and knowledge representation, the ontology describes some sort of world view, with respect to a given domain. The world view is often conceived as a set of concepts (e.g. entities, attributes, processes); their definitions and their inter-relationships. In the literature, this is often referred to as a conceptualization. This conceptualization can be either implicit or explicit. Implicit when a person has some kind of view of (a part of) the world inside his head, and explicit when we speak of a real representation of the view of (a part of) the world. This leads to one of the best-known definitions of an ontology, which states that an ontology is an explicit specification of a conceptualization. An elaborate definition is given in [17]. An interpretation of this definition is given in [19]. In our work it is assumed that an ontology is a knowledge representation formalism in which all knowledge about the Virtual World is being described. Also the knowledge that is needed in the intermediate steps to come to the Virtual World, is being described by means of ontologies.

2.3 Uses of ontologies

Originally, ontologies are used to improve communication between either humans or computers. Using plain text on the one hand is perfectly understandable by human beings and therefore usable for communication in between but not for communication between humans and computers. On the other hand, a language that is perfectly interpretable by computers is in most cases too inaccessible for most humans. This is why there is a need for something in between, and ontologies are the answer to this need. Broadly, the uses of ontologies may be grouped into the following three areas: to assist in communication between humans and agents, to achieve interoperability of systems, or to improve the process and/or quality of engineering software systems [36]. Applications based on these areas are discussed in [35].

2.3.1 Communication

An ontology enables shared understanding and communication between people within an organization or community, which often have different views onto the system or the world. Having a "model" of the system reduces the conceptual and terminological confusion. Using the vocabulary described in the ontology will allow everyone to perfectly understand what the other one is talking about in a conversation. Here, an informal ontology may be sufficient as long as it provides unambiguous definitions for terms used in the organization.

2.3.2 Inter-Operability

In contrast to communication between people, also computers themselves have to be able to interact. Many applications these days address the issue of inter-operability. Users that work with different tools have to be able to exchange their data. Applications based on domain-specific ontologies will be able to do this. To assist in the inter-operability, an ontology can also be used to support the translation between different languages and representations. Dimensions in the inter-operability can be distinguished, namely the internal inter-operability in which the various systems that require inter-operation are under control of the same organizational unit and external inter-operability where the organization wants to insulate itself from changes of the outside (e.g. differences in the format when a new version is released or if a new supplier is chosen).

2.3.3 Systems engineering

Finally, ontologies are also used to support the design and development of software systems. This can be subdivided in three parts. First, a shared understanding of the problem at hand can assist in the specification of a software system. Ontologies facilitate the process of identifying requirements and may allow to understand and reason about the system, which is very useful for the developers. Secondly, ontologies can improve the reliability of software systems. The system can be automatically checked against the declarative specification for its consistency. And thirdly, ontologies must give us support in reusability. Systems often need the ability to import ontologies that were exported by other systems and vice versa, so they must serve as reusable and interchangeable modules.

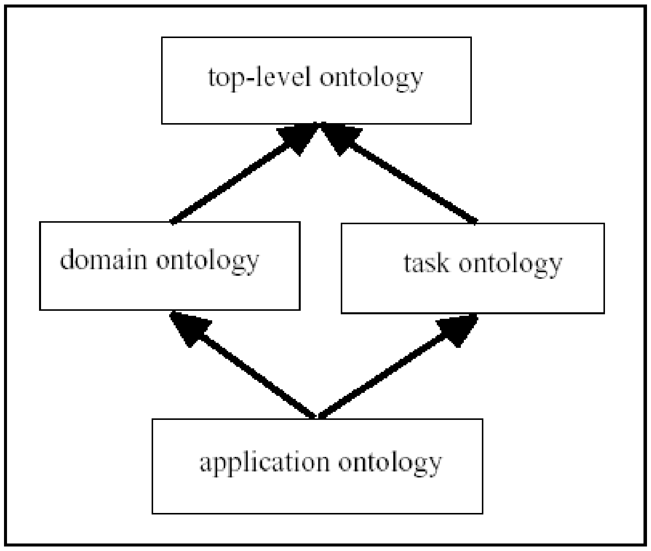


Figure 2.1: Kinds of ontologies: according to dependence

2.4 Classification of ontologies

Ontologies can be classified according to two dimensions as described in [18]. First, their level of detail can be considered. An ontology may contain very specific concepts from a specific domain. We usually refer to these ontologies as domain ontologies. On the other hand we can have ontologies that only contain very general concepts applicable in many different domains. The second dimension is the level of dependence. Ontologies can be defined in a multilevel structure, more specific ontologies depend on the concepts specified in higher-level ontologies. Figure 2.1 displays the classification of the ontologies. We will go deeper into the different types in the next subsections. This figure only shows three levels but it is always possible to define more levels of abstraction. The arrows represent specialization relationships.

2.4.1 Upper level ontology

In section 2.3, ontologies have been recognized to be important mechanisms for several tasks, such as knowledge engineering. In this section we would like to discuss one specific type of ontology, namely upper ontologies, also called core ontologies, or reference ontologies.

When building a library of very large ontologies about a specific domain, some concepts are reoccurring over and over again. In the highest level of abstraction, in the highest nodes in the hierarchy, we are dealing with very abstract terms. These terms do not really contribute to the knowledge of the domain itself because they are too abstract. They are more suitable for representational purposes than for the terms in the knowledge base itself. Hence, they may be taken away from the library and put into a "core ontology" [37]: a very general ontology independent of a certain application domain or problem. They are similar to domain ontologies, but the concepts that they define are considered to be generic across many fields while the domain ontologies describe generic knowledge, specific to a certain domain [38]. Typically, generic core ontologies define concepts like state, event, process, action, component etc. This separation will give a high reusability to the concepts defined in the library of domain ontologies. Many concepts that would otherwise be in the domain ontologies, and duplicated in many other domain ontologies, are now in one upper ontology.

In the definition of the upper ontology, the term "domain independency" is of great importance. Upper ontologies are independent of any domain, which allows us to have content-neutral inter-operability. When we spoke about inter-operability before, we were always referring to a same application domain. However, there may also be a need to have inter-operability between different fields. If different areas of scientific research for example need to share or exchange structured data and when the areas do not really rely on the content of the data. In these situations an ontology is useful since it provides rich structures for representing the form of the knowledge, but is content-neutral. Upper level ontologies provide this possibility [9].

One of the most famous upper ontologies is the IEEE Standard Upper Ontology. The Standard Upper Ontology (SUO) specifies the syntax and semantics of a general-purpose upper level ontology. It is estimated to contain between 1000 and 2500 terms plus roughly ten definitional statements for each term. The SUO plays the role of a neutral interchange format whereby owners of existing applications will be able to map existing data elements just once to a common ontology. This provides a degree of inter-operability with other applications whose representations are conforming to the SUO. This entails the SUO being able to be mapped to more restricted forms such as XML, database schema, or object oriented schema. More information about upper ontologies can be found at [24].

2.4.2 Lower level ontologies

Next to the upper level, which defines the general domain independent concepts, we can distinguish two other kinds of ontologies, dependent on the upper level ontology. These are the domain ontology and the task ontology. These kinds of ontologies describe, more specific but still very generic, the concepts related to respectively a particular domain or task, by specializing the concepts in the upper level ontology.

A specialization of the domain ontology and the task ontology, is being called an application ontology. This type of ontologies describe concepts relying on the concepts of both the higher level ontologies. Here, the concepts are not abstract anymore but very specific towards some kind of domain and even a particular application. In the system, that we will describe in chapter 3, we also have a domain ontology. The ontology there only carries this name because it describes the concepts from the problem domain, the Virtual World the designer wants to generate.

2.5 Summary and Evaluation

Ontologies have become very popular in the last decade. They describe a set of concepts, and the relationships among them, which results in a representational vocabulary with which a knowledge-based program may represent knowledge.

Our view of an ontology is a formalism in which the user can specify a Virtual World, as we will explain in chapter 3. In the current system it is only possible to describe very simple Virtual Worlds containing simple objects. The purpose of this thesis is to make it possible to describe more complex objects. But to specify more complex objects we need to provide the designer with some predefined concepts, or language constructs, to do so. These will be provided my means of an upper ontology. Afterwards, the user can specify the world in terms of the general concepts provided in our upper ontology.

However, we would like to note that our meaning of an upper ontology will differ a bit from the one described earlier in the chapter. Our high-level ontology will not be a high-level ontology in the strict sense of the word, defining very general concepts independent of any application domain, but it will define concepts related to the domain of describing complex (structures of) objects in a Virtual Environment. The concepts will be independent of any specific Virtual World that is defined in the domain ontology.

Chapter 3

Generating VEs from Ontologies

3.1 Introduction

Currently, a number of different ways are available to develop VR applications. Those ways can be categorized in toolkits on the one hand, and the authoring systems on the other hand. The approach used here to generate Virtual Worlds from knowledge representations is very different. Virtual Environments were discussed in section 1.3.

In the conventional cases skilled people are needed to develop a VR application because they either have to have a good knowledge of how to write programming code in a specific Virtual Reality language or have to know how to work with the tools to design a Virtual Environment through a user interface. In the last case it is required to use some scripting language for specifying complex behaviour. So, in both cases, the user has to write programming code. The new approach, which is developed at the WISE (Web Information and Systems Engineering) lab at the Vrije Universiteit Brussel, enables the user to generate a Virtual World from an ontology, a formal description of the world, thus specifying the Virtual World in terms of concepts and objects in the problem domain, very intuitive, and without the need for programming skills or to work with complicated tools.

In this chapter we will describe the new way of creating Virtual Environments by means of ontologies into more detail. For more information I would like to refer the reader to [33][5]. Note however that the system described here differs a bit from the one described in [33] because an extension was made during my apprenticeship. First, an overview of the approach is given (section 3.2). Next, in section 3.3, the different ontologies used by the system are explained. In the following section, section 3.4, more information about the generation process is given and finally, section 3.5 will briefly present the latest prototype of the system, called OntoWorld. A summary of the chapter will be given in section 3.6.

3.2 Overview of the Approach

In chapter 2 we already discussed ontologies together with their uses. In the system described here, ontologies are used to make the development of software and in particular, Virtual Worlds, more intuitive and thus easier. The final goal of the research is to generate code for a Virtual World that is entirely described by means of a specific domain ontology. In the best case, the domain ontology should contain all information needed to complete, without interference of the user, the generation process. The result of the generation process will be a piece of programming code for the world that has to be generated. This programming code can then be compiled and executed. Currently, we are still far away from that goal but we hope to reach this ideal situation in the future.

The gap between the Domain ontology, containing the description of the world we want to generate, and the current Virtual World is too big to be captured in one translation step. The process is therefore broken into several phases. In a first phase the Domain ontology is being translated into a Representable Domain ontology. This Representable Domain ontology describes how the objects in the domain should be represented in the Virtual World. In the next step, the result of the first step is being translated to become a working Virtual Environment (C++ source code in the prototype of the system). Figure 3.1 shows an overview of the complete process. We can see that there is a separate ontology containing the instances, namely the Instances ontology, but the instances might have been put in the Domain Ontology as well. An elaborate description of the system of ontologies is given in the following section.

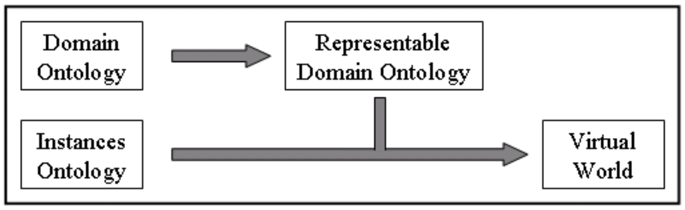


Figure 3.1: The complete generation process

In the prototype that has been developed, the code generated for the Virtual Environment can be executed using the Open Dynamics Engine physics engine written by Russel Smith [30]. ODE is a free, industrial quality physics engine for simulating articulated rigid body dynamics, like ground vehicles, legged creatures, and moving objects in VR environments. It is fast, flexible, robust and platform independent, with advanced joints, contact with friction, and built-in collision detection.

3.3 The System of Ontologies

The use of ontologies provides a very powerful way to describe objects and their relationships to other objects. In the prototype of the tool implementing this new approach, we use the DAML+OIL language for representing the ontologies [10]. The DAML language is being developed as an extension to the Extensible Markup Language (XML, [6]) and the Resource Description Framework (RDF, [21]). The latest release of the DAML+OIL language provides a rich set of constructs with which to create ontologies and to mark-up information so that it is machine readable and understandable.

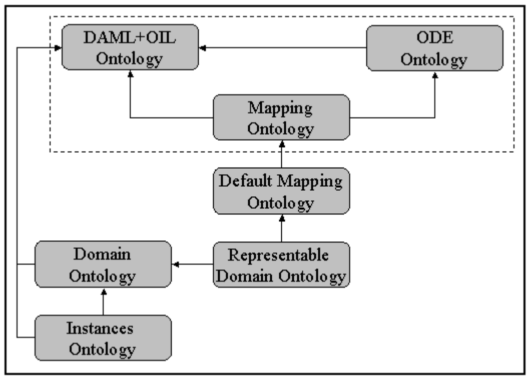


Figure 3.2: Overview of the ontologies

An overview of the ontologies used in our system and their relationship to each other is given in figure 3.2. A description of the ontologies is given in following subsections. For more details on the basic ontologies we would like to refer to [33]. Again, the ontologies here differ a bit from those in the original system.

3.3.1 The DAML+OIL ontology

The DAML+OIL ontology is the ontology describing the DAML+OIL language as specified by [10]. DAML+OIL is a collection of classes, properties, and objects that are added to RDF and RDF-Schema. The ontology allows us to describe information - objects and their relations with each other. It also serves as the vocabulary that the creator of the Virtual Environments is allowed to use to describe (part of) the universe. The complete specification of this ontology can be found on the DAML website [11].

3.3.2 The ODE ontology

The ODE ontology, previously called the vortex ontology, is the ontology that gives us the description of the object types available in the ODE physics engine, including the object types of the primitives that can be displayed using the engine, which are called the representables. Besides these representables, the ontology also describes concepts which are not representable, the non-representables e.g. gravity in the world, bounciness of objects, etc. . . The ODE ontology is written in the DAML+OIL language as indicated by the arrow in figure 3.2.

3.3.3 The Mapping ontology

The Mapping ontology is the ontology describing the function that maps DAML+OIL concepts onto ODE concepts.

```
<daml:Class rdf:ID="ObjectRepresentation">
  <rdfs:subClassOf>
    <daml:Restriction daml:cardinality="1">
      <daml:onProperty rdf:resource="#Source"/>
      <daml:toClass rdf:resource="<<DAML.daml>>#Thing"/>
    </daml:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <daml:Restriction daml:cardinality="1">
      <daml:onProperty rdf:resource="#Target"/>
      <daml:toClass rdf:resource="<<OdeOntology.daml>>#WorldConcept"/>
    </daml:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <daml:Restriction daml:minCardinality="0">
      <daml:onProperty rdf:resource="#Representation"/>
      <daml:toClass rdf:resource="#AttributeRepresentation"/>
    </daml:Restriction>
  </rdfs:subClassOf>
</daml:Class>
```

In the code above, we can clearly see that the function *ObjectRepresentation* has a source-object of type "daml:Thing". This means that every DAML+OIL concept can be a source-object. Furthermore, the function has a target-object that has to be a concept described in the ODE ontology. Besides the source and target, the function also has some representation properties, each containing a representation of an attribute. The function *AttributeRepresentation* has, like the *ObjectRepresentation*, a source and a target. This is different from the system described in [33] where the representations of attributes were not included.

3.3.4 The Default Mapping ontology

The Default Mapping ontology is an ontology containing an instantiation of the mapping in the Mapping ontology and hereby defining a default mapping. The default mapping is a kind of template mapping that is always supposed to exist and used when no specific mapping is specified by the developer. This way, the developer does not need to specify representations for all the objects because the system will provide a default representation in case no representation is present. Objects for which there isn't any representation will be represented according to this default mapping. This is partly to satisfy the requirement of having as less interaction as possible. But it also allows the designer to do fast prototyping by specifying only the most important representations and thus saving a lot of time in the specification process. In our system, the default representation is a sphere - which is one of the primitives in the ODE ontology. The default will only be used when there is no specific mapping defined.

3.3.5 The Domain ontology

The Domain ontology is the ontology that contains all the information that is needed to build the Virtual Environment. It contains a set of classes, the objects, properties and attributes, which we want to be able to visualize in the Virtual World. In contrast with the old system, as described in [33], the classes will not be represented in the world itself, but instances of the classes will be used for this purpose.

The following piece of DAML+OIL code defines an object *Planet* with one property *Color*.

```
<daml:DatatypeProperty rdf:ID="Color">
  <DefaultValue>red</DefaultValue>
</daml:DatatypeProperty>

<daml:Class rdf:ID="Planet">
  <rdfs:subClassOf>
    <daml:Restriction daml:cardinality="1">
      <daml:onProperty rdf:resource="#Color"/>
      <daml:toClass rdf:resource="<<XMLSchema>>#string"/>
    </daml:Restriction>
  </rdfs:subClassOf>
</daml:Class>
```

3.3.6 The Instances ontology

The Instances ontology describes all the instances of the classes in the Domain ontology that will actually be visualized in the Virtual Environment. The Domain ontology with its classes defines the common properties of the instances of a class. The Instances ontology keeps the values for the attributes of the instances.

The following code is an instance of the *Planet* class as shown above with one attribute named *Color*.

```
<Planet rdf:ID="Venus">
  <Color>blue</Color>
</Planet>
```

3.3.7 The Representable Domain ontology

The Representable Domain ontology is the ontology containing the actual mapping from domain objects to ODE objects. It contains instances of the *ObjectRepresentation* class. For every class in the Domain ontology there will be a representation. If not, the default mapping will be used (given in the Default Mapping ontology).

```
<RDFNsId1:ObjectRepresentation rdf:about="<<repre.daml>>#PS">
  <RDFNsId1:Source rdf:resource="<<domain.daml>>#Planet"/>
  <RDFNsId1:Target rdf:resource="<<OdeOntolog.daml>>#Sphere"/>
  <RDFNsId1:Representation>
    <RDFNsId1:AttributeRepresentation rdf:about="<<repre.daml>>#CC">
      <RDFNsId1:Source rdf:resource="<<domain.daml>>#Color"/>
      <RDFNsId1:Target rdf:resource="<<OdeOntology.daml>>#Color"/>
    </RDFNsId1:AttributeRepresentation>
  </RDFNsId1:Representation>
</RDFNsId1:ObjectRepresentation>
```

We can see in the code above that the instance has a *source* and a *target* containing respectively the domain object and the ODE object to be related to each other. In this example, all the instances of the class *Planet* will be represented by a sphere. In addition, the *ObjectRepresentation* contains multiple *Representation* properties. One for each property of the domain object. This property contains an instance of the *AttributeRepresentation* class, which describes the mapping of a property in the domain to an attribute in ODE.

3.4 Generation Process

As was already explained in section 3.2 the generation process is decomposed into two steps (see fig 3.1). These two steps will be explained in detail in the following sections.

3.4.1 Step 1: Building the Representable Domain Ontology

The first step is to create a Representable Domain ontology. The Domain ontology is used as input for this step. For every class in the Domain ontology the user is asked for a suitable physical representation from the ODE ontology. After connecting a class from the Domain ontology to an ODE object from the ODE ontology, the user is asked to give representations for the attributes that are needed for the ODE object on which the object is mapped. The needed attributes can be found in the ODE ontology. Classes without physical representation are given the default mapping as described in the Default Mapping ontology. The output of this step is the Representable Domain ontology containing mappings as is described in section 3.3.

At this moment this process is still highly interactive, the designer has to define the mappings manually. For each class created in the Domain ontology the designer has to specify the mapping (unless he wishes to use the default representation). We are however striving to have as less interactivity as possible.

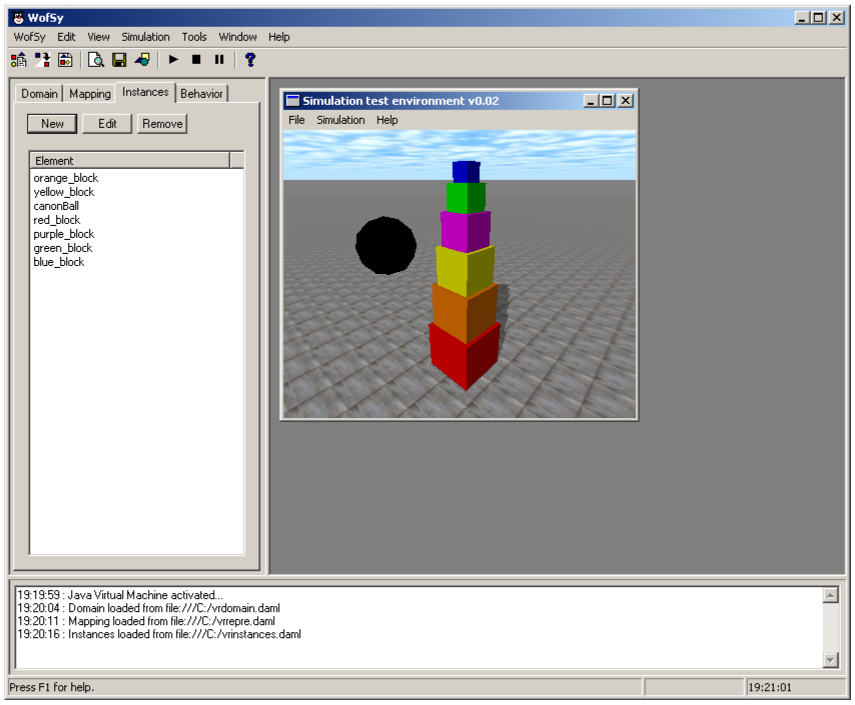
3.4.2 Step 2: Building the Virtual World

The second step is building the actual Virtual World itself. In this step, the Instances ontology and the Representable Domain ontology are required as input. The Instances ontology is taken and for every instance of a class in the Domain ontology a translation is made by means of the Representable Domain ontology. With all the instances expressed in terms of ODE concepts we can easily build the necessary C++ code for the Virtual World. After this step, the code can be compiled and executed.

3.5 The OntoWorld Tool

The generation process described in this chapter is implemented in a prototype tool. A Domain ontology can be loaded after which a Representable Domain ontology and an Instances ontology can be created, or loaded. The Virtual World can be previewed in code format or in a simulation window and alternatively, the world can be written to a C++ file. This C++ file will make it much easier to use the Virtual Worlds elsewhere, or perhaps directly merging the code with other source code to build special purpose Virtual Worlds, Games, etc. . .

Figure 3.3 displays a screenshot of the interactive tool. The tool described here will be the workbench into which we will implement possible solutions to the problems introduced in chapter 1.



```
19:19:59 : Java Virtual Machine activated...
19:20:04 : Domain loaded from file:///C:/vrdomain.daml
19:20:11 : Mapping loaded from file:///C:/vrrepre.daml
19:20:16 : Instances loaded from file:///C:/vrinstances.daml
```

Figure 3.3: Screenshot of the Interactive VR Generation Tool: OntoWorld

3.6 Summary

In this chapter we presented the approach used in this thesis for developing Virtual Reality. A high-level description of the world is translated into a working Virtual World by means of a two step process. For this high-level description we are using an ontology. The approach allows the user to generate Virtual Environments in a very intuitive manner, without the need of any specific, area related, skills as it is the case with the conventional approaches.

Earlier, the first prototype has been expanded to a class-instance based type of system. This allows to create several instances for the same class. However, this is not enough, the system still has to be improved further on. There is a need for representing more advanced worlds.

Chapter 4

Part-whole Relationships

4.1 Introduction

The commonsense notion of the part-whole (part-of) relation has been studied in many fields like philosophy, cognitive science, and artificial intelligence. In psychology, it is a known fact that people, when describing an object they usually do this in terms of the parts of the object. When asking someone to describe a house for example. Most of the times we would get back an answer where a house is being defined as an object with four walls, a roof on top of it, a door and some windows in it. A hierarchical structure with objects having parts and those parts having parts on their own is the most intuitive way for human beings to make a representation of compositions of objects. Treating many objects as one object reduces complexity in understanding.

The part-whole relations are therefore our first important aspect in describing more complex objects in a Virtual World. In the field of philosophy, the *part-of* relations are classified into seven types [23]. These different kinds of *part-of* relations and hence with different meanings allows us to make a distinction in interpretation. In the first place, these distinctions are mostly regarding to the inheritance of properties of the whole in the parts. In the second place, we can also make distinctions with regard to the constraints the whole imposes on the parts. This is in contrast to the regular drawing tools in which there is no meaning attached to the *part-of* relation. Something is just a part of another thing, nothing more and nothing less.

First, in section 4.2 a definition of the general *part-of* relation is given. This general relation is being divided into subtypes of relations based on differences in their meaning in section 4.3. Afterwards, in section 4.4, relations, which could be easily confused with some of the part-whole relations are explained. In section 4.5, we discuss that, in order to model a composition, it is not enough to just mention the kind of relation which holds between the objects in question. Some additional relationships or constraints will have to be stated too. Finally, this chapter is summarized and evaluated in section 4.6. The part-whole relations that are useful for modelling Virtual Reality will also be identified here.

4.2 What are Part-of relations?

Theories of parthood (mereology, from the Greek word "meros", which means "part") relations have been studied for ages. The *proper-part-of* relation introduced by the Classical Extensional Mereology (CEM, [29]) is the most basic and most intuitive mereological concept. It is a binary strict ordering relation and hence is transitive and asymmetric on the domain. Thus, it follows that the relation is irreflexive and antisymmetric as well. That is, nothing is part of itself (irreflexiveness), if one thing is a proper part of another, then the second is not a proper part of the first (antisymmetry) and if one thing is a proper part of a second thing, and this second thing is a proper part of a third thing, then the first is a proper part of the third (transitivity).

The relation of proper part to the whole is a strict partial ordering but this does not mean that all strict partial ordering relations are therefore *part-of* relations. They also have to satisfy some additional principles:

Extensionality Two individuals are identical if and only if they have the same parts. Two composites with all their parts in common are said to be identical.

Principle of Sum There always exists the individual composed by any two individuals of the theory meaning that two different parts of an entity can be considered together as one part, different from either component.

Supplementation If an individual x is a proper-part-of an individual y, then a different individual z exists which is the missing part from y. So an individual that has a proper part needs other parts in addition to supplement this one to obtain the whole.

However, these properties and principles can be criticized. There are cases for example, in which we use the *part-of* relations and where the transitivity property does not hold. We will see this in the next section. There is also no guarantee of the existence of sum-individuals and the criterion saying that individuals having all parts in common are identical is generally false. Therefore, some minimal requirements have been stated by [29]. The *part-of* relation has to have an asymmetry in its proper parts, transitivity in its proper parts and must be conform to a weak form of the supplementation principle. This weak supplementation principle states that if an individual has a proper part, it has a proper part that is disjoint from the first proper part.

4.3 Kinds of part-whole relations

Incorporating and relying on only one general part-whole relation, to represent aggregated objects or to describe the structure of composite objects, according to [39] and as is described above, is a too restrictive way to consider physical reality. Modelling the real world, either physically or virtually, needs more than just the ability to say that one object is part of another object. This shortage of part-whole relations is a known, and a many times confronted problem, in various fields where there is a need to model artefacts like machines in engineering, buildings in architectural fields and human beings in the biological area. It has been recognized that several different kinds of (subtypes of) the *part-of* relation should be a useful resource for Virtual Reality or 3D modelling.

In [28] an extensive argumentation is given about why there is a need to have a more elaborate set of part-whole relations in building interactive 3D-models of human anatomy. When modelling the gyrus, a part of the human brain, we can distinguish the following relations:

1. gyrus x *part-of* gyri
2. gyrus x *part-of* lobus y
3. gyrus x *part-of* gray matter
4. gyrus x *part-of* visual system

All these relations have different meanings of the *part-of* relation. In the first some gyrus x is defined as a part of a collection, the second says that same gyrus x is a morphological demarked building block of lobus y, the third meaning describes the gyrus as a portion of the gray matter and finally in the fourth relation it has a functional meaning: gyrus x is a functional part of the visual system. We want to stress that with the gyrus denoted by 'x' in all four relations means that we are dealing with one and the same gyrus involved all four relations. Thus, knowing the exact meaning of part-whole relations is of crucial importance, especially when inheritance or inference is involved. This is only one field that we have considered but the arguments can be easily expressed in other fields as well.

Apart from the more intuitive argument for having different kinds of *part-of* relations we could also follow the algebraic path. One of the most important properties of the *part-of* relation is its transitivity. We can say for example that a finger is a part of a hand and a hand is part of the arm thus hereby saying that a finger is part of the arm too. However, various people [23] have discovered that when modelling real world situations this property does not always hold anymore. If we take for example the fact that an arm is part of a musician and the musician is part of an orchestra.

Now, is it right to infer that the arm is part of the orchestra too? This would be an awkward thing to say. So there must be a number of different kinds of part-whole relations and using multiple kinds intermixed will not keep the transitivity property valid, the transitive property only holds when the same kind of part-whole relation is used throughout the premises.

These subrelations of the general *part-of* relation are subject of many investigation and discussions and still there are some open questions [27]. What is the interrelationship between these part-whole relations; is there some kind of hierarchy or are they all independent from each other? How do they interact with each other if one is used in combination with the other? We have already said that when using the same kind of relations, the transitivity property stays valid but what happens with the existing properties when using combinations of different kinds of relations or what happens when new properties arise? A last question is whether the collection of relations presented is complete? Are there maybe other kinds of relations that differ in some way from the relations already proposed in the literature?

From linguistic and cognitive studies, [23] proposed a classification of the part-whole relations overcoming the problems as described earlier in this section. The different ways in which parts could be composed to form a whole are divided on the basis of three properties:

- Configurability or functionality
  Some part may have a particular functional or structural relationship that restricts their spatial or temporal location inside the whole.
- Homogeneity
  Homogeneous parts are the same kind of things as their wholes while non-homogeneous parts are different from their wholes.
- Separability
  Separable (variant) parts are parts that can be separated from the whole while inseparable (invariant) parts can not.

Combinations of these properties will lead to seven different kinds of part-whole relations, which will be discussed next. Medical fields, like we have already mentioned above, where knowledge based applications are extensively used, have made more elaborate sets of part-whole relations specifically for their domain as is described in [1]. Here, we will restrict ourselves to the seven basic composition relations as described by [25][3][23].

4.3.1 Component-integral object

A component-integral object relation is the most common form of composition relation. The whole or the integral object is composed of component parts, which are objects in their own right. The components have a particular relationship to one another and to the whole. They can be removed from the integral object (e.g. wheels are part of a car, scenes are part of a film).

4.3.2 Material-object

The material-object relation or stuff-object relation is a kind of component-integral object relation with the difference that in this kind of relation the parts can not be removed from the whole because the relationship between the part and the whole is not really known. These relations thus define what objects are made of (e.g. a car is partly iron, bread is partly flour).

4.3.3 Portion-object

The portion-object (portion-mass) relation differs from the first two relations above in the fact that here the parts are the same kind of thing as the whole when in the above relation they were different from each other and the whole. This similarity allows us to inherit many more properties of the whole than with the above relations (e.g. this slice is part of a pie, a meter is part of a kilometre).

4.3.4 Place-area

In the portion-object relationship, the various similar pieces, or pieces of the same kind, of the whole could be removed but in the place-area relationship the places can not be removed from the area they are part of. It is usually defined between geographic places and particular locations within them (e.g. Belgium is part of Europe, a peak is part of a mountain).

4.3.5 Member-bunch

All the relations above had some kind of structural or functional requirement on the parts in between or inside the whole. When we drop this requirement, the only requirement that is left is the fact that a part has to be a member of some kind of collection and hence should not be confused with the regular classification relationship (e.g. an employee is part of a union but is not a union on its own, a tree is part of a forest but one tree is not a forest).

4.3.6 Member-partnership

Members in a partnership are a bit more restricted than just members in a bunch in the fact that the member-partnership relation is an invariant form, the members can not be removed from the partnership without destroying the partnership itself, so like a component the member plays a functional role and is not separable (e.g. Stan Laurel is part of Laurel and Hardy).

4.3.7 Feature-activity

A last type of relations is indicated by the use of ”part” to designate the features or phases of activities and processes. This relation is similar to the component-integral relation. In the same way as integral objects are made up of components, complex activities are structured by combining subactivities. The part plays a functional role in the whole and can not be detached from it (e.g. Dating is part of adolescence).

4.4 Non-compositional relations

Besides the basic composition relations we discussed above there are also a number of relations which could also in some way be interpreted as *part-of* relations [25].

4.4.1 Classification

The extension of a concept is defined as the set of objects to which the concept applies. If we have an instance of an object like a book for example we can say that the instance is part of the set of objects to which the book concept applies. So in this case we could confuse the classification relationship with the member-bunch relationship. This class-instance structure has been added to the system as part of my apprenticeship as was explained in chapter 3.

4.4.2 Attribution

The attributes of an object (instance of an object) could also be confused with some kind of composition relation because the object is actually composed of different properties that represent all the knowledge we know about an object. For example, the height and weight of the book can be considered as part of the properties of the book in question. However, the properties are not components of the object itself but it does give us the ability to create more complex objects, an increase in properties will increase the complexity of the object. Adding this was also a part of my apprenticeship as explained in chapter 3.

4.4.3 Attachment

Finally, we will consider the attachment relations. When two objects are attached to each other it could mean that the one object is a part of the other, but this relation does not always hold. If a phonebook is attached to the phone booth with some kind of string it does not mean that the phonebook is a real part of the phone booth. Therefore, the attachment relations describe another kind of complexity of objects as the one that was explained in the introduction.

4.5 Vertical and Horizontal relationships

To clearly model complex objects as a composition of various parts it is not enough to just describe their structure by means of the composition relations we just mentioned. We should somehow also be able to express how this whole is related to the parts and how the parts are composed together to form the whole. In [3] there were two kinds of relations that can be distinguished:

- Vertical relationships
  The vertical relations can be subdivided into two kinds of relations. The first kind covers the dependence relations that express the dependency of the existence of the whole and the existence of the parts within the whole and also the reverse. According to [29] a part is dependent on the whole if the former can not exist unless the latter exists. As an example, consider the relation between a person and its brain: if we change the brain, we cannot assume the person to be the same any more. The second kind of vertical relations are the dependence relations that express dependencies between the properties of the whole and the parts and vice versa. Here we have the example that the weight of the part is always less than the weight of the whole.

- Horizontal relationships
  The integrity or wholeness of an object comes in degrees. An object is more integrated when it is one continuous piece then when it is a collection of pieces scattered all over. A country is more integrated when it is one piece of land instead of several different islands. These horizontal relations, or integrity relations, are the constraints between the parts that characterize the integrity of the whole.

4.6 Summary and evaluation

In this chapter we described the part-whole relations as a way to describe the composition of component parts to form a whole, a complex structure. The part-whole relations are of great importance because they enable us to manipulate a group of objects as if it were one object.

Figure 4.1 gives us a graphical representation of the model of the meronymic relations described in this chapter. The part-whole relation is expressed as a ternary relation. Two objects are related to each other by means of some kind of part-whole relation which has a type belonging to one of the values as can be seen in the figure.

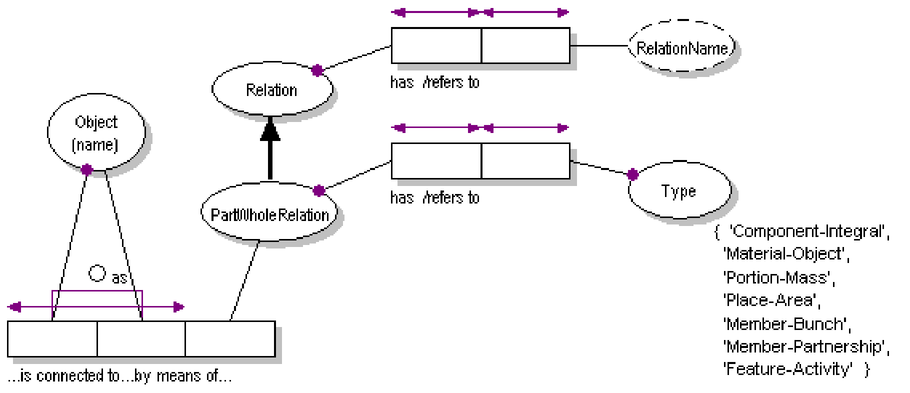

RelationName
Relation
has /refers to
Object (name)
has /refers to
Type
PartWholeRelation
as
...is connected to...by means of...
{ 'Component-Integral', 'Material-Object', 'Portion-Mass', 'Place-Area', 'Member-Bunch', 'Member-Partnership', 'Feature-Activity' }

Figure 4.1: Graphical representation of the model for the part-whole relations

However, from the different kinds of part-whole relations, not all are of equal concern for our purpose. Making some differences in part-whole relations infers differences in the inheritance of properties in the part from the whole. This could allow us to describe objects with the same characteristics as the other objects in the group it belongs to. The differences in part-whole relations also create constraints on the parts that are imposed by the whole.

The first and most general part-whole relation is the component-integral relation. This kind of relation will be used to describe compound objects in the Virtual World. Compound objects being one of the most frequent structures in real life makes this kind of relation a very useful one. The material-object relation does not really contribute to the hierarchical structuring of the Virtual World. Maybe in later phases this kind of relation will become more important. Instead of using the material-object relation, attributes will be used for this. Both the portion-object and the place-area relations are relations for describing compound objects but they are not used as much as the component-integral relation. However, they are also useful in some cases. The portion-object relation is useful for describing pieces of an object that was once one whole object. The pieces are separable from the whole. The place-area relation will be used a lot to define specific locations inside the Virtual World, or part of the Virtual World. The member-bunch and the member-partnership relations are very important relations. These will be used to define collections of different objects that are not attached to each other. The difference between the two relations is that with the member-bunch relation the parts can be separated from the whole and with the member-partnership relation they can not. The last relation, the feature-activity relations will not have much use. There is no need for concepts like "features" or "phases" or "processes" when defining a Virtual Environment, they are not useful to serve as a hierarchical structuring mechanism.

The dependency relations and integrity relations are for the moment a bit far fetched for describing complex objects. It is more like a philosophical question. For example, is it still the same object if I tear down one part from it? These kinds of relations are not having much effect on describing a static world but in later stages, when the worlds are becoming dynamic, they could have some importance.

Chapter 5

Spatial Relationships

5.1 Introduction

The physical world is made up of objects, each with a well-defined shape and a position in space. People construct some kind of mental model in order to talk about the objects and their relations in the world. This is done by means of spatial prepositions (relations). The semantics of spatial relationships have been intensively studied in all the disciplines of cognitive science, linguistics and even AI. In the latter, the major fields were spatial reasoning, image understanding and language processing. The fundamentals of the spatial relations, and the spatial prepositions in particular, are described in [20].

The spatial relations play an important role in connecting visual and verbal space. For a human being to be able to reason about space or even to let robots or mobile agents reason about space [31]. In contrast to the numerical approaches (e.g. CAD/CAM) where all the objects are placed on an exact location in the environment, the spatial relations provide us the intuitive way we are looking for.

First, we will give a definition of a spatial relation in section 5.2. The class of spatial relations distinguishes between distance, orientation and direction by means of respectively topological, orientation and directional relations as will be seen in section 5.3. The orientation relations and directional relations require an additional reference frame to be able to unambiguously specify the correct location of the primary object from the location of the reference object. This is discussed in section 5.4. Next, a graphical representation of the model of a spatial relation is given in section 5.5 together with a summary and an evaluation.

5.2 What are spatial relations?

First, we will restrict ourselves to the general case of a spatial relation. Spatial relations can be defined by specifying conditions between objects such as the distance or the relative position. In this sense, the spatial relations characterize a class of relations denoting object configurations. Strictly speaking, they represent relations between spatial entities. Special purpose spatial relations will be discussed in the following sections where the relations will be categorized.

A spatial relation is an n-ary relation, a ternary at least. In order to establish a spatial relation, we first of all need a located object (LO), also called the primary object. This is the object we want to localize with respect to one or more other objects, different from the first. These are often called the reference objects (REFO). At least we need one reference object but there could be more of them in special relations (e.g. for the "between" relation, where some object is in between two or more objects). We also need the type of relation that gives it a specific meaning, apart from just being related. Types of relations will be discussed in the next section. Another thing that has to be specified, is the reference frame. We will see more about reference frames in section 5.4.

5.3 Categorisation

5.3.1 Topological relations

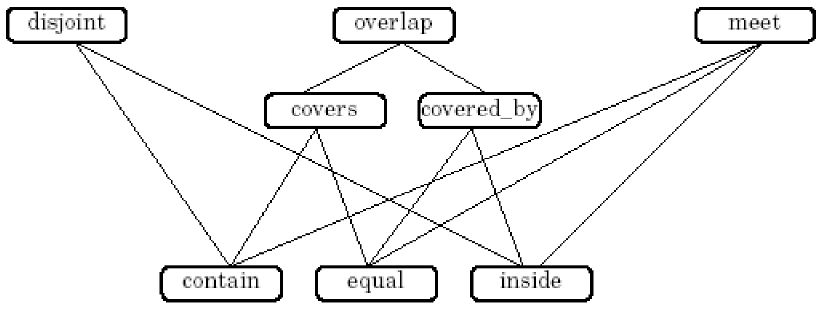


Figure 5.1: Topological relations

Topology is one of the fundamental aspects of space. A binary topological relation between two regions embedded in a three-dimensional space is based upon the comparison of one regions' interior, boundary and exterior with the other regions' interior, boundary and exterior. In other words, these are relations that describe neighbourhood and incidence. See figure 5.1 for an overview of the basic topological relations [26]. Many approaches have been developed that describe the topological relation. The Region Connection Calculus (RCC) defines a formalism to capture spatial relationships based upon binary topological relations between spatial objects [8]. A set of eight jointly exhaustive and pair wise disjoint base relations has been created. This calculus is also called the RCC-8 calculus. An alternative approach is the n-intersections approach [8]. In most n-intersections calculi, three sets of points are associated with every region, interior, boundary and complement. Afterwards, the relationship between two regions can be characterized by a 3x3 matrix, each of whose elements denotes whether the intersection of the corresponding sets from each region are empty or not. This model is also called the 9-intersection model.

5.3.2 Orientation relations

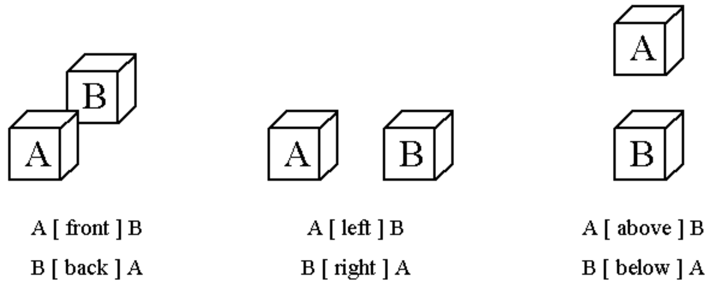


Figure 5.2: Orientation relations

By orientation we mean that the objects are placed relative to one another. The relations considered here are *in front of*, *behind*, *left*, *right*. Others are *above* and *below* (see figure 5.2). If there is a need to have more specific relations, the basic orientation relations can be combined to form other relations. The *left*, *behind* and *above* relations could be combined for example to form the *left-behind-above* relation. This relation will describe an object that would be positioned "left" and "behind" and "above" another object. There is one relation not mentioned before, and that is not displayed on the figure, namely the *between* relation. This relation describes an object that is positioned "between" two other objects. The *between* relation needs multiple reference objects. Unlike the topological relations, orientation relations are not binary relations. When people talk about two (or more) objects in the space they automatically use some kind of reference frame to do this. This is needed to infer the correct position of the primary object. A primary object can be on the "left" of the reference object for one person but it can be on the "right" of the reference object for another person. This person would then be standing on the other side of the object facing the first person. For yet another third person, the primary object could be "behind" the reference object. Therefore, using some kind of reference point and reference direction from which the objects are seen is needed to clear out these problems.

5.3.3 Directional relations

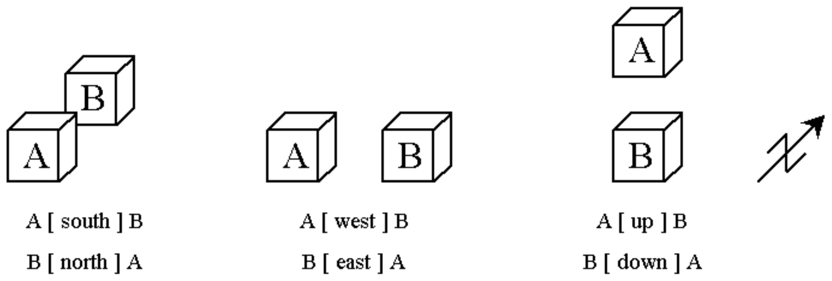


Figure 5.3: Directional relations

Geographic Information Systems, and geography and cartography in general, deal with large-scale space, space whose structure is at a significantly larger scale than the observations available at an instant [15]. Directional relations by means of cardinal directions (*north*, *south*, *west*, *east*, *north-east*, *north-west*, *south-west*, *south-east*), as can be seen in figure 5.3, are usually used for this purpose although some cultures use these absolute relations in every day life. The cardinal directions can be represented in the same way as the orientation relations. The difference with the orientation relations is that all the directional relations rely on one single reference direction for the entire space. The cardinal directions are then defined as some kind of landmarks, with a global reference frame, defined by the movement of the sun [7]. When people talk about objects in the space using the directional relations, we will not encounter the same problems as with the orientation relations. No matter from which viewpoint the objects are seen, the relations between the objects will remain the same.

5.4 Reference Systems

If there is a need to relate multiple objects to each other by means of spatial relations not only the objects involved and the relation itself have to be made explicit but also the reference frame has to be given in order to be able to locate an object in relation to the other objects. There has been extensive empirical research in the cognitive linguistics area exploring different reference systems used to describe spatial situations across different cultures. After having given the definition of a spatial reference frame, the kinds of reference systems are classified. The reference systems can be divided around two properties. The first property is their viewpoint as described by [7] and [16]. The second property is their use in the system as formulated in [31][2].

5.4.1 Definition of a Reference Frame

One way to think about frames of reference is to attach a three-dimensional coordinate system to the object we are talking about. This coordinate system can be derived from the properties of the object itself (e.g. a car has a specific front, left, top,...). We are then able to define the x-axis as the left-right axis of the object. The negative region represents the left of the object while the positive region represents its right. Doing the same with the y-axis describing the front and back of the object and the z-axis describing the above and below region of the object. We now have full capability of defining every object relative to the object that has been given the reference frame.

It is generally observed that people are capable of transforming their perception of space into the perception of space from a different point. This is called talking in perspective. It is fundamental to understand other people's description of space from their perspective. Thus, humans sometimes tend to refer to locations of objects in some other kind of reference frame. This frame of reference is always specified by three characteristics in order to be able to specify the axes system: an origin, an orientation and a handedness. We first have the origin of the coordinate system that can be either a point within the object itself or a point outside the object. In our case, the origin will mostly be the position of the speaker, the one who is talking about the object. The orientation of the coordinate system that is given by the direction to which the speaker is looking at. Knowing the front of the speaker, we already have one axis of the reference frame. The second axis has to be specified by determining the left and right of the speaker. The handedness of the coordinate system describes in what way both axes (front-back and left-right) are related to each other. This can be either a left-handed system which anti-clockwise follows: front, left, back, right, front or a right-handed system which anti-clockwise follows: front, right, back, front (see figure 5.4). In the right side of the figure we can see a right-handed axes system, on the left we can see a left-handed coordinate system. We now know the second axis of the references frame. The third axis can be found using the following rule of thumb. When you hold your right hand so that the fingers curl from the positive x-axis towards the positive y-axes, your thumb points along the positive z-axis. This rule can be done in the same way for the left-handed system.

To clearly understand a spatial relation all three above characteristics are needed although all of them can be replaced by default values. The origin could be, for

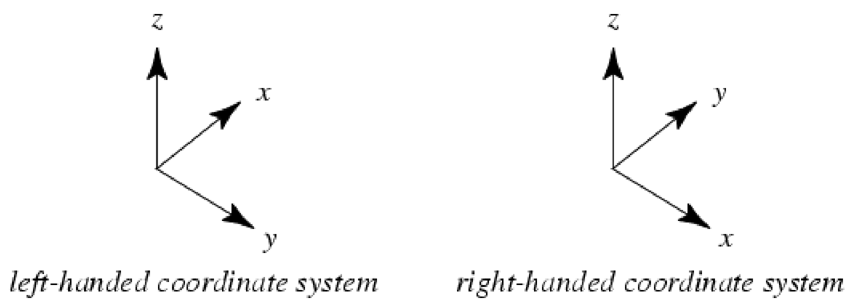


Figure 5.4: Handedness of the Reference Frame

example, the centre of the reference object. The orientation could also be taken from the object itself and the handedness may be replaced by the conventional right-handed system.

5.4.2 Classification based on viewpoint

Relative Reference Systems

The first type of system determines reference by dividing the world into regions that are extensions from the body of the speaker (see figure 5.5). These regions necessarily shift as the speaker shifts the alignment or location of his or her body. That is, they are perspective dependent. The system is invariant under translation or rotation of the full configuration (speaker, ground, and figure), but not under rotation or translation of either speaker or ground. This has to be taken into account when reasoning about the spatial environment. Such systems are called speaker-relative or just relative reference systems.

Absolute Reference Systems

The second type of reference system is an absolute or perspective independent system of spatial reference. There are many different types of absolute systems which all describe the relationship of two locations in such a way that the position of the speaker and hearer are irrelevant (see figure 5.6). The system will not be rotated with a rotation of the front of the reference object as is the case with the relative system. They are however not invariant under rotation of the whole configuration (the figure and the ground). The absolute systems make reference to commonly agreed upon directions, which will always remain constant throughout the world commonly encountered by the speakers making spatial reasoning much easier.

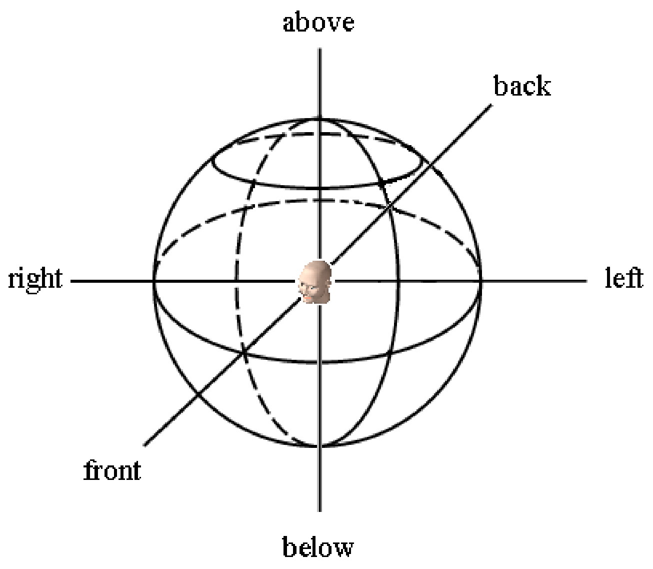


Figure 5.5: Relative reference model

5.4.3 Classification based on use

Intrinsic reference frame

The orientation is given by an inherent property of the reference object. An inherent property of an object can be given by perceptual organs (of humans and animals), by the characteristic direction of use or movement (e.g. the front of a car) and by functional properties (e.g. of the two goals in the soccer domain).

Extrinsic reference frame

The orientation is determined by the position of a possible imaginary observer, i.e., it is given by contextual factors, such as the accessibility of the reference objects or objects in its vicinity (e.g., "Looking through the window, the red box is behind the black box", or the actual direction of movement of a vehicle).

Deictic reference frame

In the extrinsic case, the observer is spatially separated from the speaker. When the observer and the speaker are one and the same, they are spatially in the same place, we say the reference frame is deictic (e.g. "From my point of view, the red box is behind the black box").

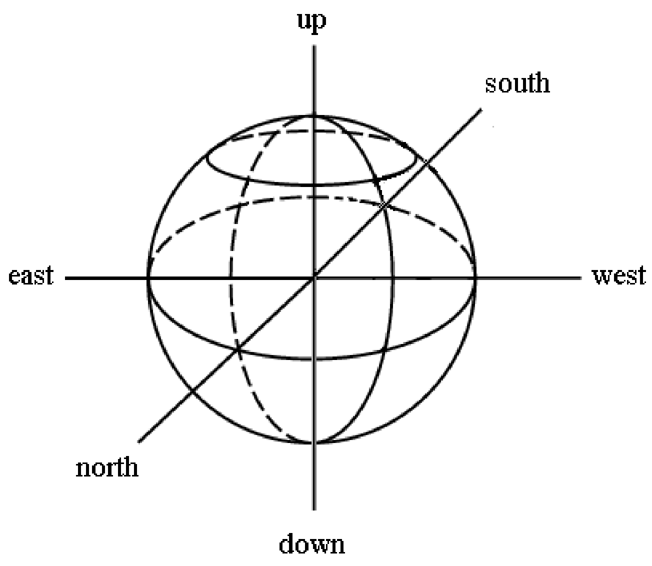


Figure 5.6: Absolute reference model

5.5 Summary and evaluation

In this chapter we discussed the spatial relations. The spatial relations enable us to intuitively describe a spatial situation, a world consisting of objects that are not connected to each other.

In figure 5.7, a graphical representation of the model for the spatial relations is given. We can see that the spatial relation is an n-ary relationship. It needs a primary object, the one that has to be located, a number of reference objects used for locating the primary object and the type of relationship. We can define three rules here:

rule 1 each *Topological relation* is a *spatial relation* with type disjointWith, overlapWith, meetWith, equalWith, coveredBy, containedIn or insideOf.

rule 2 each *Direction relation* is a *spatial relation* with type northOf, southOf, up, down, westOf or eastOf.

rule 3 each *Orientational relation* is a *spatial relation* with type infrontOf, backOf, belowOf, aboveOf, leftOf or rightOf.

First we discussed the topological relations, the relations that speak about the neighbourhood and incidence of an object with respect to another object. A pitfall in the topological relations area is the fact that they are much more useful for modelling in 2D and in lesser form in 3D. However, we can expand the topological relations to

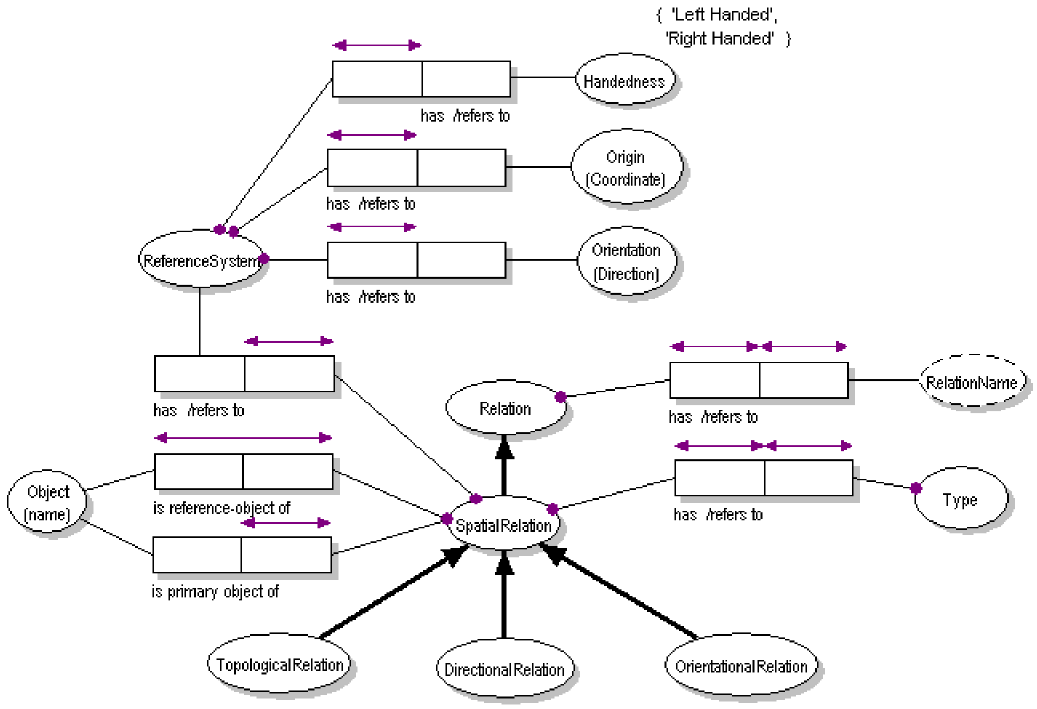


Figure 5.7: Formal representation of the model of spatial relations

be useful for 3D. Then, an object could be *disjoint* from another object meaning that the object is completely outside another object. The reverse relation, *inside*, describes an object being positioned inside another object. A second disadvantage is that the topological relations are widely used in qualitative spatial reasoning, so reasoning about already existing worlds, but are far too abstract to use in the construction of complex Virtual Environments (what we are trying to accomplish). A relation "Object A is disjoint with object B" would not make sense when building a world but could be useful as a constraint on objects A and B. However one topological relation could be useful for incorporating into the system, namely *touches* in which two objects have one point or plane in common. In combination with orientation or directional relations we could derive an exact location. The other topological relations that are not really useful for building Virtual Worlds could be used as constraints on the objects in the Virtual Environment.

Next the orientation relations with *in front of*, *behind*, *left*, *right*, *above*, *below* and the directional relations with *north*, *south*, *west*, *east*, *up*, *down* are the most important ones discussed. For the purpose of our system it is not really necessary to make a finer distinction then stated here when talking about the orientation relations respectively directional relations. A more precise localization can be given by combining multiple relations with each other which will give us all the expressive power we need for the purpose of the system without much extra need for computational power because most of the time combinations of not more than three relations are used, one for each direction. Combining multiple relations of the same direction with each other would make no sense.

We would like to note that the above relations delimit a region of space relative to the object itself. This is good to infer the location of an object relative to one another when reasoning about the world but it is not when planning to build the world. Therefore we have to make a simplification and presuppose that with e.g. *in front of* we actually mean on one line, following the front direction. Similar simplifications are needed for all the relations described above. It will be a rather big restriction on the possibility on placing an object relative to another, but with combinations of the relations we can come to a number of locations that will satisfy our need in fast prototyping a Virtual Environment.

We also discussed different kinds of reference frames in this chapter. However, we are not yet able to make use of all these reference systems. For now, the objects we are able to visualize in the Virtual World are primitive geometrical objects (e.g. a sphere, a box, a cylinder,. . .). These objects do not have properties, like a front for example, from which we can derive a reference frame. Therefore, the relative references systems can not be used in our system. The absolute reference systems do not have these kinds of problems and can therefore be used for our purpose.

Chapter 6

Solution

6.1 Introduction

So far, we have introduced the different kinds of relations we would like to use in our system to model Virtual Worlds by means of ontologies. To be able to generate code from the specifications using these relations, we need to give them a precise definition (i.e. define their semantics). For defining composed or complex objects, we need to have explicit definitions of the part-whole relations. To be able to specify the location of objects relative to other objects we also need exact definitions of the spatial relations. These definitions are given in this chapter.

This is done in the same order as we have introduced them. First, we will give the definitions of the different part-whole relations (section 6.2). A definition of a reference frame is given in section 6.3. The orientation relations, directional relations and the topological relations are defined respectively in section 6.4, 6.5 and 6.6. Afterwards, these definitions are moulded into an upper ontology. This upper ontology is discussed in section 6.7. Next, a summary of this chapter is given in section 6.8.

6.2 Part-whole relations

In chapter 4 we discussed the part-whole relations. Section 4.3 described the subtypes of the general part-whole relation. These subtypes were made on the basis of the characteristics of the relation. Each of the part-whole relations have some differences, or in other words, each is identified by a combination of several properties. For defining the part-whole relations we will use a theory of semantic relations called the Relation Element Theory [23]. This theory says that each relationship (or relation) may be regarded as a composition of a set of simpler relation elements. Relations are no longer viewed as unanalyzable primitives, but as the set of relational properties that distinguish them.

Definitions

1. $xRy \rightarrow (Ea...En)$
2. $xRy \rightarrow (E1(E2))$

The first definition defines a semantic relation (R) between two concepts (x and y) as a complex structure composed of one or more primitive relation elements (Ea,. . . ,En). The relations may share one or more elements and the greater the proportion of elements two relations have in common, the more similar they are. The relation elements may also be hierarchically organized in such a way that one dependent element (E2) can only occur when another independent element (E1) is present, as is given by the second definition. This is what we need to define the part-whole relations because each part-whole relation is a composition of properties.

With the two definitions above we are able to construct the definitions for the part-whole relations we discussed in chapter 4. The central idea of the part-whole relations is a *connection* of the whole and its parts. This connection differs in three ways, on whether the part is *functional*, *homogeneous*, and *separable*. Variations in these elements are responsible for the different types of part-whole relations. Expressing the first part-whole relation, the component-integral relation, would result in the following definition.

Definition

$$xCIy \rightarrow (Connection, (Functional, Nonhomogeneous, Separable))$$

As we have seen in section 4.3, the component-integral (CI) relation was described as a relation that has parts that support a functional role with respect to the whole. The parts are not similar to the other parts or to the whole and they are separable from the whole (see table 6.1).

Table 6.1 describes all the five types of part-whole relations that we said to be important for modelling Virtual Environments. Each type is given together with its relation elements. These elements match the description of the relations we have given in section 4.3. The Functional column in the table means that the parts in the relations play a functional role (+) with respect to the whole or not (-). The Homogeneous column tells us whether the parts are homogeneous (+) or not homogeneous (-) to each other and to the whole. Whether the parts can be separated (+) or can not be separated (-) from the whole is given in the Separable column.

Relation	**Relation Elements**		
	Functional	**Homogeneous**	**Separable**
Component-Integral	+	-	+
Portion-object	+	+	+
Place-area	+	+	-
Member-bunch	-	-	+
Member-partnership	-	-	-

Table 6.1: Part-whole Relations with their characteristics

With these relation elements we can now build the definitions of all the part-whole relations in a similar way as we did for the component-integral relation above. Due to the similarity we will not explicitly write down the definitions of the remaining part-whole relations.

6.3 Reference Frame

An object's location may be given in function of, or relative to, some other object, called the reference object. A chair for example could be "before" a desk. In this example the desk plays the task of reference object. The object we want to localize is the chair. In another example we could say that the chair is "before" someone standing inside the Virtual World. In this case that person is used as the reference object. For being able to derive the position of the object we want to localize, several things are assumed to be known. First of all, we need to know the place or location of the reference object. Next, we need an understanding of the concept direction (what is north, east, before, right-of, . . .). Finally we need some kind of unit measure to express the distance between two objects. Besides the reference object, the reference frame will provide all the information we need to derive the location of the primary object from the location of the reference object.

We will now give a formal definition of a reference frame. For this we use mathematics (see figure 6.1). A reference frame always has a point called the origin. Besides the origin three other points are necessary to establish a reference frame. Notice that these four points have to be in such a position that they form three planes that intersect each other in three lines. One line will be labelled the X-axis; another one the Y-axis and the last one, the Z-axis. Next to this, we need unit measures. These will be expressed in terms of basic vectors or unit vectors along all three axes [32]. These unit vectors have a length and a direction representing respectively the unit measure and basic directions.

In the following three sections, a coordinate system will be designated by a symbol

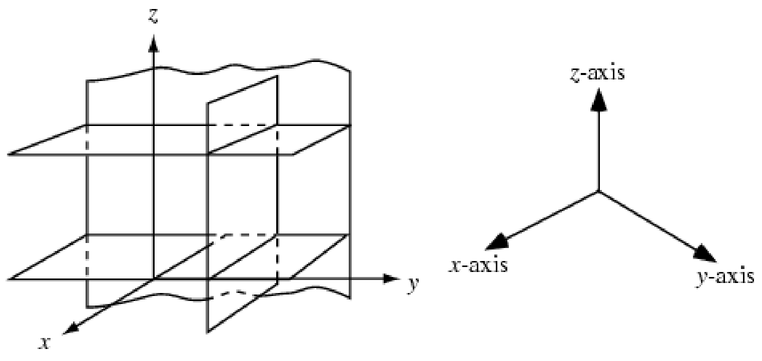


Figure 6.1: Euclidian Reference Frame

F with a subscript for the intended name of the reference system. Axes are labelled X, Y and Z with the appropriate subscript. The unit vectors along X, Y, and Z will be denoted by **i**, **j** and **k** respectively. They are directed line segments from the origin to the points (1,0,0),(0,1,0) and (0,0,1) respectively. After we have assigned the basic directions to the axes of the coordinate system we are able to define the relations.

Sections 6.4, 6.5 and 6.6 gives the definitions of respectively the orientation relations, the directional relations and the topological relations. Table 6.2 gives an overview of the reference frames we will define in those sections. For the orientation relations we will define a reference system named "Fundamental Reference Frame". The directional relations will be defined in a reference system named "Global Reference Frame". Both reference frames are absolute reference frames. As we have said in chapter 5, we can not yet use the relative reference systems because the objects we are able to visualize in the Virtual Environment do not have intrinsic properties like a front, a left,.... The difference between both reference frames we will define, is that the one is defined inside the other. The "Fundamental Reference Frame" serves as a kind of universal reference frame. A reference system that represents the universal coordinate system of the world. The "Global Reference Frame" is defined inside the universal "Fundamental Reference Frame". The topological relations are independent of any reference system as we will see later. We will not have to define a reference frame for this kind of relations.

6.4 Orientation Relations

6.4.1 Fundamental Reference Frame

When people are describing a world or scene they mostly do so from a kind of virtual view-point from which they can overlook the entire scene. Therefore, this viewpoint

Relation	**Reference Frame**	**Type**
Orientation	Fundamental reference frame	Absolute reference frame
Directional	Global reference frame	Absolute reference frame
Topological	***	***

Table 6.2: Reference frames for the different spatial relations

is outside the world described. This is a "naive" [20] view, in the sense that it appeals only to a common sense conception of space and objects. This is actually an objective or a 'viewpoint-independent' view of the world. It is not a view of how things might appear to someone standing inside the world, but it is a view of how things appear to someone standing outside the world (an external observer). This observer can look to the world, which is extended in all directions, with an "all-seeing" eye as illustrated in figure 6.2. In this view, space is three-dimensional, isotropic and Euclidean. The space has a universal reference frame, one that is used by someone outside the Virtual World and who can see this Virtual World entirely at once. In the remains of this work we will call this reference frame the Fundamental Reference Frame, denoted by F_F. It is supposed to be available at all times. With this reference frame we will associate *orientation* relations (before, after, right, left, above and below), which can be used to indicate the position of objects in the world.

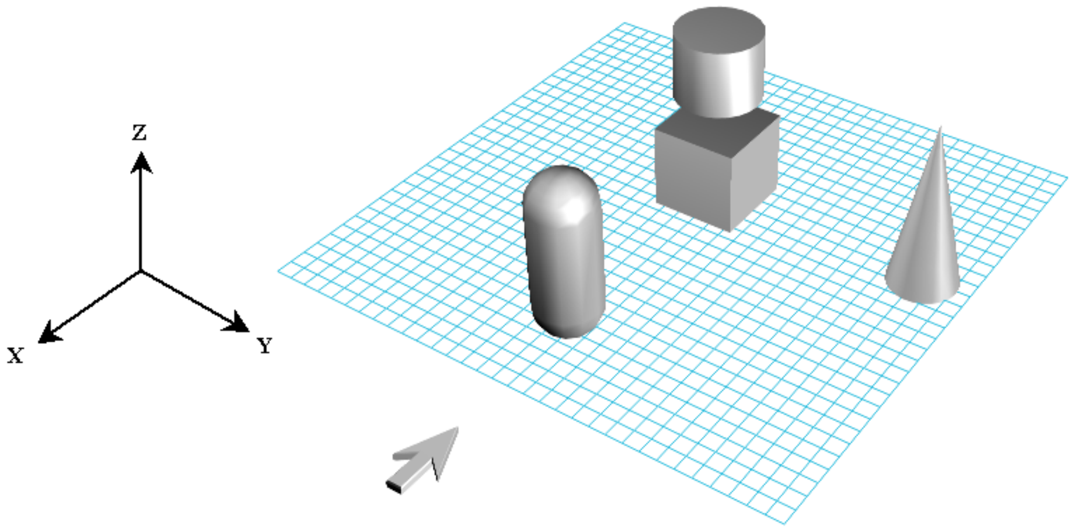


Figure 6.2: Fundamental Reference Frame

The left side of figure 6.2 displays the reference frame. Once we have this reference frame we can specify directions. In our case we will say that the basic vector $\mathbf{i}_F$ which runs along the X-axis represents the *before* direction. The *right* direction will be connected to the vector $\mathbf{j}_F$ (along the Y-axis). The *above* direction runs parallel with the Z-axis and is described by the $\mathbf{k}_F$ vector. The reverse vectors represent the *after*, *left* and *below* directions respectively. These assignments are just a matter of personal taste. We could as well have made another choice.

6.4.2 Definitions of the basic relations

Since we have the reference frame we have all we need to be able to define the orientation relations. Expressing the relations in terms of the unit vectors corresponding to the basic directions is now an easy and straightforward job to do.

We will start with the *Before* relation. Take figure 6.2 as an example situation. Here, the capped cylinder is said to be "before" the cube. In general, an object A is "before" another object B when it has a larger x-value. So drawing a half-line beginning at the position of object B and in the *before* direction would describe all possibilities of an object A being "before" object B. Or in other words, when object A is positioned on that line we could say it is "before" object B. In mathematics we would write the following.

Definition

$$Before(A, B, d, F_F)$$
$$\Updownarrow$$
$$A = B + (di_F + 0j_F + 0k_F) \text{ with } d \in IR_0^+ \text{ and } A, B \in IR^3$$

The "+"-operator here is specified as the algebraic addition of vectors in space. In other words, we could say that the position vector A, representing the position in which object A is located, is given by the addition of the position vector B, describing the location of object B, and the vector corresponding with the direction of the relation. The parameter d, which has been given as an argument to the *Before* predicate describes the distance between the reference object and the object which has to be located. This parameter gives us the length of the vector we have to add to the position vector B to become position vector A. In the definition above, A would be at a distance of d before B. The "+"-operator and the parameter d have in all of the following definitions the same functionality.

We would like to remind the reader that *before* does not necessarily mean *in front of*. When we say that object A is "before" object B does not infer that the object A is "in front of" object B. The "front" of the object B could be in some other direction. In this example, object A that is "before" object B for the modeller could be "behind", "right", "above", ... of object B for the object B itself since we do not know its orientation. So, for the moment we will not be concerned with the possible orientation of the reference object. This will also be the case for the following definitions except when explicitly stated.

We now proceed with the *RightOf* relation. Figure 6.2 illustrates clearly that the cone is "right-of" the cube. When we generalize this we can say that an object A is "right-of" another object B when it has a larger y-value. Again, drawing a half-line beginning at the position of object B in the *right* direction, object A is "right-of" object B when it is somewhere on that line. In figure 6.2 we could say that the cone is "right-of" the centre cube. This results into the next definition.

Definition

$$RightOf(A, B, d, F_F)$$
$$\Updownarrow$$
$$A = B + (0i_F + dj_F + 0k_F) \text{ with } d \in I\!R_0^+ \text{ and } A, B \in I\!R^3$$

In the third dimension we have the *Above* relation. In figure 6.2 again we can see that the cylinder in the middle is above the cube. A definition of the *Above* relation would state that an object A is somewhere "above" object B when its position coincides with a point on the half-line parallel with the *above* direction and with the position of object B as end point. We can see this relation displayed in figure 6.2 where the cylinder is "above" the cube. Translating this relation so it is written in terms of vectors we get the following definition.

Definition

$$Above(A, B, d, F_F)$$
$$\Updownarrow$$
$$A = B + (0i_F + 0j_F + dk_F) \text{ with } d \in I\!R_0^+ \text{ and } A, B \in I\!R^3$$

The definitions of the remaining orientation relations are described in table 6.3. The first three *After*, *LeftOf* and *Below* are the inverses of the relations we have seen above. When object A is *Before* object B, the reverse is also true, object B is *After* object A. The same connection holds between the *LeftOf* and the *RightOf* relations. The fourth, the *MiddleOf* relation is a bit different. The function *MiddleOf* takes two known locations, the ones of object B and C, as input and results in the location of object A as output. Object A is situated "in-the-middle-of" object B and C. The distance from object B to A is the same as the distance from object C to A.

6.4.3 Combination of the basic relations

The orientation relations described in previous section are only the basic orientation relations. Sometimes it is needed to be able to specify spatial relations in more detail.

Relation	**Definition**
$After(A, B, d, F_F)$	$A = B + (-di_F + 0j_F + 0k_F)$ with $d \in IR_0^+$ and $A, B \in IR^3$
$LeftOf(A, B, d, F_F)$	$A = B + (0i_F - dj_F + 0k_F)$ with $d \in IR_0^+$ and $A, B \in IR^3$
$Below(A, B, d, F_F)$	$A = B + (0i_F + 0j_F - dk_F)$ with $d \in IR_0^+$ and $A, B \in IR^3$
$MiddleOf(A, B, C, F_F)$	$A = \frac{1}{2}(B + C)$ with A, B and $C \in IR^3$

Table 6.3: Definitions of Orientation Relations

An object might not be just "after" another object, or just "right-of" another object but a little in between. The object would be "right-of" and "after" another object. To do this, we can combine several basic relations into more detailed relations. Combining the *After* relation with the *RightOf* relation results in a relation *Right-After-Of*. Putting this relations in a phrase would mean that the object would be "right-after" (right and after) another object. As another example with three relations we could say that an object is "before-left-above-of" another object by combining the *Before*, the *LeftOf* and *Above* relation. This means that the object would be positioned "before", "left" and "above" another object. Putting two relations together which are the same would have no result and thus would not be of any use.

In math, the aspect of combining the basic relations only requires us to add up the unit vectors involved in the new relation to become a vector representing the direction of the relation. Afterwards, this vector has to be normalized to become the unit vector representing the direction of the relation. The mathematical definitions of the different combinations of orientation relations will not be discussed here.

6.5 Directional Relations

6.5.1 Global Reference Frame

When talking about objects far away from us, we usually tend to use another kind of reference frame, namely a reference frame that is independent of the orientation of someone standing inside the Virtual World. No matter from which view-point the modeller would look at the world, the reference frame will not change. This reference frame is inside the world. It is similar to laying a compass on a map of the world. No matter where we would lay the compass, it would always point to the same directions. Relations between places on the map can be specified using the directions pointed out by the compass. For example, an object A is "north-of" another object B. So, here, the compass provides the reference frame. This reference frame is called the Global Reference Frame and is an absolute reference system as we have seen in section 5.4. With this Global Reference Frame we will associate directional relations (north, south, east, west, up and down), which can also be used to indicate the position of objects in the world. We said earlier that a reference frame should have an origin, and three axes denoting the directions. The Global Reference Frame does not need all these properties. It does not have an explicit origin. Giving an object a position in terms of this reference frame makes no sense. Only the directions and how they are oriented relative to the Fundamental Reference Frame are important for calculating the correct location of the primary object on the basis of the reference object. Thus, we would only use the directions the needle of the compass points to.

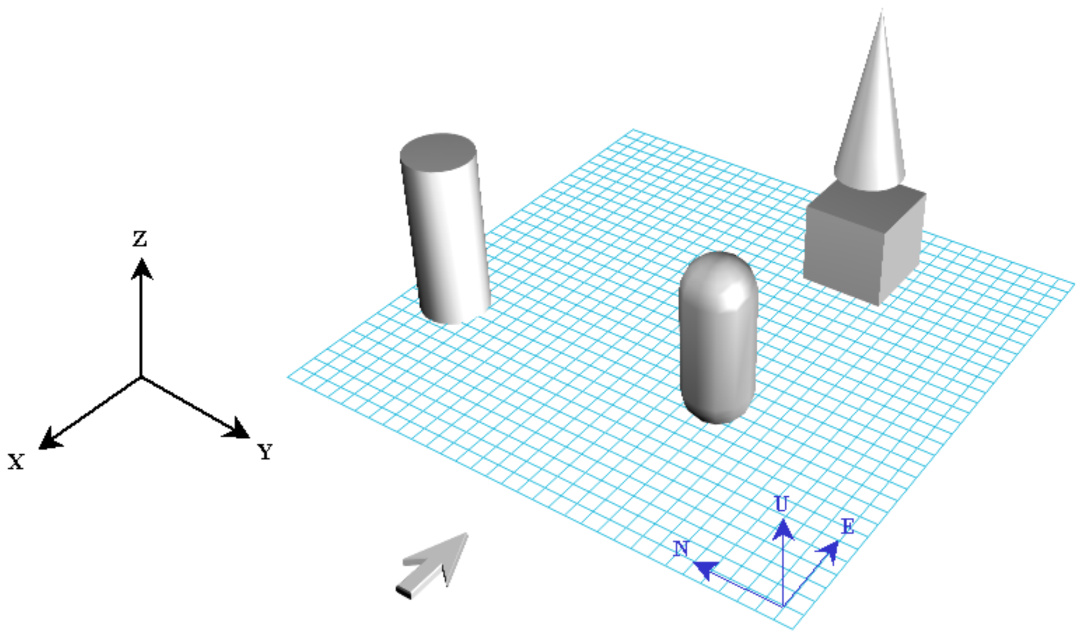


Figure 6.3: Global Reference Frame

In figure 6.3, a Global Reference Frame is given besides the Fundamental Reference Frame discussed above. The direction from which we view the Virtual World is again given by the arrow. This Global Reference Frame is drawn inside the world as can be seen in the figure. We will say that the vector $\mathbf{i}_G$, which runs along the N-axis, represents the north-side. The vector $\mathbf{j}_G$, which is parallel with the E-axis, represents the east-side. The vector $\mathbf{k}_G$ in the direction of the U-axis, represents the up-side. The reverse vectors are connected to the south-, west- and down-side respectively. This again is a personal choosing. These unit vectors will be the basis for the definitions of the directional relations, as we will see in the next section.

6.5.2 Definitions of the basic relations

Using the north-side direction, we can describe the possibilities an object A has in being "north-of" object B. In other words, if object A would be somewhere on the line beginning at the position of object B and have the same direction as mentioned by the north direction, it would be "north-of" object B. In the figure 6.3 we can see the cylinder positioned "north-of" the capped cylinder. Thus, in mathematical terms the statement below holds.

Definition

$$NorthOf(A, B, d, F_G)$$
$$\Updownarrow$$
$$A = B + (di_G + 0j_G + 0k_G) \text{ with } d \in I\!R_0^+ \text{ and } A, B \in I\!R^3$$

Please, pay attention to the subscript carried by the unit vectors. These state that we are working with the directions as specified by the Global Reference Frame. This is different from the section above where we used the directions of the Fundamental Reference Frame.

All the other directional relations can be defined in analogy with the *NorthOf* relation. They will not be further discussed. An overview is given in table 6.4. We can easily see the connection between the inverse relations *NorthOf-SouthOf*, *EastOf-WestOf*, *Up-Down*. In the example situation in figure 6.3, we can see the cube is "east-of" the capped cylinder. The capped cylinder is "south-of" the cylinder and the capped cylinder is "west-of" the cube. The cone is "up" the cube and the cube is "down" the cone.

Relation	**Definition**
$EastOf(A, B, d, F_G)$	$A = B + (0i_G + dj_G + 0k_G)$ with $d \in I\!R_0^+$ and $A, B \in I\!R^3$
$Up(A, B, d, F_G)$	$A = B + (0i_G + 0j_G + dk_G)$ with $d \in I\!R_0^+$ and $A, B \in I\!R^3$
$SouthOf(A, B, d, F_G)$	$A = B + (-di_G + 0j_G + 0k_G)$ with $d \in I\!R_0^+$ and $A, B \in I\!R^3$
$WestOf(A, B, d, F_G)$	$A = B + (0i_G - dj_G + 0k_G)$ with $d \in I\!R_0^+$ and $A, B \in I\!R^3$
$Down(A, B, d, F_G)$	$A = B + (0i_G + 0j_G - dk_G)$ with $d \in I\!R_0^+$ and $A, B \in I\!R^3$

Table 6.4: Definitions of Directional Relations

6.5.3 Combination of the basic relations

Analogous with the orientation relations, only having six basic directional relations is too restrictive. An object could be "east-of" another object and "north-of" that same object at the same time. In common-sense terms we will say the object is "north-east-of" the other object. Apart from the reference frame, which is different here, the definitions of the combinations of directional relations are completely similar to those of orientation relations.

6.6 Topological Relations

6.6.1 Independent of any Reference Frame

People sometimes use other relations, namely topological relations as we have seen in section 5.3, than the directional and orientation relations to describe positions of objects. Objects can be touching each other, two objects can be disjoint with each other, or one object can be inside another object. Some of the topological relations are more used in the sense of a constraint between the objects involved in the relation. This is because they do not really describe a specific location by which we are able to position the object on the basis of another object. If an object A is disjoint from object B, it could be everywhere in the Virtual World as long as it does not have any points in common with object B. In contrast to the directional relations and the orientation relations, the topological relations are not dependent on a reference frame.

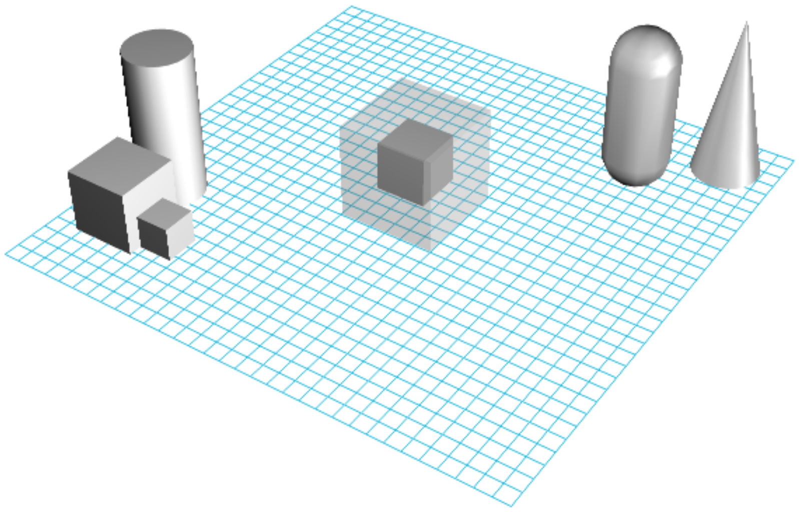


Figure 6.4: Topological Relations, independent of a reference frame

6.6.2 Definitions of the basic relations

We first introduce some preliminary mathematical definitions, which we will use for the definitions of the topological relations. The topological relations will be based on the comparison of the interior, boundary and exterior of an object with the interior, boundary and exterior of another object. Therefore, we first define the concepts interior, boundary and exterior of an object. This has been done within the field of the point-set topology [40][12].

We start with the boundary points. A point is on the boundary of an object if it is not completely contained inside the object but also not completely outside. If A is a subset of the three dimensional space (IR^3), then a point $x \in IR^3$ is a boundary point of A if every neighbourhood of x contains at least one point in A and at least one point outside A. The neighbourhood of a point $x \in IR^3$ is the set of points inside a ball with centre x and radius $\epsilon > 0$.

Definition

The boundary is the set of all the boundary points of an object. The boundary of A contains the points x with every neighbourhood ($B_r(x)$) that contains at least one point a in A and at least one point b in the complement of A, A^c.

$$Boundary(A) = \\ \{x \mid \forall B_r(x)\ (\exists a(a \in A) \wedge \exists b(b \in A^c))\ with\ a, b \in B_r(x)\ and\ r > 0\}$$

Definition

The interior is that portion of a region lying ”inside” a specified boundary being the set of all interior points. Every internal point x has a neighbourhood ($B_r(x)$) which is contained entirely in A.

$$Interior(A) = \{x \mid \exists r\ (r > 0 \wedge B_r(x) \subset A)\}$$

Definition

The exterior of an object is that portion lying ”outside” a specified boundary. The exterior of A is the set of all exterior points of the object A. In other words, it is the set of all interior points of the complement of object A.

$$Exterior(A) = \{x \mid x \in Interior(A^c)\}$$

Now, we can easily construct the topological relations in terms of the Boundary, Interior and Exterior functions we just described. Four different topological relations will be described in the remaining of this section: *Disjoint*, *Touch*, *Inside*, *Equal*.

An object A is said to be "disjoint" of object B when they have no boundary points in common, thus if the intersection of the boundary points of A with the boundary points of B is empty. In figure 6.4, the cone and the capped cylinder on the right are disjoint objects. Writing this as a mathematical expression results in the following definition.

Definition

$$\boxed{\begin{array}{c} Disjoint(A,B) \\ \Updownarrow \\ (Boundary(A) \cap Boundary(B) = \emptyset) \wedge (Interior(A) \cap Interior(B) = \\ \emptyset) \wedge (Boundary(A) \cap Interior(B) = \emptyset) \wedge (Interior(A) \cap Boundary(B) = \emptyset) \end{array}}$$

The relation, *Touch* says the opposite. Here, an object A "touches" another object B when the intersection of their boundary sets has some elements in common. Object A shares some points of its boundary with the boundary of object B.

Definition

$$\boxed{\begin{array}{c} Touch(A,B) \\ \Updownarrow \\ (Boundary(A) \cap Boundary(B) \neq \emptyset) \wedge (Interior(A) \cap Interior(B) = \\ \emptyset) \wedge (Boundary(A) \cap Interior(B) = \emptyset) \wedge (Interior(A) \cap Boundary(B) = \emptyset) \end{array}}$$

On the left side of figure 6.4, the small cube is against the bigger cube, they have a set of points in common and thus are said to "touch" each other. The same counts for the cylinder and the big cube.

An object A is "inside" another object B when all elements of object A are elements of the object B. In mathematics, we will get the following definition.

Definition

$$\boxed{\begin{array}{c} Inside(A,B) \\ \Updownarrow \\ (Interior(A) \cap Interior(B) \neq \emptyset) \wedge (Boundary(A) \cap Interior(B) \neq \\ \emptyset) \wedge (Interior(A) \cap Boundary(B) = \emptyset) \end{array}}$$

When we look at the centre of figure 6.4 we can see the big cube, which has been made a little transparent in order to see the little cube. About this structure we can say that the big cube contains the little cube and the little cube is "inside" the big cube.

If every point of an object A is also a point of another object B and when the reverse, if every point of object B is also a point of object A, we can say that object A and object B are "equal". In mathematics this can be expressed as in the definition below.

Definition

$$\boxed{\begin{array}{c} Equal(A,B) \\ \Updownarrow \\ (Boundary(A) \cap Boundary(B) \neq \emptyset) \wedge (Interior(A) \cap Interior(B) \neq \\ \emptyset) \wedge (Boundary(A) \cap Interior(B) = \emptyset) \wedge (Interior(A) \cap Boundary(B) = \emptyset) \end{array}}$$

6.7 Building the Upper Ontologies

Now that we have defined both the part-whole relations and the spatial relations we are able to mould them into an upper ontology. We will first describe the upper ontology that contains the language constructs for using the part-whole relations. Afterwards we will describe the upper ontology consisting of all the structures the user will need to define spatial relationships between objects in a Virtual Environment. These ontologies will have to be incorporated into the system that we described in chapter 3 to give the user the possibility to use them in their domain ontologies. To illustrate the use of the upper ontologies, an example is given in the next chapter.

6.7.1 The Part-Whole Ontology

The first ontology describes the part-whole relations and the concepts that are needed to define the part-whole relations. Figure 6.5 gives us the overall structure of the Part-whole ontology. The main class is the *Part-Whole Relation*. This class is a subclass of the general *Relation* class. The general *Part-Whole Relation* refers to two objects of type *Spatial Object*: the "part-object" and the "whole-object" of which the part-object is a part. These two properties are inherited by all the specialized part-whole relations. So if one wants to create a specialized part-whole relation, the part-object and the whole-object need to be given as properties of the relation on top of the other properties defined for the specialized part-whole relation.

The five special part-whole relations we found to be important for our work, and which we have defined earlier in this chapter, are all defined as subclasses of the general *Part-Whole Relation*. All these relations are defined by means of a hierarchy of relation elements, instances of the *RelationElement* class (Functional, Homogeneous, ...)(corresponding to our definition given in section 6.2). This is the main part of

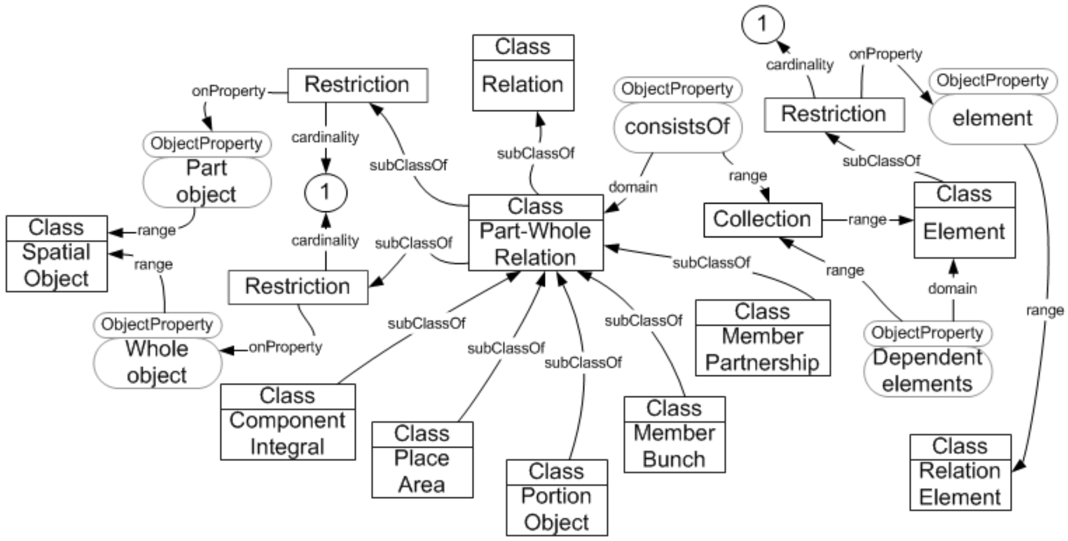


Figure 6.5: Part-Whole relations ontology

the Part-whole ontology. The complete specification of the Part-whole ontology can be found in appendix A.

6.7.2 The Spatial Ontology

The second ontology describes the spatial relations. Figure 6.6 is an illustration of the graph that represents the Spatial ontology. Only the most important concepts are displayed here. For the entire printout of the Spatial ontology we would like to refer the reader to appendix B. Here, the main concept in the figure is the *Spatial Relation* being a subclass of the more general concept *Relation*. The *Spatial Relation* is restricted to have at least one primary object and one or more reference objects, which have to be instances of *Spatial Object*.

Three subclasses of the spatial relation class are distinguished to represent the different kinds of spatial relations: orientation relations, directional relations and topological relations. Each of these relations require some "type" properties which have to be instances of respectively an *Orientation* ("LeftOf", "Above", "Before", "MiddleOf", ...), a *Direction* ("NorthOf", "EastOf", "Up", ...) or a *Topology* ("DisjointWith", "Touches", "Inside", ...). These instances are predefined in the upper ontology. Each relation of a certain kind can have multiple "type" properties of the kind corresponding with the kind of relation. This way, more specialized relations can be built. The orientation relations and the directional relations need an additional property (distance) that has to be a decimal value. The directional relations also

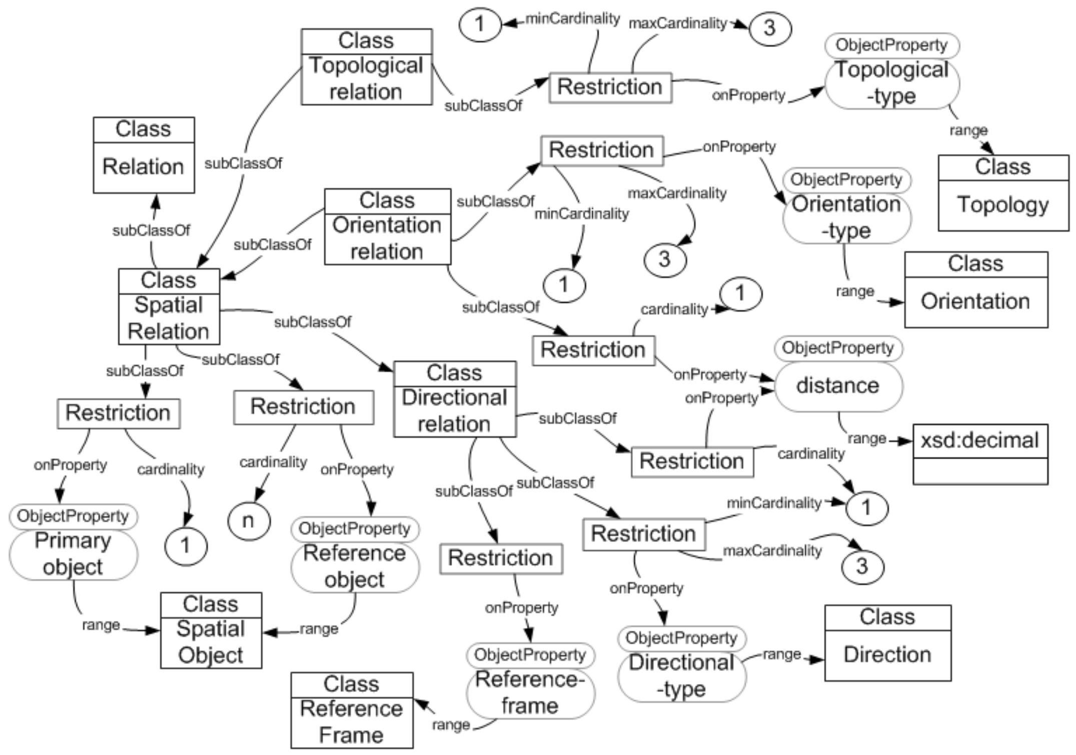


Figure 6.6: Spatial relations ontology

need a reference frame, an instance of the *ReferenceFrame* class, on top of the other properties.

6.8 Summary

In this chapter we introduced the formal definitions of the spatial relations and the part-whole relations. All these relations are described in an upper ontology. The new method for building Virtual Worlds from domain ontologies used at the WISE lab can now be extended with this work. When the upper ontologies are incorporated into the prototype system we discussed in chapter 3, the concepts defined in those upper ontologies can be used by the developer of the Virtual Environment, the user of the system, to describe the Virtual World in a more intuitive manner. An example of the use of the upper ontologies will be presented in the following chapter.

Chapter 7

Example: A real World into a Virtual World

In this chapter we will illustrate how to use the concepts defined in the upper ontologies that we have built in the previous chapter. This will be done with a real life example. The relations between the objects in the example are expressed in terms of the concepts defined in the upper ontologies, although, as far as this is possible. Afterwards we will show how this example would look like once we would have completed the generation of the Virtual World. The extensions we have to make to the OntoWorld tool to realise this are discussed in the following chapter.

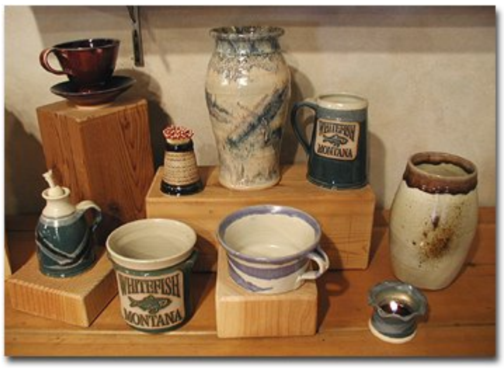


Figure 7.1: Whitefish Pottery products

For the example we will take a product composition of a store. The *Whitefish Pottery* is a small company specialized in handmade ceramics featuring a wide selection of vases, bowls, mugs, pots,... A small part of their new collection of products is displayed in figure 7.1.

The following pieces of DAML+OIL code represent two orientation relations. The first one says that the beer pot on the big wooden block is *RightOf* the big vase in the middle of the figure at a distance of about 10 cm. The next relation combines the *RightOf* relation and the *Before* relation saying that the small vase is right as well as before the big block at about 25 cm.

```
<Orientation-Relation rdf:ID="BeerPot-Vase">
  <primary-object rdf:resource="#BeerPot"/>
  <reference-object rdf:resource="#Vase"/>
  <topological-type rdf:resource="#RightOf"/>
  <distance>10.0</distance>
</Orientation-Relation>

<Orientation-Relation rdf:ID="Vase-LargeBlock">
  <primary-object rdf:resource="#Vase"/>
  <reference-object rdf:resource="#LargeBlock"/>
  <orientation-type rdf:resource="#RightOf"/>
  <orientation-type rdf:resource="#Before"/>
  <distance>25.0</distance>
</Orientation-Relation>
```

Another kind of relation we have seen is the topological relation. We can constrain the vase in the middle of the figure and the beer pot on its right to be disjoint by setting a *disjointWith* relation between them. This relation would then be expressed as follows.

```
<Topological-Relation rdf:ID="Vase-BeerPot">
  <primary-object rdf:resource="#Vase"/>
  <reference-object rdf:resource="#Beerpot"/>
  <topological-type rdf:resource="#DisjointWith"/>
</Topological-Relation>
```

However, not all the relations can easily be expressed by means of the various concepts we defined in the upper ontologies. Objects that need to be positioned in finer grained angles to each other need a more exact positioning mechanism. Also (to be able to generate the Virtual World) the first object in the Virtual World has to be given an exact location without making use of the relations because otherwise it may be possible that we cannot determine the location of a reference object in order to localize the primary object. Therefore the developer is still able to place objects at specific locations by giving the object a set of coordinates, as was the case for all the objects in the old system. The way to place an object at specific coordinates is shown in the following code.

```
<SpatialObject rdf:ID="LargeBlock">
  <Coordinate>
    <x-coordinate>0.0</x-coordinate>
    <y-coordinate>0.0</y-coordinate>
    <z-coordinate>0.0</z-coordinate>
  </Coordinate>
  ...
</SpatialObject>
```

Now, we illustrate the part-whole relations. The coffee cup in the back on the left of the same figure belongs to the small plate underneath it. When one of them is being moved, we want that the other one will move accordingly. Therefore we can say that both of them are part of one 'coffee-set'. A part-of relation, the *Component-Integral* relation, is suitable for doing this. The following DAML+OIL code will set a relation between the coffee cup and the coffee-set.

```
<ComponentIntegral-Relation rdf:ID="CoffeeCup-CoffeeSet">
  <part-object rdf:resource="#CoffeeCup"/>
  <whole-object rdf:resource="#CoffeeSet"/>
</ComponentIntegral-Relation>
```

We may also consider all objects shown as a collection. This will allow us, later on, in the Virtual World, to move or manipulate them as a whole. Hence, the *Member-Bunch* relation is the right relation for expressing this. The soup bowl in the front of the figure is said to be a part of the new collection by the following code.

```
<MemberBunch-Relation rdf:ID="Bowl-Collection">
  <part-object rdf:resource="#Bowl"/>
  <whole-object rdf:resource="#Collection"/>
</MemberBunch-Relation>
```

Figure 7.2 shows us the Virtual World that results from the generation process. We can see that with a rather small set of relations quite complex Virtual Worlds can be built in an intuitive way.

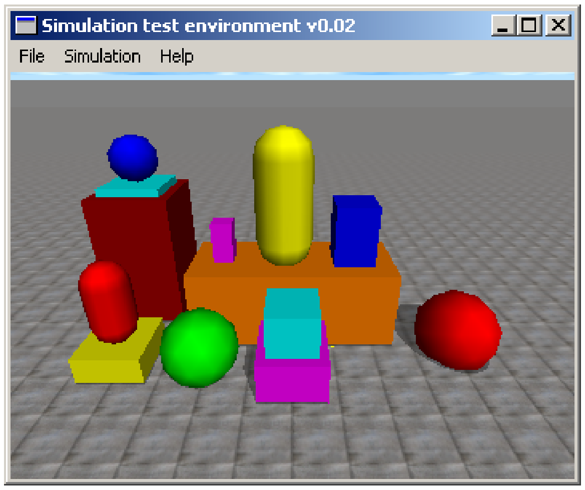


Figure 7.2: The Virtual World

Chapter 8

Extensions to the OntoWorld tool

8.1 Overview

In this chapter we will describe the changes that have to be made to the OntoWorld tool we discussed in chapter 3. The general process of the approach is almost the same as before. The process is broken into the same phases but each phase is a bit different. In a first phase the Domain ontology is being translated into a Representable Domain ontology. As before, this Representable Domain ontology describes how the objects in the domain should be represented in the Virtual World. In the next step, the result of the first step is used to come to a working Virtual Environment. Figure 8.1 shows the overview of the generation process. These two steps have to be changed, as we will see in the following sections. Section 8.2 will describe the new system of ontologies. Section 8.3 will elaborate on the changes that have to be made to the different steps of the generation process.

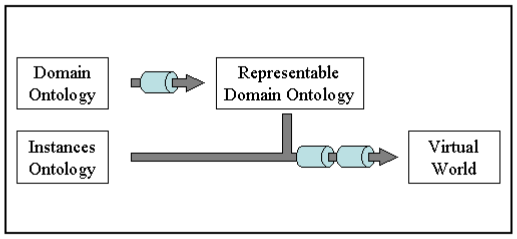


Figure 8.1: The complete generation process

8.2 The Expanded System of Ontologies

An overview of the new system of ontologies and their relationship to each other is given in figure 8.2. A detailed description of the ontologies is given in section 3.3. The only thing in which this system is different from the old system is the inclusion of the upper ontologies that we have defined in section 6.7. Before, the developers of the Domain ontology and the Instances ontology could only use concepts defined in the DAML+OIL ontology, which provided them only with the basic language constructs. The "class" construct was used to define an object. The "property" construct was used to define the properties for objects. Now, they can also use the concepts that are defined in the upper ontologies. The Part-Whole ontology contains all the concepts related to the part-whole relations and the Spatial ontology contains all the concepts related to the spatial relations.

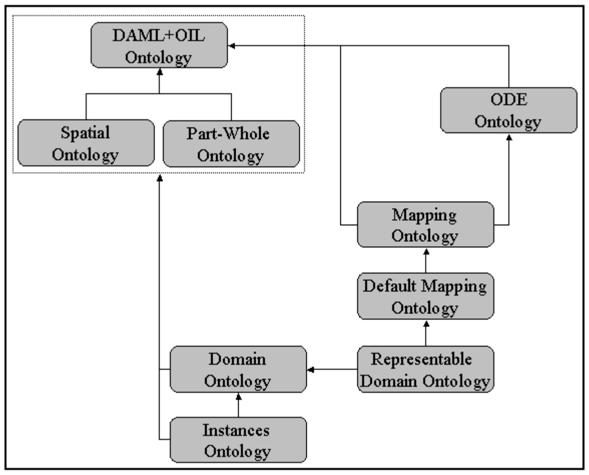


Figure 8.2: Overview of the ontologies

8.3 The New Generation Process

The first step of the generation process is to create a Representable Domain ontology. For every class in the Domain ontology the user is asked for a suitable physical representation. Then the user is asked to give representations for the attributes that are needed for the ODE object on which the world object is mapped. Because the locations of the objects are specified by concepts in the Spatial ontology there is no need to create attribute mappings that are related to the localization of the object.

As a result, the user has to specify less mappings and this will speed up the generation process.

Also the second step needs some change. In this step, the Instances ontology is taken and for every instance, a translation to C++ code is made using the information in the Representable Domain ontology. We will now divide this step into two phases because in addition, we have to derive the locations of the objects by means of the spatial relations in the Instances ontology. The remaining attributes are retrieved in the same way as before. Due to the part-whole relations also the connections between the objects have to be set. Before we can do all this, some checking is needed to see whether all the relations that are specified between the objects are consistent.

8.3.1 Consistency Checking

One thing that we have to take into account is the fact that, when the developer is building a large Virtual Environment with many relations, mistakes may occur. The part-whole relations are subject to some properties. For example, there have to be several disjoint objects to form a whole or one object can not be a part of itself. Furthermore, if one object is a part of another, different from the first, then the second can not be a part of the first anymore. The user could violate these properties so we need some sort of notification when this happens. The spatial relations can also raise errors. E.g. if an object is given a location using different specifications, then these locations should be consistent. An algorithm has to be implemented that can report inconsistencies or ambiguities in the specification of the Virtual World, so the user can resolve them.

8.3.2 Virtual World Generation

The actual generation of the Virtual World is the most important step. In the old system this was a rather easy step because all the information needed to generate the Virtual World was explicit. Now, we have to make additional calculations to derive part of the data. The new generation process can be divided into two steps. First an algorithm has to be implemented that will generate a list of C++ objects corresponding to the instances in the Instances ontology. This will be the most difficult step in the process. In the second step, the actual code is generated by means of the list of C++ objects. The result will be an Object-Oriented program representing the Virtual Environment, which can be executed afterwards. The following subsections will discuss the two steps in more detail.

Object Generation

This step will take the Instances ontology and the Representable Domain ontology as input and produces a list of C++ objects as output. On the one hand we have the *primitive objects*. The objects that are actually instantiations of the C++ classes representing the primitives that are supported by ODE. On the other hand we have the *composite objects*. These objects are playing the role of a whole in the Virtual World. The *composite objects* do not correspond with one of the primitive ODE objects. They are instances of a C++ class that only carries links to the part objects. How these two kinds of objects will look like can be seen appendix C. Because a lot of information about the objects is now given in the spatial relations and the part-whole relations, we can not use the same algorithm for generating the Virtual World as we did before. We have to define a new one. The following pseudo code gives us a general overview of what has to be done in this new algorithm. Note that the consistency checking could also have been implemented together with this algorithm instead of separating it but we believe that for large Virtual Environments it is desirable to have a separate consistency checker.

```
GENERATE_OBJECTS(Representable Domain ontology: R, Instances ontology: I)
1.  L ← {}
2.  for each spatial object instance $i$ in I
3.      if i isa composed object instance
4.          Create object $o$ of type ComposedObject
5.      else
6.          Calculate position of $i$
7.          Get mapping $m$ from R with $source_m = i$
8.          Create object $o$ of type $target_m$
9.      for each attribute mapping $a$ in $m$
10.         Set $target_a$ in $o$ with the value of property $source_a$ of $i$
11.     Fix the part-whole links of $i$ in $o$
12.     Add $o$ to the list L
13. return L
```

We start by creating an empty list that will be filled with the objects and that will be returned after the algorithm finishes. The Instances ontology contains instances of either the relations in the upper ontologies or the classes in the Domain ontology. We will refer to this last kind of instances as spatial object instance. For every spatial object instance in the Instances ontology we have to perform three things:

- First, we have to calculate the exact position of the spatial object by means of the spatial relations that have this particular object as a primary object.

When we know the locations of the reference objects in the relations this can be done easily using the definitions of the spatial relations. The position of a composed object will not have to be calculated because these will not be explicitly visualized in the Virtual World.

- Then, we have to create an object of a type that corresponds to the one that is given as the target in the Representable Domain ontology. Suppose we wanted an object in the domain be visualized as a sphere in the Virtual World, then an instantiation of the "Sphere" class is made as is described in appendix C. For all the different kinds of graphical primitives (Box, Capped Cylinder,...), which are supported in the ODE engine, we will create a class similar to the "Sphere" class that is described in the appendix. In case of a composed object an instance of the "ComposedObject" class (see appendix C) will be created. After we have created the object we can fill in the remaining attributes using the attribute mappings of that particular instance. The values are looked up in the Instances ontology and set in the object.

- Finally, we also have to fix all the part-whole links that exist between the objects through the part-whole relations. This means that if an object A is a part of an object B, we have to add A to the list of child-objects of B in order for object B to know it has object A as a part.

The algorithm will result in a list that contains all the objects in the Virtual World together with the values for all their attributes. This list will be used to perform the next step.

Generation of Initialisation Code

The generation of the code that initializes the world starts with generating the code for the include files. We always need the same include files, so we can simply insert the include instructions into the source file. Then, three procedures are generated that are necessary for running the simulation of the Virtual World. To run the Virtual World, the "Start" procedure is invoked. In this procedure the correct parameters are specified for setting the camera position. After this procedure has been executed the "Simloop" will take care of the continuous drawing and refreshing of the objects in the Virtual World. When the user of a Virtual World closes the world, then a procedure named "Stop" is called. This procedure contains the code to destroy the objects in the world and the world itself. All these procedures can be put into one class. This class will contain the methods "Start", "Simloop" and "Stop" similar to the equal named procedures and will be defined as follows:

```
class WorldGenerator {
public:
  /** Constructor & Destructor */
  WorldGenerator();
  ~WorldGenerator();

  // Simulation functions
  void start();
  void simLoop(int pause);
  void stop();

  void addObject(Object* o);
private:
  list<Object*> objects_;
};
```

Finally, the code for the "main" procedure has to be generated. In the old system, after having generated the code for the initialisation of the Virtual World, all the objects were created and all their states (position, density, etc.) were set one by one. When there were many objects in the Virtual World this results in a lot of programming code and much of the code is duplicated over and over again. Therefore, this will be replaced by just generating code for the initialisation of the WorldGenerator class. This class will then initialize the world. All we have to do is generating the code for the initialisations of the C++ objects obtained in the first step of the process and also generating the code for adding them to the worldgenerator. The code generation will be ended with calling the "Simloop" method of the worldgenerator. This main source file together with the other source files that were already pre-built, the WorldGenerator and the files containing the different graphical component classes (Sphere, Box,. . .), will result in an Object-Oriented executable after compilation. The Virtual World is now ready and can be executed independently of the generation tool.

Chapter 9

Future Work

In this thesis we have proposed two new concepts (spatial relations and part-whole relations) which will contribute to the modelling power of the OntoWorld tool. However, still a lot of research needs to be done to arrive at a full operable system with which a developer can generate large and complex structured Virtual Environments and specify them as intuitively as possible.

With the spatial relations defined here, we are able to define structures of objects that are not physically connected to each other. With the exception of very simple objects, such as a ruler, most objects we encounter in every day life have more than one component part put together. When you design a set of components for an assembly, the relative dimensions and positions of parts, and how they fit together, are crucial. To design a complex assembly that has a large number of parts, you organize the parts into smaller assemblies (sub-assemblies) and put the subassemblies into a larger assembly. Objects can be combined with each other by means of constraints of some sort on the components. All the 3D engines provide us an elaborate set of joints to do this. We have the standard joints like the ball-and-socket joint, the hinge joint, the spring joint,.... Some physics engines also support more exotic types of joints like rolling joints and sliding joints. We can clearly see that the introduction of connections is a necessity to the system.

When we have all these joints available, we are able to build real mechanical systems. Systems are only of some use, when we could also make the mechanical parts of the system move. It would thus be a very interesting topic to include forces into the Virtual World. If we look at the Virtual Environment as being in a particular state at a particular time, a force is needed to bring the Virtual Environment into another state at the next timestep. In other words, for objects in the Virtual World to be able to move, they have to be subject to some force. For example, an object could fall down after the user has pushed it from the table, because of the gravity in the Virtual Environment.

Another thing that can be done is investigating the constraints between the objects upon user interaction. What will happen to the other objects when we place an object to another location? The spatial relations defined could behave as begin conditions; in this case if one object would be moved, the objects that are related would not move. This means that the spatial relations would not hold anymore; they were only used to specify the begin state of the Virtual Environment. However, we could also consider the spatial relations as time independent constraints. Then, all the objects that are directly or indirectly related to the object that had been moved will also move. Intuitively, the part-whole relations should also behave like this. If we would move the whole, all the parts should move accordingly. If we would move a part-object from the whole, then the movement of the other part-objects would be according to whether the relation is set to be separable or not. Maybe there are additional constraints, within the light of this topic, that have to be added to the system. Like for example if we have a table-set consisting of a table and a chair. How far can we move the chair from the table until it will not be regarded as a part of the set anymore? Some space before the table could be defined as a constraint space in which the chair has to stay in order to be a part of the table-set. The same question holds for the connections. How far could we move an object that is connected with another one before the connection would break? Here, we could also define a region in which the relation will hold. Or we could set some kind of force metric to define the amount of force that the connection can bear before it breaks. Another question would be in which direction an object is allowed to move. Again, constraints could be added to define this.

Chapter 10

Conclusions

We will end this thesis with some conclusions that can be drawn from it. The goal of this work was to investigate ways to build more complex Virtual Worlds in an intuitive way and with as little user interaction as possible.

Describing the Virtual World as a set of unrelated objects is too simple. To help the designer in building Virtual Environments, we have introduced two kinds of relations to express relationships between objects: the part-whole relations and the spatial relations. In chapter 4 we revised and discussed part-whole relations and in chapter 5 we revised and discussed spatial relations. These relations can be used respectively to combine objects together or to specify the location of objects relative to some other objects. In chapter 6 we have given the definitions of all these relations and we have built the upper ontologies (chapter 2) containing the concepts we defined. These upper ontologies are needed in order to be able to use the concepts when building a Virtual World. Chapter 7 gave us an example of how to use these upper ontologies. In chapter 8 we discussed the extension that has to be made to the OntoWorld tool we explained in chapter 3.

We believe that the introduction of the spatial relations and the part-whole relations makes the approach for building Virtual Environments much more powerful without violating the intuitiveness of the approach. In addition, the use of the spatial relations speeds up the building process because the developer does not need to specify an exact location for every object in the Virtual World. The part-whole relations are extremely useful for bringing structure into the Virtual World. The user is able to group the objects instead of just building a Virtual World with lots of unrelated and unstructured objects. Although we are still in a very early phase with this research we can feel that this is a step in the right direction for a more intuitive way of building Virtual Environments.

Appendix A

Part-Whole relations upper ontology

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
  xmlns:xsd="http://www.w3.org/2000/10/XMLSchema#"
  xmlns:pwo="http://mach.vub.ac.be/~bpellens/PartWholeOntology.daml#"
  xmlns="http://mach.vub.ac.be/~bpellens/PartWholeOntology.daml#">

  <daml:Ontology rdf:about="">
  <daml:versionInfo>PartWholeOntology,2003/04/25</daml:versionInfo>
  <daml:imports rdf:resource="http://www.daml.org/2001/03/daml+oil"/>
  </daml:Ontology>

  <daml:Class rdf:ID="PropertyValue"/>
  <daml:Property rdf:ID="hasProperty">
    <daml:domain rdf:resource="<<daml+oil>>#Thing"/>
    <daml:range rdf:resource="#PropertyValue"/>
  </daml:Property>

  <daml:Class rdf:ID="SpatialObject">
    <rdfs:subClassOf rdf:resource="PropertyValue"/>
  </daml:Class>

  <daml:Class rdf:ID="Relation" />
```

```
<daml:Class rdf:ID="PartWhole-Relation">
  <rdfs:subClassOf rdf:resource="#Relation"/>
  <rdfs:subClassOf>
    <daml:Restriction cardinality="1">
      <daml:onProperty rdf:resource="#part-object"/>
    </daml:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <daml:Restriction cardinality="1">
      <daml:onProperty rdf:resource="#whole-object"/>
    </daml:Restriction>
  </rdfs:subClassOf>
</daml:Class>

<daml:Property rdf:ID="part-object">
  <daml:subPropertyOf rdf:resource="#hasProperty"/>
  <rdfs:range rdf:resource="#SpatialObject"/>
</daml:Property>

<daml:Property rdf:ID="whole-object">
  <daml:subPropertyOf rdf:resource="#hasProperty"/>
  <rdfs:range rdf:resource="#SpatialObject"/>
</daml:Property>

<daml:Property rdf:ID="consistsOf">
  <rdfs:range rdf:resource="<<daml+oil>>#List"/>
</daml:Property>

<daml:Property rdf:ID="dependent-elements">
  <rdfs:domain rdf:resource="#Element"/>
  <rdfs:range rdf:resource="<<daml+oil>>#List"/>
</daml:Property>

<daml:Property rdf:ID="element">
  <rdfs:domain rdf:resource="#Element"/>
  <rdfs:range rdf:resource="#RelationElement"/>
</daml:Property>

<daml:Class rdf:ID="RelationElement"/>
```

```
<daml:Class rdf:ID="Element">
  <rdfs:subClassOf>
    <daml:Restriction cardinality="1">
      <daml:onProperty rdf:resource="#element"/>
    </daml:Restriction>
  </rdfs:subClassOf>
</daml:Class>

<RelationElement rdf:ID="Connection"/>
<RelationElement rdf:ID="Functional"/>
<RelationElement rdf:ID="Homogeneous"/>
<RelationElement rdf:ID="Separable"/>
<RelationElement rdf:ID="NonFunctional"/>
<RelationElement rdf:ID="Nonhomogeneous"/>
<RelationElement rdf:ID="NonSeparable"/>

<daml:Class rdf:ID="ComponentIntegral-Relation">
  <rdfs:subClassOf rdf:resource="#PartWhole-Relation"/>
  <pwo:consistsOf rdf:parseType="daml:collection">
    <pwo:Element>
      <pwo:element rdf:resource="#Connection"/>
      <pwo:dependent-elements rdf:parseType="daml:collection">
        <pwo:Element>
          <pwo:element rdf:resource="#Functional"/>
        </pwo:Element>
        <pwo:Element>
          <pwo:element rdf:resource="#Nonhomogeneous"/>
        </pwo:Element>
        <pwo:Element>
          <pwo:element rdf:resource="#Separable"/>
        </pwo:Element>
      </pwo:dependent-elements>
    </pwo:Element>
  </pwo:consistsOf>
</daml:Class>

<daml:Class rdf:ID="MemberBunch-Relation">
  <rdfs:subClassOf rdf:resource="#PartWhole-Relation"/>
  <pwo:consistsOf rdf:parseType="daml:collection">
```

```
    <pwo:Element>
      <pwo:element rdf:resource="#Connection"/>
      <pwo:dependent-elements rdf:parseType="daml:collection">
        <pwo:Element>
          <pwo:element rdf:resource="#NonFunctional"/>
        </pwo:Element>
        <pwo:Element>
          <pwo:element rdf:resource="#Nonhomogeneous"/>
        </pwo:Element>
        <pwo:Element>
          <pwo:element rdf:resource="#Separable"/>
        </pwo:Element>
      </pwo:dependent-elements>
    </pwo:Element>
  </pwo:consistsOf>
</daml:Class>

<daml:Class rdf:ID="MemberPartnership-Relation">
  <rdfs:subClassOf rdf:resource="#PartWhole-Relation"/>
  <pwo:consistsOf rdf:parseType="daml:collection">
    <pwo:Element>
      <pwo:element rdf:resource="#Connection"/>
      <pwo:dependent-elements rdf:parseType="daml:collection">
        <pwo:Element>
          <pwo:element rdf:resource="#NonFunctional"/>
        </pwo:Element>
        <pwo:Element>
          <pwo:element rdf:resource="#Nonhomogeneous"/>
        </pwo:Element>
        <pwo:Element>
          <pwo:element rdf:resource="#NonSeparable"/>
        </pwo:Element>
      </pwo:dependent-elements>
    </pwo:Element>
  </pwo:consistsOf>
</daml:Class>

<daml:Class rdf:ID="PlaceArea-Relation">
  <rdfs:subClassOf rdf:resource="#PartWhole-Relation"/>
  <pwo:consistsOf rdf:parseType="daml:collection">
```

```
      <pwo:Element>
        <pwo:element rdf:resource="#Connection"/>
        <pwo:dependent-elements rdf:parseType="daml:collection">
          <pwo:Element>
            <pwo:element rdf:resource="#Functional"/>
          </pwo:Element>
          <pwo:Element>
            <pwo:element rdf:resource="#Homogeneous"/>
          </pwo:Element>
          <pwo:Element>
            <pwo:element rdf:resource="#NonSeparable"/>
          </pwo:Element>
        </pwo:dependent-elements>
      </pwo:Element>
    </pwo:consistsOf>
  </daml:Class>

  <daml:Class rdf:ID="PortionObject-Relation">
    <rdfs:subClassOf rdf:resource="#PartWhole-Relation"/>
    <pwo:consistsOf rdf:parseType="daml:collection">
      <pwo:Element>
        <pwo:element rdf:resource="#Connection"/>
        <pwo:dependent-elements rdf:parseType="daml:collection">
          <pwo:Element>
            <pwo:element rdf:resource="#Functional"/>
          </pwo:Element>
          <pwo:Element>
            <pwo:element rdf:resource="#Homogeneous"/>
          </pwo:Element>
          <pwo:Element>
            <pwo:element rdf:resource="#Separable"/>
          </pwo:Element>
        </pwo:dependent-elements>
      </pwo:Element>
    </pwo:consistsOf>
  </daml:Class>

</rdf:RDF>
```

Appendix B

Spatial relations upper ontology

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
  xmlns:xsd="http://www.w3.org/2000/10/XMLSchema#"
  xmlns:so="http://mach.vub.ac.be/~bpellens/SpatialOntology.daml#"
  xmlns="http://mach.vub.ac.be/~bpellens/SpatialOntology.daml#">

  <daml:Ontology rdf:about="">
  <daml:versionInfo>SpatialOntology,2003/04/25 </daml:versionInfo>
  <daml:imports rdf:resource="http://www.daml.org/2001/03/daml+oil"/>
  </daml:Ontology>

  <daml:Class rdf:ID="PropertyValue"/>
  <daml:Property rdf:ID="hasProperty">
    <daml:domain rdf:resource="<<daml+oil>>#Thing"/>
    <daml:range rdf:resource="#PropertyValue"/>
  </daml:Property>

  <daml:Class rdf:ID="FrameOrientation">
    <rdfs:subClassOf rdf:resource="#PropertyValue"/>
    <rdfs:subClassOf>
      <daml:Restriction cardinality="1">
        <daml:onProperty rdf:resource="#front-to-back"/>
      </daml:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <daml:Restriction cardinality="1">
        <daml:onProperty rdf:resource="#side-to-side"/>
      </daml:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <daml:Restriction cardinality="1">
        <daml:onProperty rdf:resource="#top-to-bottom"/>
      </daml:Restriction>
    </rdfs:subClassOf>
  </daml:Class>

  <daml:Property rdf:ID="front-to-back">
    <daml:range rdf:resource="#Coordinate"/>
  </daml:Property>
  <daml:Property rdf:ID="side-to-side">
    <daml:range rdf:resource="#Coordinate"/>
  </daml:Property>
  <daml:Property rdf:ID="top-to-bottom">
    <daml:range rdf:resource="#Coordinate"/>
  </daml:Property>

  <daml:Property rdf:ID="orientation">
    <daml:subPropertyOf rdf:resource="#hasProperty"/>
    <daml:range rdf:resource="#FrameOrientation"/>
  </daml:Property>

  <daml:Class rdf:ID="Coordinate">
    <rdfs:subClassOf rdf:resource="#PropertyValue"/>
    <rdfs:subClassOf>
      <daml:Restriction cardinality="1">
        <daml:onProperty rdf:resource="#x-coordinate"/>
      </daml:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <daml:Restriction cardinality="1">
        <daml:onProperty rdf:resource="#y-coordinate"/>
      </daml:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <daml:Restriction cardinality="1">
        <daml:onProperty rdf:resource="#z-coordinate"/>
      </daml:Restriction>
    </rdfs:subClassOf>
  </daml:Class>

  <daml:DatatypeProperty rdf:ID="x-coordinate">
    <daml:range rdf:resource="<<XMLSchema>>#decimal"/>
  </daml:DatatypeProperty>
  <daml:DatatypeProperty rdf:ID="y-coordinate">
    <daml:range rdf:resource="<<XMLSchema>>#decimal"/>
  </daml:DatatypeProperty>
  <daml:DatatypeProperty rdf:ID="z-coordinate">
    <daml:range rdf:resource="<<XMLSchema>>#decimal"/>
  </daml:DatatypeProperty>

  <daml:Property rdf:ID="origin">
  <daml:subPropertyOf rdf:resource="#hasProperty"/>
    <daml:range rdf:resource="#Coordinate"/>
  </daml:Property>

  <daml:Class rdf:ID="Handedness">
    <rdfs:subClassOf rdf:resource="#PropertyValue"/>
  </daml:Class>

  <Handedness rdf:ID="LeftHanded"/>
  <Handedness rdf:ID="RightHanded"/>

  <daml:Property rdf:ID="handedness">
    <daml:subPropertyOf rdf:resource="#hasProperty"/>
    <daml:range rdf:resource="#Handedness"/>
  </daml:Property>

  <daml:Class rdf:ID="ReferenceFrame">
    <rdfs:subClassOf>
      <daml:Restriction cardinality="1">
        <daml:onProperty rdf:resource="#origin"/>
      </daml:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
```

```
      <daml:Restriction cardinality="1">
        <daml:onProperty rdf:resource="#orientation"/>
      </daml:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <daml:Restriction cardinality="1">
        <daml:onProperty rdf:resource="#handedness"/>
      </daml:Restriction>
    </rdfs:subClassOf>
  </daml:Class>

  <daml:Class rdf:ID="SpatialObject">
    <rdfs:subClassOf rdf:resource="#PropertyValue"/>
  </daml:Class>

  <daml:Class rdf:ID="Relation"/>

  <daml:Class rdf:ID="Spatial-Relation">
    <rdfs:subClassOf rdf:resource="#Relation"/>
    <rdfs:subClassOf>
      <daml:Restriction cardinality="1">
        <daml:onProperty rdf:resource="#primary-object"/>
      </daml:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <daml:Restriction minCardinality="1">
        <daml:onProperty rdf:resource="#reference-object"/>
      </daml:Restriction>
    </rdfs:subClassOf>
  </daml:Class>

  <daml:Property rdf:ID="primary-object">
    <daml:subPropertyOf rdf:resource="#hasProperty"/>
    <rdfs:range rdf:resource="#SpatialObject"/>
  </daml:Property>
  <daml:Property rdf:ID="reference-object">
    <daml:subPropertyOf rdf:resource="#hasProperty"/>
    <rdfs:range rdf:resource="SpatialObject"/>
  </daml:Property>
  <daml:Property rdf:ID="type">
    <daml:subPropertyOf rdf:resource="#hasProperty"/>
  </daml:Property>
  <daml:Property rdf:ID="reference-frame">
    <daml:subPropertyOf rdf:resource="#hasProperty"/>
    <rdfs:range rdf:resource="#ReferenceFrame"/>
  </daml:Property>

  <daml:Class rdf:ID="Orientation">
    <rdfs:subClassOf rdf:resource="#PropertyValue"/>
  </daml:Class>
  <daml:Property rdf:ID="orientational-type">
    <daml:subPropertyOf rdf:resource="#type"/>
    <rdfs:range rdf:resource="#Orientational"/>
  </daml:Property>

  <Orientational rdf:ID="LeftOf"/>
  <Orientational rdf:ID="RightOf"/>
  <Orientational rdf:ID="Before"/>
  <Orientational rdf:ID="After"/>
  <Orientational rdf:ID="Above"/>
  <Orientational rdf:ID="Below"/>
  <Orientational rdf:ID="MiddleOf"/>

  <daml:Class rdf:ID="Direction">
    <rdfs:subClassOf rdf:resource="#PropertyValue"/>
  </daml:Class>
  <daml:Property rdf:ID="directional-type">
    <daml:subPropertyOf rdf:resource="#type"/>
    <rdfs:range rdf:resource="#Direction"/>
  </daml:Property>

  <Directional rdf:ID="NorthOf"/>
  <Directional rdf:ID="SouthOf"/>
  <Directional rdf:ID="WestOf"/>
  <Directional rdf:ID="EastOf"/>
  <Directional rdf:ID="Up"/>
  <Directional rdf:ID="Down"/>

  <daml:Class rdf:ID="Topology">
    <rdfs:subClassOf rdf:resource="#PropertyValue"/>
```

```
</daml:Class>
<daml:Property rdf:ID="topological-type">
  <daml:subPropertyOf rdf:resource="#type"/>
  <rdfs:range rdf:resource="#Topology"/>
</daml:Property>

<Topology rdf:ID="DisjointWith"/>
<Topology rdf:ID="Touches"/>
<Topology rdf:ID="Inside"/>
<Topology rdf:ID="Equals"/>

<daml:DatatypeProperty rdf:ID="distance">
  <rdfs:range rdf:resource="<<XMLSchema>>#decimal"/>
</daml:DatatypeProperty>

<daml:Class rdf:ID="Orientational-Relation">
  <rdfs:subClassOf rdf:resource="#Spatial-Relation"/>
  <rdfs:subClassOf>
    <daml:Restriction minCardinality="1" maxCardinality="3">
      <daml:onProperty rdf:resource="#orientational-type"/>
    </daml:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <daml:Restriction cardinality="1">
      <daml:onProperty rdf:resource="#distance"/>
    </daml:Restriction>
  </rdfs:subClassOf>
</daml:Class>

<daml:Class rdf:ID="Directional-Relation">
  <rdfs:subClassOf rdf:resource="#Spatial-Relation"/>
  <rdfs:subClassOf>
    <daml:Restriction minCardinality="1" maxCardinality="3">
      <daml:onProperty rdf:resource="#directional-type"/>
    </daml:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <daml:Restriction cardinality="1">
      <daml:onProperty rdf:resource="#distance"/>
    </daml:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <daml:Restriction cardinality="1">
        <daml:onProperty rdf:resource="#reference-frame"/>
      </daml:Restriction>
    </rdfs:subClassOf>
  </daml:Class>

  <daml:Class rdf:ID="Topological-Relation">
    <rdfs:subClassOf rdf:resource="#Spatial-Relation"/>
    <rdfs:subClassOf>
      <daml:Restriction minCardinality="1" maxCardinality="3">
        <daml:onProperty rdf:resource="#topological-type"/>
      </daml:Restriction>
    </rdfs:subClassOf>
  </daml:Class>
</rdf:RDF>
```

Appendix C

Source code of the C++ objects

C.1 Primitive Object

```
class Sphere : public Object {
public:
  /** Constructor & destructor */
  Sphere(string id);
  ~Sphere();

  virtual void draw();

  // Additional inspectors and mutators
  // ...
private:
  string id_, color_;
  float xpos_, ypos_, zpos_, radius_, density_;
  bool state_;
};
```

C.2 Composite Object

```
class ComposedObject : public Object {
public:
  /** Constructor & destructor */
  ComposedObject(string id);
  ~ComposedObject();

  virtual void draw();

  // Additional inspectors and mutators
  // ...
private:
  string id_;
  list<Relation*> parts_;
};
```

Bibliography

[1] Jeremy Rogers Alan. Galen's model of parts and wholes: Experience and comparisons. *Journal of the American Medical Informatics Association, Fall Symposium Special Issue*, pages 819–823, November 2000.

[2] E. Andre, G. Herzog, and T. Rist. Natural language access to visual data: Dealing with space and movement. Technical Report 63, Universitat des Saarlandes, 1989.

[3] Alessandro Artale, Enrico Franconi, Nicola Guarino, and Luca Pazzi. Part-whole relations in object-centered systems: An overview. *Data and Knowledge Engineering*, 20(3):347–383, 1996.

[4] Ekkehard Beier. Conceptual 3d modeling. Habilitationsschrift, Technischen Universitat Ilmenau, 1999.

[5] Wesley Bille. From knowledge representations to virtual environments. Licentiate's thesis, Vrije Universiteit Brussel, 2002.

[6] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, and Eve Maler. *Extensible Markup Language (XML) 1.0 (Second Edition)*, 2000.

[7] Jane Brennan and William Wilson. An approach to formalising relationships between speaker-relative and absolute spatial reference systems. Technical Report 9908, University of New South Wales, 1999.

[8] A G Cohn and S M Hazarika. Qualitative spatial representation and reasoning: An overview. *Fundamenta Informaticae*, 46(1-2):1–29, 2001.

[9] Robert M. Colomb. Use of upper ontologies for interoperation of information systems: A tutorial. Technical report, ISIB-CNR, 2002.

[10] Dan Connolly, Frank van Harmelen, Ian Horrocks, Deborah McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. *Annotated DAML+OIL Ontology Markup*, 2001.

[11] The Defense Advanced Research Projects Agency DARPA. Daml+oil project. http://www.daml.org/2001/03/daml+oil.daml, 2000.

[12] M. Egenhofer and R. Franzosa. Point-set topological spatial relations. *Int. J. of Geographical Information Systems*, 5(2):161–174, 1991.

[13] Stephen Ellis. What are virtual environments? *IEEE Computer Graphics and Applications*, 14(1):17–22, 1994.

[14] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer Graphics: Principles and Practice.* Addison-Wesley, 2nd edition, 1995.

[15] Andrew U. Frank. Qualitative spatial reasoning: Cardinal directions as an example. *International Journal of Geographical Information Science*, 10(3):269–290, 1996.

[16] Andrew U. Frank. Formal models for cognition — taxonomy of spatial location description and frames of reference. *Lecture Notes in Computer Science*, 1404:293–314, 1998.

[17] Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.

[18] Nicola Guarino. Semantic matching: Formal ontological distinctions for information organization, extraction, and integration. In *SCIE*, volume 1299, pages 139–170. Springer, 1997.

[19] Nicola Guarino and Pierdaniele Giaretta. Ontologies and knowledge bases: Towards a terminological clarification. In *Towards Very Large Knowledge Bases.* IOS Press, 1995.

[20] Annette Herskovits. *Language and spatial cognition: An interdisciplinary study of the prepositions in English.* Cambridge University Press, 1986.

[21] Ora Lassila and Ralph R. Swick. *Resource Description Framework (RDF) Model and Syntax Specification*, 1999.

[22] Tomasz Mazuryk and Michael Gervautz. Virtual reality - history, applications, technology and future, 1996.

[23] Winston Morton, Roger Chaffin, and Douglass Herrmann. A taxonomy of part-whole relations. *Cognitive Science*, pages 417–444, 1987.

[24] I. Niles and A. Pease. Origins of the ieee standard upper ontology. working notes of the ijcai-2001 workshop on the ieee standard upper ontology, seattle, 2001.

[25] James J. Odell. Six different kinds of composition. *Journal of Object-Oriented Programming*, 5(8), January 1994.

[26] Dimitris Papadias and Theodoros Andronikos. Relation-based information processing with symbolic spatial indexes. In *IGIS*, pages 288–291, 1994.

[27] U. Sattler. Description logics for the representation of aggregated objects. In W.Horn, editor, *Proceedings of the 14th European Conference on Artificial Intelligence*. IOS Press, Amsterdam, 2000.

[28] Rainer Schubert and Karl Heinz Hohne. Partonomies for interactive explorable 3d-models of anatomy. In Christopher G. Chute, editor, *A Paradigm Shift in Health Care Information Systems: Clinical Infrastuctures for the 21st Century, Proc. 1998 AMIA Annual Fall Symposium*, pages 433–437, 1998.

[29] Peter Simons. *Parts, a Study in Ontology.* Oxford University Press, Walton Street, Oxford, 1987.

[30] Russel Smith. Open dynamics engine. http://opende.sourceforge.net/, 2001-2003.

[31] Eva Stopp, Klaus-Peter Gapp, Gerd Herzog, Thomas Laengle, and Tim C. Lueth. Utilizing spatial relations for natural language access to an autonomous mobile agent. In *KI - Kunstliche Intelligenz*, pages 39–50, 1994.

[32] George Thomas and Ross Finney. *Calculus and Analytic Geometry.* Addison Wesley, 9th edition, 1996.

[33] O. De Troyer, W. Bille, R. Romero, and P. Stuer. On generating virtual worlds from domain ontologies. In *Proceedings of the 9th International Conference on Multi-Media Modeling*, Taiwan, 2003.

[34] Barbara Tversky. Remembering spaces. In Endel Tulving and Fergus I. M. Craik, editors, *The Oxford Handbook of Memory*, chapter 23. Oxford University Press, 1st edition, 2000.

[35] M. Uschold and R. Jasper. A framework for understanding and classifying ontology applications. In *Proceedings of the IJCAI99 Workshop on Ontologies and Problem-Solving Methods(KRR5)*, volume 18, August 1999.

[36] Mike Uschold and Michael Grüninger. Ontologies: principles, methods, and applications. *Knowledge Engineering Review*, 11(2):93–155, 1996.

[37] A. Valente and J. Breuker. Towards principled core ontologies. In B.R. Gaines and M. Mussen, editors, *Proceedings of the KAW-96*, 1996.

[38] G. van Heijst, A. Schreiber, and B. Wielinga. Using explicit ontologies in kbs development. *Int. J. of Human-Computer Studies*, 46:183– 292, 1997.

[39] A. Varzi. Parts, wholes, and part-whole relations: The prospects of mereotopology. *Data and Knowledge Engineering*, 20(3):259–286, 1996.

[40] Eric Weisstein. *CRC Concise Encyclopedia of Mathematics.* CRC Press, 2nd edition, 2002.