A constraint-based approach to modelling spatial semantics of vitual environments

Thanh-Hai Trinh

THÈSE / UNIVERSITÉ DE BRETAGNE OCCIDENTALE
sous le sceau de l'Université européenne de Bretagne

pour obtenir le titre de
DOCTEUR DE L'UNIVERSITÉ EUROPÉENNE DE BRETAGNE
Mention : Informatique
École Doctorale SICMA

présentée par
Thanh-Hai Trinh
Préparée au Centre Européen
de Réalité Virtuelle

A Constraint-based Approach to Modelling Spatial Semantics of Virtual Environments

Thèse soutenue le 05 avril 2012
devant le jury composé de :

Guillaume MOREAU
Professeur, Ecole Centrale de Nantes / *rapporteur*

Christophe CLARAMUNT
Professeur, Ecole Navale / *rapporteur*

Philippe MATHIEU
Professeur, Université Lille 1 / *examinateur*

Pierre DE LOOR
Professeur, ENI de Brest / *examinateur*

Pierre CHEVAILLIER
Maître de conférences, ENI de Brest / *examinateur*

Ronan QUERREC
Maître de conférences, ENI de Brest / *examinateur*

A Constraint-based Approach to Modelling Spatial Semantics of Virtual Environments

Thanh-Hai Trinh
{trinh@enib.fr}

Abstract

Within Virtual Reality Environments (VREs), spatial relationships among objects convey fundamental knowledge about the environment, namely direction ("left", "right", "front of"), distance ("near", "far"), topology ("inside", "disjoint"), and projection ("between", "surrounded by"). Modelling spatial relationships is critical in a variety of applications of VREs, such as human learning environments, virtual museums, or navigation-aids systems. However, spatial relationships have been considered as abstract information and thus, difficult to specify.

Addressing this issue, this thesis proposes an approach to model spatial relationships among virtual objects in VREs. First, we formalise a formal model of spatial relationships dedicated to VREs. Second, we provide a language and a framework to specify spatial relationships at a conceptual level. Finally, we apply our model to specify spatial relations in two real applications: Virtual Physics Laboratory – a VRE for learning physics, and BrestCoz – a cultural heritage application for visiting Brest harbour in the 18th century. We claim that the proposed language is a relevant basis to specify spatial constraints related to activities of agents and users within VREs.

Keywords: spatial constraints, spatial languages, semantic virtual environments.

Table of Contents

List of Figures

List of Tables

Introduction

This thesis investigates *spatial semantics* of Virtual Environments (VEs). Spatial semantics are generally expressed by spatial relationships among entities (Zlatev, 2007). Within a VE, spatial relationships convey fundamental spatial knowledge about the environment, including direction, distance, or topology. Modelling spatial relationships is critical in a variety of applications of VEs, such as human learning environments, virtual museums, or navigation-aids systems.

We propose an approach to model spatial semantics of VEs based on *semantic modelling* (Latoschik and Blach, 2008). We formalise a semantic model of spatial relationships dedicated to VEs, and provide a language and a framework to specify spatial relationships at a conceptual level.

In this introductory chapter, Section 1 provides a brief introduction to semantic modelling and spatial semantics that motivate the thesis. We focus at analysing the complexity and issues related to the modelling of spatial semantics of VEs. The research aim and objectives are set out in Section 2. We give an overview of our approach in Section 3 and describe our main contributions in Section 4. The scope of this research is presented in Section 5. Section 6 summarises the overall structure of the thesis.

1 Problem Statement

1.1 Semantic Modelling of Virtual Environments

Semantic modelling implies the process to closely capture the meaning of user's data and provide a concise, high-level description of this information. A *semantic model* should offer a simple, natural, implementation-independent, flexible, and non-redundant specification of information (Rishe, 1992). Semantic modelling has been studied and successfully applied in several scientific domains, such as databases (Bouzeghoub and Métais, 1991), multimedia databases (Al-Khatib et al., 1999), Geographical Information Systems (Egenhofer, 2002), and World Wide Web (Lee et al., 2001).

Semantic modelling is specially relevant for VEs. The introduction of a semantic model into a VE can be justified from several perspectives.

- Promoting intelligent VEs. A semantic model is becoming essential for an important class of VEs that response to users' interactions in an *intelligent* manner. Intelligence of a VE implies its abilities to exhibit human-like be-

haviours, and to assist users to solve specific problems, rather than relying entirely on users' knowledge and skills (Aylett and Luck, 2000). For example, in a VE for training, the learner can receive educational assistances to achieve a pedagogical goal. Similarly, in a virtual tour, a virtual agent explains information about exhibited objects to visitors. These intelligent behaviours intensively require a semantic model that provides information about virtual objects, such as their types, their structures, their states, their relationships with others, and the operations that one can apply on them. Based on a semantic model, it is possible to incorporate Artificial Intelligent methods into VEs, such as to explain causal behaviours, to carry out path planning, or to perform high-level reasoning (Aylett and Cavazza, 2001).

- Increasing the reusability and adaptability of VEs. A VE is often tailored for a particular requirement. Traditionally, the construction of a VE consists in building its graphical model. A graphical model describes how a VE is structured in terms of geometric primitives and scene graph using standard formats such as VRML (Virtual Reality Modeling Language) or X3D (eXtensible 3D). Consequently, VEs are defined at a low level and platform-dependent. In contrast, a semantic model increases the separation of conceptual and physical representations of a VE. A semantic model of a VE can be used for many purposes, such as for interactions, animations, and visualisations (Gutierrez, 2005).

However, the semantic modelling of VEs has to cope with a number of difficulties. Currently, there exists no standard (not even a *de facto* standard) for the semantic modelling of VEs. No common language for semantic modelling has been proposed. Existing work attempts either to add semantic annotations to existing VEs, or to build VEs based on a pre-existent semantic level (e.g., using Geographical Information Systems to build urban VEs). There exists a few work that introduces semantics into the design phase of VEs (Ibáñez and Delgado-Mata, 2011). In this case, semantic modelling is mainly grounded on abstraction mechanisms that introduce further representation and implementation issues. Indeed, the semantic modelling of VEs still lacks both a conceptual and technical base (Latoschik and Blach, 2008).

Because of the above critical observations, the semantic modelling of VEs is challenging. The construction of semantic VEs is an interesting problem and largely an open research area.

1.2 Spatial Semantics and Its Complexity

VEs are spatial in nature. As a consequence, spatial semantics is of central importance for the semantic representation of VEs.

Spatial semantics is expressed by spatial expressions that specify the location or change of location of an entity in terms of its relationship to a second entity

in space (Zlatev, 2007). According to Hernández (1994), spatial expressions can be generally formalised using a quadruplet as follows:

$< primary\_obj, rel, ref\_obj, ref\_frame >$

In this generic formalisation, $primary\_obj$ is a primary object, corresponding to a target object. ref_object is a reference object, corresponding to an already located object or a landmark. rel represents a spatial relationship between the primary object and the reference object. The relationship is given with respect to ref_frame – a frame of reference that defines the context of the relationship. In this thesis, we shall investigate spatial semantics strictly under this definition.

The main motivations to introduce spatial semantics into VEs are as follows.

- First, spatial relationships convey many semantics about an environment. A spatial semantic model allows to specify spatial relationships such as "inside", "between", "near", "far", "left", "right", or "in front of". This spatial knowledge is critical in many applications of VEs, range from VEs for training, virtual tours, to navigation-aids systems. Furthermore, spatial relationships are used in everyday communication and human activities. An explicit representation of spatial relationships in VEs can be exploited (e.g., by virtual agents) to produce human-like behaviours, or to assist users in many tasks, such as searching an item, or localising an object in space.

- Second, previous research in spatial cognition (Waller et al., 2000), spatial navigation (Darken and Peterson, 2001), and the transfer of spatial knowledge from a virtual to a real environment (Durlach et al., 2000) has made evidence that spatial relationships constitute a human cognitive (or mental) representation of an environment. The cognitive model is the highest stage of development of an individual's cognitive representation of space (Waller et al., 1998). This provides a more flexible and abstract representation of the environment than other initial ones, for instance landmark-based description. The transition from an initial to a cognitive representation of an environment is, before all, the most important issue of research in VEs.

Nevertheless, spatial relationships are inherently complex. The prominent questions in modelling spatial semantics of VEs can be stated as follows.

- Spatial relations are manifold.
  To express spatial relations, there exists a variety of spatial prepositions. Spatial prepositions can be gathered into groups. Each group of spatial prepositions corresponds to a specific aspect of space. For instance, a spatial relation can be related to direction ("the chair is in front of the table"), distance ("the chair is located 1 meter from the table"), or topology ("the drawer is inside the table").

- Spatial relations are qualitative.
  In daily communication and human activities, spatial relations are often given in an *imprecise* and *incomplete* manner. One gets used to saying that "the table is on the left of the window", instead of "the table is 90 degrees from the view of the window". That is to say, spatial relationships are described in a *qualitative* way.

- Within VEs, spatial relations are multi-dimensional and ambiguous.
  In 2D spaces, spatial entities are points or regions that are often represented using a vector-based approach. Whereas, in 3D spaces, spatial relations exist among 3D entities that are built upon 3D complex shapes. The *multidimensionality* of 3D spaces significantly increases the complexity of spatial semantics. Another difficulty is that spatial relations can contain *ambiguities*. A VE is a mixed community of virtual agents and users. Different points of view of agents or users can yield different observations to a spatial relation.

- Within VEs, spatial relations are abstract.
  Some of existing work attempted to enrich VEs with spatial annotations (Bowman et al., 2003; Kleinermann et al., 2008). However, spatial relationships are still classified as *abstract* information and thus, difficult to specify. To indicate spatial relations, the existing approaches merely use multimedia landmarks such as texts, videos, images, or Web links. For example, a left-arrow associated with a text is used to indicate a "left" relationship. To the best of our knowledge, no previous work has dealt with a semantic model of spatial relationships for VEs.

2 Aim and Objectives

From the problem statement, the aim of this thesis is to investigate the modelling of spatial relationships among virtual objects of VEs.

To carry out our investigation, we set out the following research objectives:

- to identify and model spatial concepts (e.g., spatial object and reference frame) needed for the specification of spatial relationships in VEs
- to identify the main families of spatial relationships and to propose an appropriate model for describing them in VEs
- to develop a modelling language that enables a high-level, consistent, and unambiguous specification of spatial relationships in VEs
- to develop algorithms for computing spatial relationships in VEs
- to develop methods for visualising, and thus materialising, abstract spatial relationships in VEs
- to utilise the model of spatial relationships to specify spatial activities of virtual agents and users in VEs

3 Approach Overview

This thesis proposes an approach to model spatial relationships among virtual objects in VEs. We particularly focus on modelling, computing, and visualising spatial relationships. The original characteristics of our approach are as follows:

- Constraint-based: In our model, spatial relationships are represented in terms of *spatial constraints*[1]. The motivation is that a human often depicts spatial relations among objects in the form of constraints (Renz and Nebel, 2007). Spatial constraints can be unary ("the length of the table must not exceed 2 metres"), binary ("the cube must be on the table"), ternary ("the chair must be placed between the table and the window"), or n-ary in general. A constraint-based representation of spatial relations has several obvious advantages. On the one hand, being represented by spatial constraints, spatial relations can be used in more various contexts. For example, spatial relations can be used as system invariants ("the chair is *always* between the table and the window"). Otherwise, spatial relations can be used as a goal or a checking condition for a user's interaction ("the user *must* put the cube on the table"). In the other words, spatial constraints can be used as pre- and post-conditions of the user's actions. The satisfaction or violation of spatial constraints allow to enable/disable the user's actions. On the other hand, a constraint-based approach facilitates high-level manipulations with spatial knowledge, such as constraint queries, or constraint-based reasoning.

- Model-based: We follow a model-based approach as defined in (OMG, 2011), i.e., models are used to direct the understanding and conceptualising of spatial entities as well as spatial constraints among them. This is done by extending an existing model-based approach, named MASCARET, for semantic modelling of VEs (Chevaillier et al., 2011). In MASCARET, the Unified Modeling Language (UML) serves as the common language to model VEs. MASCARET offers three levels of modelling: the meta level, the conceptual level, and the instance level. VEs are first designed at the conceptual level, and then are instantiated and executed at the instance level. The MASCARET's meta-model is dedicated to VEs and allows to introspect the conceptual model and the instance model at runtime. In our approach, we intervene in MASCARET both at the conceptual and meta levels. First, we enrich the meta-level with a meta-model of spatial constraints. We then formalise a spatial language, named VRX-OCL, to express, and query, spatial constraints at the conceptual level. The language is defined as a Virtual Reality eXtension of the Object Constraint Language that comes along with UML to express constraints in UML-based conceptual models (OMG, 2006).

The sequential steps in our approach are the following.

[1] For this reason, in our model, the terms "spatial relations" and "spatial constraints" are used indistinctly.

Formalising The Concepts of Spatial Entity and Frame of Reference in VEs

Previously, spatial expressions were formalised as a quadruplet involving a primary entity, a reference entity, a relationship, and a frame of reference. Consequently, prior to defining spatial relationships, it is necessary to represent the concepts of spatial entity and frame of reference. As highlighted in (Casati and Varzi, 1997), every model of spatial representation and reasoning must be combined with a manner for defining spatial entities and frames of reference.

We thus propose a conceptual model of spatial entity and frame of reference. Our model of spatial entities is an abstract representation of scene's entities, with additional information to define spatial constraints. Our model of frame of reference allows to clarify the contexts in which spatial constraints will be evaluated. We model different types of frame of reference that can be implicit or explicit, egocentric (i.e., first-person perspective) or allocentric (i.e., third-person perspective).

Elaborating a Formal Model of Spatial Relationships Among Spatial Entities

Based on the conceptual definitions of spatial entity and frame of reference, we elaborate a formal model of spatial relationships. By surveying existing spatial models, we classify and identify the four main families of spatial constraints, namely topological, projective, directional, and distance constraints. Topological constraints and projective constraints cover the non-metric aspects of VEs (i.e., topology and projection). Meanwhile, directional constraints and distance constraints cover the metric aspects of VEs (i.e., direction and distance).

To cope with the complexity of spatial semantics, our approach is mainly based on a qualitative description of spatial constraints (Cohn and Renz, 2008). An important motivation for such a qualitative description is that it has been shown to be closer to human mental model of space. As a result, our model is able to take into account the natural imprecision and incompleteness of spatial knowledge.

Formalising an Architecture and a Constraint Language for Conceptualising and Querying Spatial Constraints

Based on our formal model of spatial constraints, we propose Vrx-Ocl as a formal language to conceptualise spatial constraints. Vrx-Ocl is both a constraint and query language. That is, Vrx-Ocl can be used both to specify and query spatial constraints in a conceptual model of a VE. In our context, we use the Mascaret framework to design the conceptual model of a VE, and then use Vrx-Ocl to specify spatial constraints within the conceptual model.

We design the Vrx-Ocl language at two levels: the conceptual level and the meta level. At the conceptual level, Vrx-Ocl specifies spatial constraints in terms of spatial expressions. Vrx-Ocl explicitly defines spatial concepts

involved in spatial expressions, such as primary entity, reference entity, and frame of reference. Spatial relationships are defined in VRX-OCL by means of spatial operators. VRX-OCL expressions can be used as invariants, guard conditions for state transitions, or pre-/post-conditions of actions within the conceptual model. Otherwise, VRX-OCL spatial expressions can be used to query spatial relationships between elements in the conceptual model. At the meta level, VRX-OCL comes with a meta-model that explains how the concepts of spatial entity, frame of reference, and spatial constraint are formalised.

Applying the Proposed Constraint Language to Specify Activities of Agents and Users

As the overall goal of a semantic model is to facilitate the design of intelligent VEs, we apply our semantic model of spatial constraints to specify spatial activities of agents and users in VEs. We illustrate our model with two real applications. In Virtual Physics Laboratory (VPLab), a VE for learning physics, spatial constraints are used to describe invariants of learners' interactions, or to indicate a goal that the learners must achieve. In BRESTCOZ, a cultural heritage application, we use spatial constraints to simulate human-like activities of virtual agents. The satisfaction of spatial constraints decides whether an activity can be activated, or it has been already accomplished or not.

4 Contributions

The research presented in this thesis makes several contributions to the field of semantic modelling of VEs. The major advantages, which make this work significant, are the following:

- An integrated model of spatial constraints among spatial entities.
  We propose an integrated model that covers the main spatial aspects of space, including topology, projection, distance, and direction. The model is *formal* and *generic*. Spatial constraints are specified based on the conceptual definition of spatial entity and frame of reference that is independent from any graphical and conceptual framework.

- A computational model for spatial constraints.
  We provide a unified computational model for spatial constraints. The computational model is expressed in the form of algorithms for calculating spatial constraints. Spatial constraints are evaluated based on basic spatial information about spatial entities and thus the computational model can be easily extended to other application domains.

- An architecture and language for conceptualising spatial constraints.
  The conceptualising of spatial constraints is carried out using a model-based architecture. At the meta level, we formalise a meta-model that abstracts the spatial concepts (e.g., spatial entities, frames of reference, and spatial constraints). At the conceptual level, we propose the VRX-OCL language as a constraint and query language of spatial constraints.

The language can be used 1) to be integrated into a semantic modelling framework for conceptualising spatial constraints independent of graphical representations of VEs; 2) to query spatial constraints among virtual objects; and 3) to specify activities of agents and users of VEs. Overall, VRX-OCL allows to express a shared understanding about spatial knowledge among agents and users. The representation of spatial knowledge is thus unambiguous and consistent.

- A meta-model and architecture for visualising spatial constraints.
  In our approach, spatial constraints are no more abstract. Instead, they can be visualised. The visualisation of spatial constraints, termed as *semantic areas*, is specified at the conceptual level.

5 Thesis Scope

This thesis is concerned with semantic modelling of VEs in general, and with spatial semantics in particular. However, semantic modelling is an emerging research interest in the field. Neither a unified modelling method nor a unified semantic model of VEs has been proposed. Instead, each semantic model deals with a specific aspect of VEs. As stated in (Tutenel et al., 2008), a semantic model can enrich a virtual world in many ways, for example by visual semantics (color, size, shape), physical semantics (weight, mass, temperature), or behavioural semantics (services that virtual objects can provide).

The research conducted in this thesis is limited to spatial semantics of VEs. More precisely, we study spatial semantics in terms of spatial relationships among virtual objects within VEs. Nevertheless, spatial semantics will not be studied isolatedly but in relations to other semantics of VEs.

6 Thesis Structure

The remainder of this thesis is organised in three parts.

Part I aims to review related work and present the context of our work. It includes two chapters. Chapter 1 reviews related approaches in the field of semantic modelling of VEs, and then describes the MASCARET approach that is used as the context of our work. Chapter 2 discusses related work of spatial semantics. It presents the relevant models and approaches for conceptual modelling of spatial relationships. The work presented in this part has been published in (Chevaillier et al., 2011).

Part II details our main contributions. It includes two chapters. Chapter 3 presents our integrated model of spatial constraints for VEs. Chapter 4 presents the VRX-OCL language for specifying spatial constraints in VEs. A detailed description of the architecture, syntax, and meta-model of the language is given. The work presented in this part has been published in (Trinh et al., 2010a,b, 2011).

Part III describes the applications of our model. In includes a single Chapter 5. We illustrate how the proposed model is applied in two real applications,

VPLab – a VE for learning physics, and BrestCoz – an application for cultural heritage preservation. The work presented in this part has been published in (Trinh et al., 2008, 2009; Barange et al., 2011).

In Conclusion, we summarise our approach, and give suggestions for future work.

Appendix A includes a list of published papers concerning the Vrx-Ocl approach.

Appendix B contains the grammar of the Vrx-Ocl language.

Part I

Related Work & Context

Chapter 1

Semantic Modelling of Virtual Environments

The purpose of this chapter is twofold. First, it presents the different motivations that have driven the development of semantic modelling for VEs. Second, it describes the context of our work. The conceptual model of spatial semantics we propose in this thesis has been implemented in the context of MASCARET, which is both a framework and a methodology for the semantic modelling of VEs.

This chapter is structured as follows. Section 1.1 reviews prior work in the field of semantic modelling of VEs. The relevant techniques, methodologies, and frameworks for the semantic modelling of VEs are presented. Section 1.2 gives an overview of the MASCARET framework, with a specific enlightenment on the underlying concepts we anchored our model on. Section 1.3 summarises the chapter.

1.1 Semantic Modelling of Virtual Environments

Drawing a model implies that its designer adopts a specific *point of view*, a central concept in the model-driven approach (OMG, 2011). Along the development process of the system, different models are produced, sequentially or in parallel. It ensues that existing models differ from each other by their point of view, their level of abstraction, and the semantics they convey, although they share some basic concepts. Basically, semantic modelling aims to reduce the gap between different representations of subject matters that experts, designers, engineers, and end-users can have on the system (here the virtual environment). The single concepts all these actors actually share are those of the specific domain of the application, not those related to technical aspects.

For instance, a common sense representation of a room is that it is composed of a floor, a ceiling, walls, at least one door, and optionally windows. Inside the room, there may be some pieces of furniture, such as tables and chairs. All these objects have properties such as colour, size, and style. It could be a common

rule that one should seat on chairs, not on tables, and windows can be opened to refresh the air of the room. All the stakeholders involved in the design phase can also agree on what should be the configuration of the room, by defining some relationships among the elements of the room, e.g., "the table stands in front of the window" and "the sofa is between the table and the bookcase". Different combinations can be tested or changed at runtime.

Modelling languages, such as VRML[1] or X3D[2], allow to define the geometries of the objects that belong to the scene. However, they do not aim to support domain specific concepts (here common sense about architecture and domestic habits). In 3D modelling, geometries are defined using primitive shapes and operators. They are then organised along scene graphs. These hierarchical data structures are merely defined to support the rendering and for optimisation purposes. 3D models and scene graphs are computational-dependent models. The way objects and scenes are described does not necessarily make sense for end-users.

The point of view of semantic modelling is that, both designers and end-users *conceptualise* VEs rather than directly build its geometrical and computational representations. Semantic modelling is expected to offer a richer and more expressive representation of VEs. Consequently, it led to an increasing scientific interest to propose relevant formalisms and frameworks for supporting this semantic representation.

1.1.1 Overview of Related Approaches

Initial Approaches

Semantic modelling techniques are rooted in information systems, namely databases, Geographical Information Systems, and world wide web. Data representation is a crucial issue for these systems. Traditional approaches merely focus on techniques that promote efficient storage and data retrieval. In contrast, semantic modelling aims to make data meaningful and consequently machine-processable. The semantic web is a good example of what semantic modelling can provide. By inserting machine-readable metadata, it allows the meaningless information stored on web pages to be processed by algorithms and searched based on their content.

The current development of VEs shares many issues with the information systems mentioned above. To reduce the gap between the user's representation and the low-level geometrical implementation of VEs, there is a strong need for more conceptual and abstract representations of VEs. The semantic modelling of VEs pursues the same objective for Virtual Reality (VR) contents but requires specific knowledge modelling languages.

As noticed by (Ibáñez and Delgado-Mata, 2011), each research group has de-

[1] Virtual Reality Modeling Language, http://www.w3.org/MarkUp/VRML/
[2] eXtensible 3D, http://www.web3d.org/x3d/

veloped its own methodology and framework to design semantic VEs, and there is no standard for semantic modelling of VEs. These authors have identified three main approaches for the semantic modelling of VEs.

- Designing the VE along with the semantic model. The metadata are added in the model as the objects are created. This technique has been used either for content-oriented and system-oriented approaches.
- Building the VE based on a pre-existent semantic level. The main idea of this technique is to get benefits from an existing semantic model. For instance, one can use an existing Geographical Information System to build virtual urban environments.
- Adding semantic annotations to the pre-existent VE. The semantic annotations can be multimedia resources, such as texts, images, sounds, and Web links. In this case, the added information makes sense only for the user, but is not semantically interpreted by the system.

Domain-dependent vs. Domain-independent Semantic Models

Several semantic models of VEs proposed in the literature are specifically dedicated to a specific application domain, such as urban environments (Donikian, 1997; Farenc et al., 1999) and digital heritages (Liu et al., 2006). The advantage of this approach is that all the features of the modelling language are meaningful for the designer and thus sound more concrete. The main disadvantage is that these semantic models are platform-dependent and difficult to extend.

Content-oriented vs. System-oriented Semantic Modelling

Current approaches on semantic modelling can be classified into two main categories, corresponding to the kind of semantics they specifically address: *content-oriented semantics* or *system-oriented semantics*.

Content-oriented semantic modelling aims to define an explicit representation of the scene content. It is often done by mapping graphical contents to their conceptual representation, typically using ontologies (Kalogerakis et al., 2006). It provides a representation of the VE which makes sense for the users. The semantics is mostly about types, properties, structures, and relationships of the scene's entities. The approach is very close to the semantic web. Content-oriented semantic models are thus platform-independent models.

Typically, a VR application implies the coupling between graphical contents and execution systems (i.e., the underlying VR platforms). An execution system is composed of different components, e.g., for rendering, physics, and interaction. *System-oriented semantic modelling* aims to provide the information related to the components of the application. The main goal is to support the coupling between these different components and to keep them independent from each other. The semantic modelling is here motivated by issues regarding system

engineering. This is the approach that Latoschik and Fröhlich (2007) followed with SCIVE which uses the concept of reflection to access the properties of the entities of the application at runtime.

With vs. Without a Conceptual Modelling Phase

Semantic modelling does not exclude other traditional modelling methods for system engineering. Moreover, semantic modelling can be used mutually with them, particularly with conceptual modelling. Developing of VEs has been highlighted as a specialised, time-consuming, and expensive process. By introducing a conceptual modelling phase into the development process of VEs, many obstacles preventing a quick spread of this type of applications could be avoided (De Troyer et al., 2007).

According to (Vanacken et al., 2007), semantic information should be introduced during the conceptual modelling phase of VEs. As we have seen previously, one of the main semantic modelling techniques is to add semantic annotations to existing VEs. This is a time-consuming and tedious task. Moreover, this makes semantic annotations tied to the underlying systems of VEs. Because the conceptual model of the VE is system independent, the introduction of semantics into the conceptual model consequently allows to decouple the semantic layer and the system dependent layer.

Existing Approaches for the Semantic Modelling of Virtual Environments

Based on the above critical remarks, to summarise related approaches on semantic modelling of VEs, we classify them using the following criteria.

C1. Genericity: Is the proposed semantic model generic or application dependent?

C2. Content-oriented semantics: Does the semantic model convey knowledge about the entities of the simulated world?

C3. System-oriented semantics: Does the semantic model convey information about the components of the application that support the simulation of the virtual environment?

C4. Conceptual modelling: Does the semantic model result from a conceptual modelling phase?

Table 1.1 includes a list of approaches to developing semantic VEs that are relevant to our approach. The first column (Approach) includes names and references of the approaches. For each approach, a check mark (✓) at one of the four columns from C1 to C4 implies that one of the main interests of this approach is related to the criterion identified by the column. These approaches will be discussed in the following sections, according to the defined criteria.

Approach	C1	C2	C3	C4
Informed VEs (Farenc et al., 1999)		✓		
Semantic digital heritages (Liu et al., 2006)		✓		
Basic semantic level (Ibáñez and Delgado-Mata, 2006)	✓	✓		
OntologyX3D (Kalogerakis et al., 2006)	✓	✓		
Core and Specific Ontologies (Grimaldo et al., 2006b)	✓	✓		
XML-based ontology (Gutierrez, 2005)	✓	✓		
VR-WISE (De Troyer et al., 2007)	✓	✓		✓
NiMMiT (Vanacken et al., 2007)	✓	✓		✓
Information-rich VEs (Bowman et al., 2003)	✓		✓	
SCIVE (Latoschik and Fröhlich, 2007)	✓		✓	
Semantic class library (Tutenel et al., 2009)	✓		✓	

Table 1.1: Summary of related approaches for the semantic modelling of VEs.

1.1.2 Domain-dependent Semantic Modelling

Semantic Urban Environments

To provide the necessary information for the behavioural simulation of virtual humans, notably navigation tasks in urban environments, Farenc et al. (1999) proposed a solution based on the concept of Informed Virtual Environments. With VUEMS, Donikian (1997) followed a similar approach. The Informed VE is described as the hierarchical decomposition of the urban scene into environmental entities (e.g., streets, sidewalks, or buildings), providing both geometrical information and semantic concepts (*cf.* Figure 1.1). For each entity, the database contains information about its name, type, and location in the scene. Each entity may have some entry or exit points. Entities may have links to other entities, such as ascendant and descendant entities. For example, descendant entities of a junction may be a building, a traffic zone, or a non traffic zone.

To interact with environment entities in Informed VEs, environment entities are built upon *Smart Objects* (Kallmann and Thalmann, 1999). A smart object is associated with predefined possible behaviours. It is possible to help users in how to interact with it. For example, a lift in a building represented as a smart objects is able to show users where to activate the button for descending or raising.

Semantic Digital Heritages

Virtual cultural heritage applications share many common points with virtual urban environments. Besides the realistic 3D graphics modelling of ancient architecture, semantic information is needed to provide useful explanations during the user's visit, or to design a virtual guide as an artificial partner for the user.

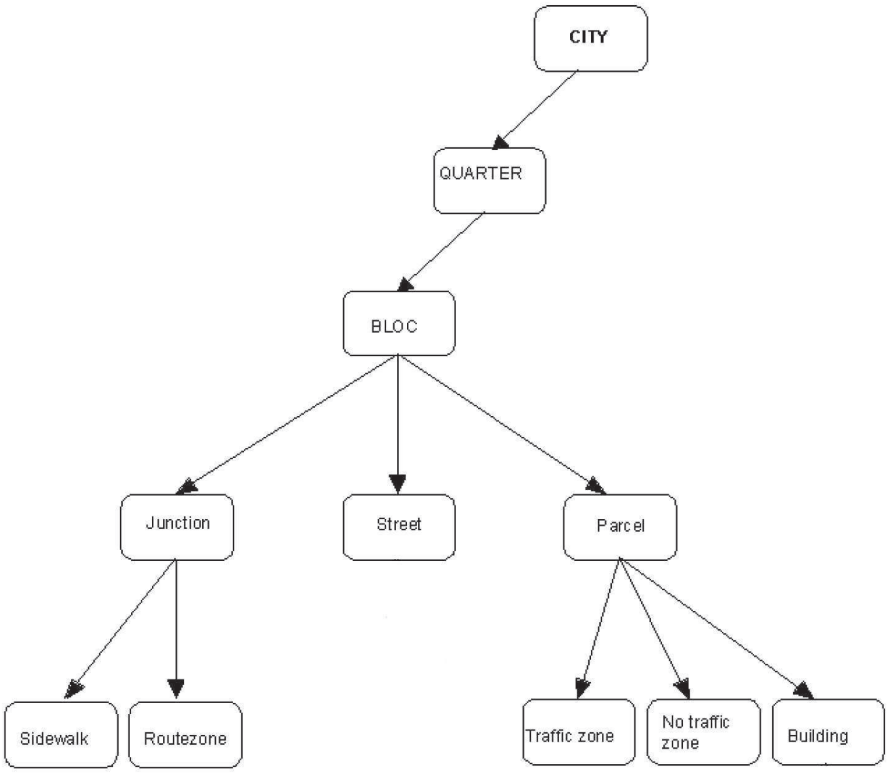


Figure 1.1: Decomposition of a urban environments into environmental entities providing both geometrical information and semantic notions, proposed in (Farenc et al., 1999).

Liu et al. (2006) proposed a semantic approach for designing digital heritages. Figure 1.2 shows the hierarchical model of the city components. In this case, the semantic modelling phase precedes the traditional graphical modelling process. This enables users to transform their modelling plans into the semantic description.

At the implementation level, semantic elements (e.g., city, streets, districts, or houses) are constructed based on graphical basic units. Graphical units and semantic elements can be combined to build higher level semantics elements. For instance, a combination rule is "a house consists of its base, several gates, one roof, more than four walls, and several windows". Furthermore, the topology of the semantic element can be used to define an architectural style. For example, a vernacular house includes a central yard, a shop wall, gate–window wall, two conjunct walls, a small north house, and a small south house. By using semantic modelling, it is then possible to control the generation of ancient architecture with different styles. In addition, information about style or topology of ancient architecture is explicit.

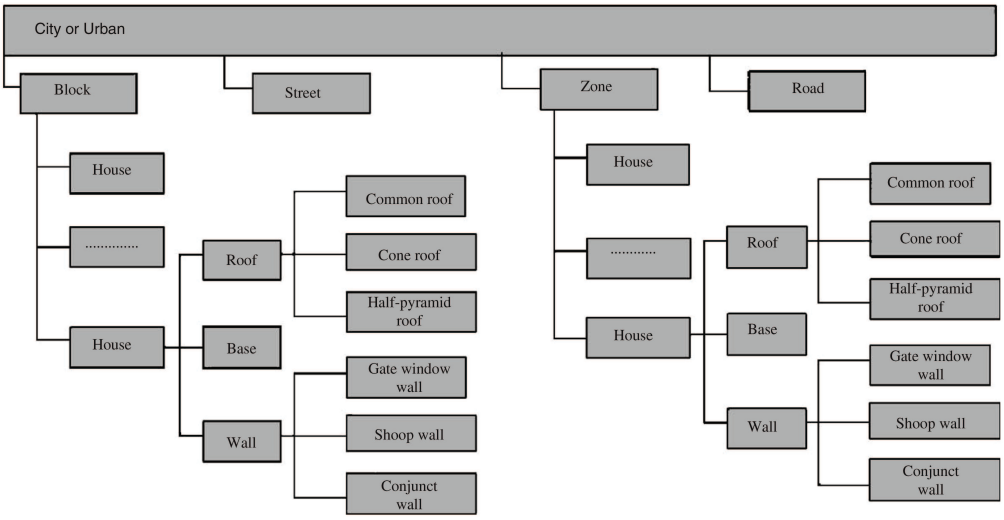


Figure 1.2: A semantic model for constructing digital heritages, proposed in (Liu et al., 2006).

1.1.3 Content-oriented Semantic Modelling

While well motivated by the benefits of a semantic description of VEs, it has been seen previously that the earlier approaches are still based on specialised techniques for specific applications. Therefore, the proposed semantic models are different each from other, and are application-dependent. To overcome this issue, researchers have been interested in defining more generic semantic models of VEs. First, they focus on identifying the basic and common semantics that can be shared among a variety of VEs. Second, they attempt to define a high-level, abstract, and domain-independent representation of 3D scene contents.

Basic and Common Semantics

Ibáñez and Delgado-Mata (2006) defined a basic and common level for the semantic representation of the content of VEs. Their model relies on ten concepts: *object type*, *navigation network*, *object identifier*, *location*, *orientation*, *width*, *height*, *depth*, *spatial containment relations*, and *minimal paths*. The main advantage of this model is that only the first two elements must be manually annotated. The other eight elements can be calculated. From the perspective of spatial relationships, the first eight elements are related to a particular object, whereas the other two elements are related to spatial relationships between objects.

Object type refers to the type of an object according to a particular ontology. Each object is associated with a reachable three-dimensional point, called accessible node. A set of accessible nodes composes a *navigation network* in the VE.

Every object is uniquely identified by an *object identifier*. *Width*, *height*, and

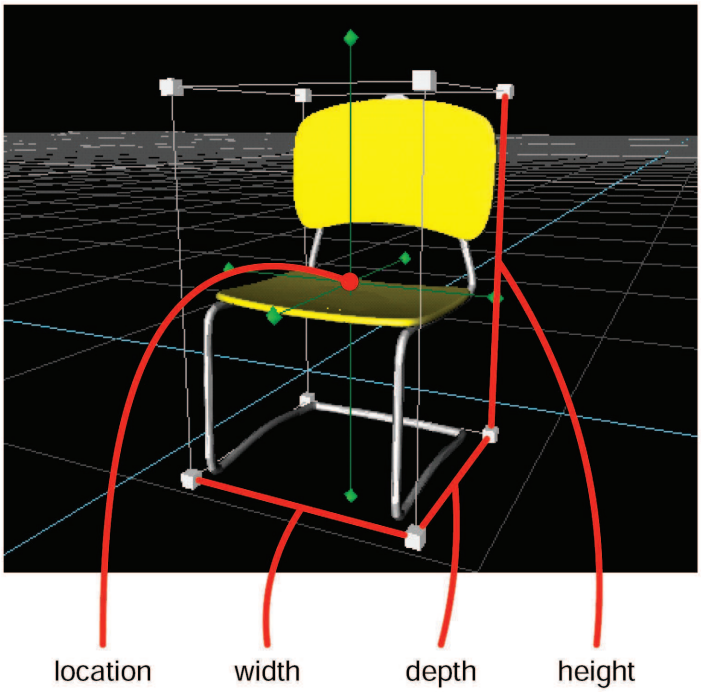


Figure 1.3: Basic and common semantics of virtual environments proposed in (Ibáñez and Delgado-Mata, 2006). Here, the description of a chair is enriched by semantic information such as location, width, depth, and height.

depth are computed based on the bounding box of the object. *Location* of an object is defined as its centroid (*cf.* Figure 1.3). *Orientation* is directly extracted from the geometric model of a VE. Similarly, a *spatial containment relations* (i.e., an object contains another object) is calculated from the geometric model. Because in geometric models given in the form of VRML and X3D formats, each node represents an object, and the organisation of nodes is hierarchical. Thus, spatial containments can be easily obtained. Finally, *minimal paths* represent a shortest path between two arbitrary accessible nodes.

Coupling Virtual Environments with Ontologies

We present here three types of ontological models that provide more complex and abstract descriptions than allowed by the approach presented above.

OntologyX3D

Kalogerakis et al. (2006) proposed to couple ontologies with graphical contents represented by the VRML or X3D standard formats. The graphical contents are mapped onto the ontology, called OntologyX3D, represented by the Ontology Web Language[3] (Owl). To build the OntologyX3D, the X3D nodes

[3]http://www.w3.org/TR/owl-ref/

representing graphical concepts are mapped into OWL classes (*cf.* Figure 1.4). Every OWL class in the OntologyX3D is the abstraction of a graphical resource. It holds information related to geometry and appearance of objects, scene navigation, lights, environmental effects, sound, interpolators (for linear animation), sensors, events, humanoid animation, and geography.

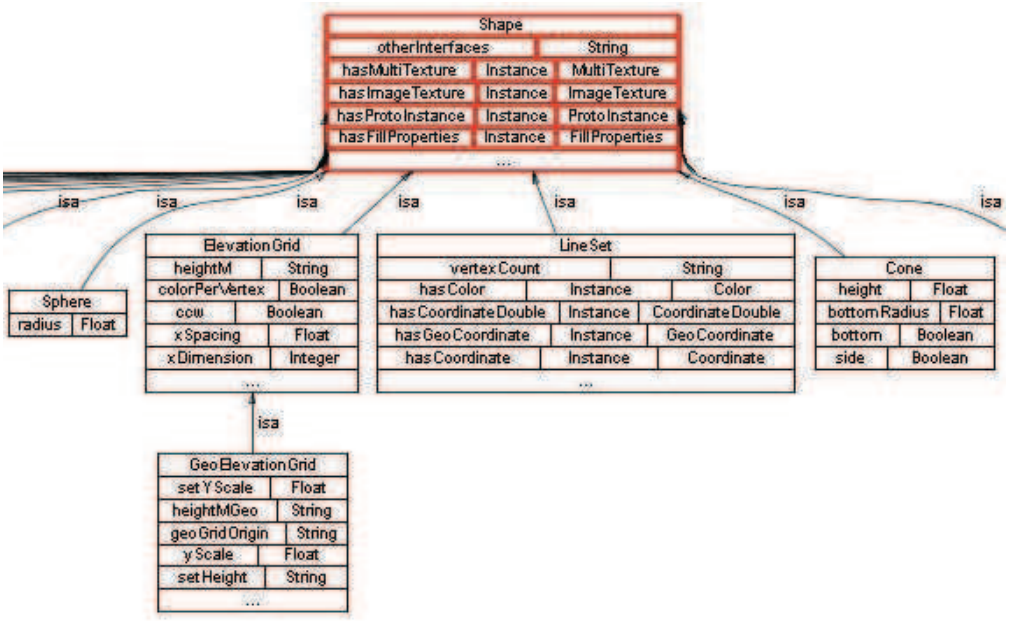


Figure 1.4: A semantic model of a 3D scene obtained by coupling ontology with graphics content, proposed in (Kalogerakis et al., 2006).

Coupling the scene with an ontology offers several important advantages. On the one hand, a semantic description of interrelationships among graphical resources is obtained. Examples of semantic relations are "isPartOf", "generalizationOf", "hasFunction", or "hasBehaviour". On the other hand, the ontology enables agents to perform automated reasoning on the graphics content. For example, OntologyX3D is combined with a rule based inference engine based on the Semantic Web Rule Language[4] (SWRL). Therefore, it is possible to answer to queries for content. For instance, to answer to the query "what is the functionality of this component?", the system first makes the corresponding between the clicked object in the scene and the mapped class in the OntologyX3D, then searches for the "hasFunction" relations of this class.

The SEVEN platform

Following the same principle of coupling graphical contents with ontologies, the SEVEN platform is proposed to design semantic VEs (Otto, 2005a,b). The authors noticed that existing VEs sometimes have the same functionalities (e.g., for visiting museums), but they are not interoperable. For example, a guide

[4] http://www.w3.org/Submission/SWRL/

agent of a museum can not exploit another environment in the same domain. The main goal thus is to develop a semantic model of VEs that contains common and abstract concepts of VEs.

Alternative to OntologyX3D, the SeVen platform uses the Resource Description Framework[5] (RDF) to describe types, properties, and relationships of elements of VEs. By coupling the scene with a RDF graph, this allows a uniform access to entities of heterogeneous environments. Furthermore, as RDF graphs are machine processable, agents can work in different RDF graphs in a transparent manner.

XML-based Ontology

A generic Xml-based semantic model for representing VEs is proposed in (Gutierrez et al., 2005). The semantic model is built around the concept of *Digital Item* that corresponds to each entity in the VE, for example virtual characters or objects (*cf.* Figure 1.5). The organisation of digital item is recursive, i.e., a digital item can contain other digital items. A digital item has different geometrical representations defined in *Geometric Descriptor* that specifies the type of *Shape* associated with the digital item. A *Controller* defines how the digital item behaves, e.g. by autonomous animations or based on user input from a device. At last, a digital item is described by *Semantic Descriptor* that provides human and machine readable information about a particular digital item. Semantic descriptors are extracted from the MPEG-7 standard in the form of Xml descriptors (Manjunath et al., 2002).

The main advantage of this semantic model is to increase the scalability, reusability, and interoperability of VEs. The semantic model of the VE can be used for many purposes, such as animation, interaction, and visualisation (Gutierrez, 2005). For instance, to animate virtual characters both in a handheld device and a large screen with different resolutions, a digital item is associated with two graphical representations with different levels of detail. Based on the semantic descriptor of the digital item, it is possible to choose the correct geometry and interface to present the digital item according to the output device.

Core Ontology vs. Domain Specific Ontologies

Ontologies have been used in the previous approaches as a relevant formalism to provide a conceptual representation of scene contents. The main idea has been a direct mapping between graphical contents and ontologies. The concepts in the ontology must be the exact copy of the specific graphical resources. This led to several inconveniences. First, onotologies can not represent entities with no graphical representation in VEs. Second, it is not possible to share common properties among a family of graphical resources. For instance, movable objects have some properties in common in compared to non-movable objects.

[5] http://www.w3.org/TR/rdf-primer/

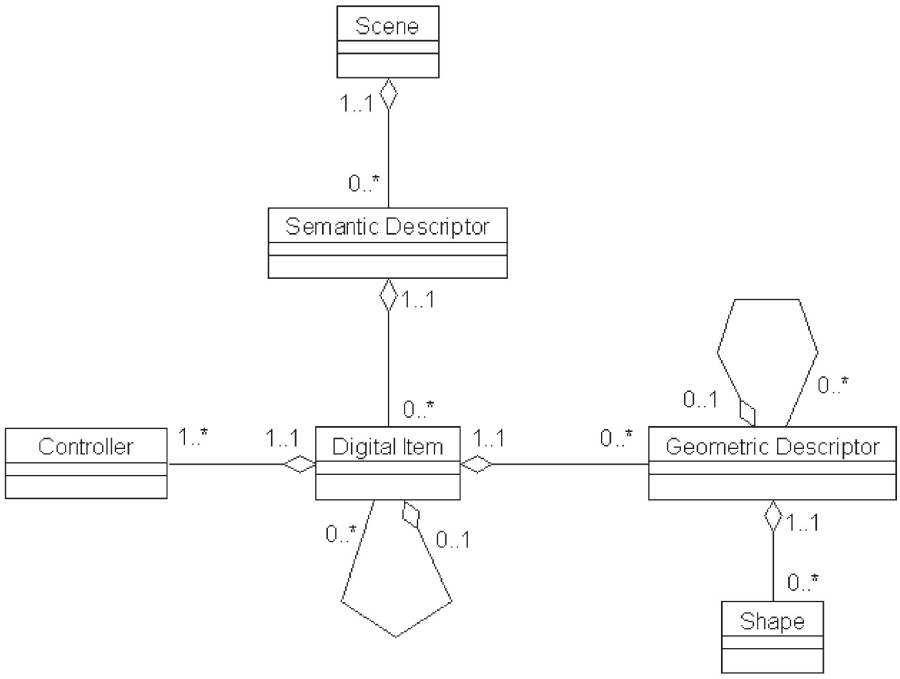


Figure 1.5: XML-based semantic model of virtual environments, proposed in (Gutierrez et al., 2005).

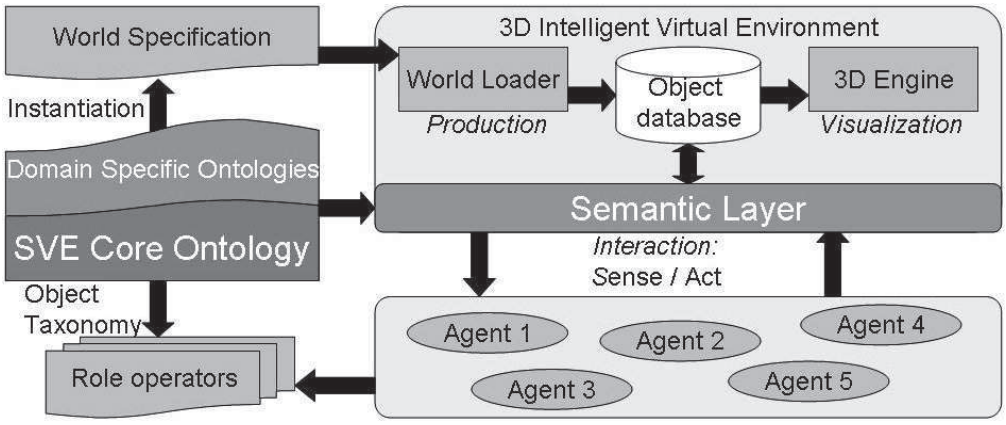


Figure 1.6: Semantic virtual environments based on Core Ontology and Domain Specific Ontologies, proposed in (Grimaldo et al., 2006a).

Grimaldo et al. (2006a) proposed a different ontology-based approach for semantic VEs. The main refinement relies on the use of two types of ontology: *Core Ontology* and *Domain Specific Ontologies* (*cf.* Figure 1.6).

The *Core Ontology* defines abstract concepts, as well as their properties and relationships between them. Abstract concepts are generic and independent to application domains. There is a taxonomy of abstract concepts based on specific terms in the field of VEs. For instance, abstract concepts are divided

into agents or objects. Objects are thereafter categorised into movable/non-movable objects, or graphical/non-graphical objects. Each abstract concept is associated with a list of properties. The properties of an abstract concept can be quantitative (e.g., location, height) or qualitative (e.g., empty, full). Relations between abstract concepts are organised in a hierarchical order. That is, an abstract concept may be the container of other ones. With regard to movable objects, there are three possible additional relations: "in", "on" and "pickedBy".

A *Domain Specific Ontology* defines concrete concepts of a particular domain. The domain-specific concepts are extended from abstract concepts predefined in the *Core Ontology*. A domain specific concept inherits properties and relations from its abstract concept. Thus, a *Domain Specific Ontology* represents knowledge about a domain.

1.1.4 System-oriented Semantic Modelling

The precedent section has discussed the main techniques used for content-oriented semantic modelling. Such semantic models provide an ontological description of the entities of the scene: identifiers, types, structures, attributes of entities, and relationships between them.

System-oriented semantic modelling adopts a different point of view. It is more oriented towards the modelling of the dynamics of the simulation. Semantic models convey information needed by the different modules of the VR application (namely different *engines*, e.g. for the rendering, the physics, or the interaction) to support the execution of the simulation. System-oriented modelling deals with concepts related to the implementation and takes advantage from mechanisms provided by object-oriented programming, like introspection.

Information-rich Virtual Environments

In (Bowman et al., 2003), semantics is considered as abstract information, i.e., information that is not directly perceptible by the user. For example, the visual appearance or surface texture of a table is directly perceptible, while meaningful information such as date and place of manufacture of the table is not. Therefore, semantic VEs should be augmented by abstract information. Such augmented VEs are called Information-rich Virtual Environments (IRVEs).

IVREs are built upon a central concept, called *semantic objects*, which encapsulate abstract information, such as text, numbers, images, audio, video, or hyperlinked resources (*cf.* Figure 1.7). At the implementation level, semantic objects are defined as reusable scenegraph nodes. More precisely, abstract information is described using VRML and X3D PROTOTYPE nodes. Thereafter, runtime accesses to semantic objects are obtained by means of APIs (Application Program Interfaces). In the case of VRML, APIs are EAIs (External Authoring Interfaces), and in the case of X3D, they are SAIs (Scene Access Interfaces).

Despite the concept of *semantic objects* accessible by APIs is helpful, the in-

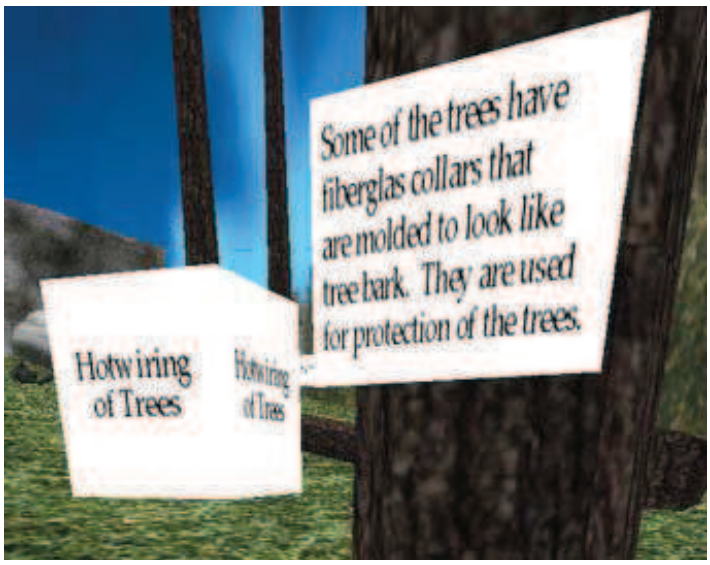


Figure 1.7: Illustration of an information-rich virtual environment, proposed in (Bowman et al., 2003).

tegration of domain knowledge into semantic objects using special scene graph nodes remains a difficult task. Moreover, research in IVREs is mainly focused at techniques for design and display of abstract information (Polys and Bowman, 2004). The questions are related to display location (i.e., where to display abstract information in a VE) or level of detail of abstract information, not to how to facilitate the conceptualisation of IVREs.

Multi-layered Semantic Reflection

A different line of research of system-oriented semantic modelling is based on the concept of *semantic reflection*. Inspired from object-oriented programming paradigms, reflection implies the abilities to access to the entities of the application during runtime. Thus, reflection provides information about the services provided by the entity and its current state (i.e., attributes and relations to other entities). Entities with semantic reflection abilities are called *semantic entities*.

Tutenel et al. (2009) introduced the *Semantic Class Library* to design semantic VEs, notably 3D games. After creating a 3D model, the designer associates the elements of the 3D model to existing classes in the library. Otherwise, the designer can create a new class with the desired properties. Beyond the 3D representations of objects within the game world, the *Semantic Class Library* provides additional semantics to the objects, such as physical attributes (e.g., the mass or material), functional information (e.g., how one can interact with an object).

SCIVE (Simulation Core for Intelligent Virtual Environments) is a framework using semantic reflection to design semantic VEs (Latoschik and Fröhlich, 2007). The architecture of SCIVE is based on *semantic net* – a network of semantic

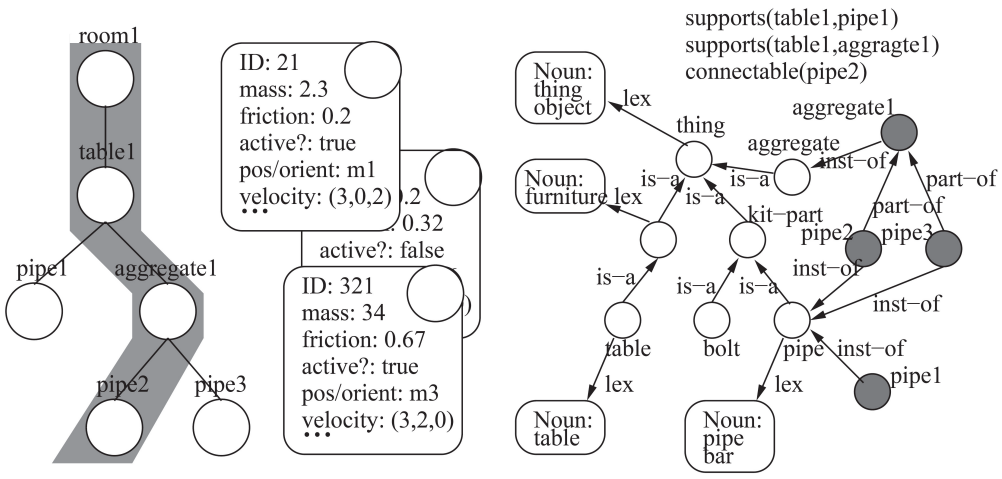


Figure 1.8: Based on the concept of semantic entity and semantic reflection, SCIVE links different representations (i.e., from left to right: graphical, physical, and knowledge representation) of a VE and provides uniform access to the application's entities (Latoschik and Schilling, 2003).

entities providing a common representation to VEs. As a VE is an integration of different representations (i.e., graphical database, physical database, and knowledge base), SCIVE provides a common representation layer of them (*cf.* Figure 1.8). This common representation layer is based on FESN (Functionally Extendable Semantic Network) (Latoschik and Schilling, 2003)(Heumer et al., 2005). In FESN, information about simulative entities, attributes, and relations is explicitly described using a XML-based representation. At the implementation level, semantic entities are programmed in C++. They allow the introspection of their attributes and services. As a consequence, information about entities stored in the semantic net can be synchronised at the execution time.

A VR application that allows users to perform assembly tasks of complex objects in 3D is a good example illustrating the concept of semantic reflection. Each virtual object has three separated representation: graphical representation, physical representation (e.g., mass, position, orientation), and knowledge representation (e.g., the object can be assembled to what objects?). SCIVE as well as the *Semantic Class Library* enable a unified knowledge layer of these three representations.

1.1.5 Semantic Modelling of Virtual Environments With a Conceptual Modelling Phase

Conceptualising Static Scenes and Behaviours

VR-WISE (Virtual Reality With Intuitive Specifications Enabled) is an ontology-based approach for the design of semantic virtual reality applications (Kleinermann et al., 2005). The main idea is to introduce a conceptual phase into the

design process of VEs. Thus, the development process in VR-WISE consists of three sequential steps, namely the specification step, the mapping step, and the generation step (*cf.* Figure 1.9).

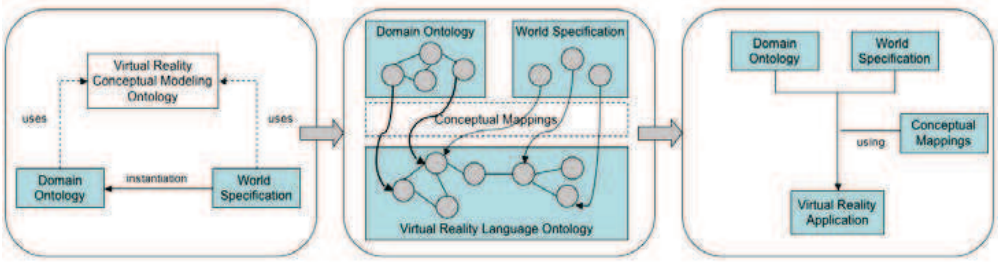


Figure 1.9: Overview of the design process according to the VR-WISE approach (from left to right): specification, mapping, and generation step (Kleinermann et al., 2005).

The specification step allows to specify the VE at a conceptual level, using domain knowledge, and without taking into account any implementation details. The specification is done at two levels: the concept level and the instance level. At the concept level, domain concepts are mapped into the concepts of the ontology, so called domain ontology. It is possible to describe the properties as well as the relationships between concepts. Behaviours of concepts such as "move", "turn" or "roll" can also be specified. The invocation of objects' behaviours is based on events (time events, user events, or object events) (Pellens et al., 2005). At the instance level, the instances represent the objects that will populate the VE, as specified by the domain ontology.

In the second step, the mapping from the conceptual level to the geometric primitives is specified. The purpose of this mapping is to specify how concepts of the domain should be represented in the virtual world. Similarly to the specification step, the mapping is defined at two levels. The domain mapping links the concepts of the ontology to VR implementation primitives. For example, a bin is represented by a cylinder. Although several instances may be instantiated from the same concept, in some cases, they may require different representations. Therefore, the world mapping allows defining mappings for instances to VR implementation primitives. In this way, the default mapping, specified in the domain mapping, can be overwritten.

Finally, the generation step will produce the source code for the VE defined in the specification step, using the mappings defined in the second step. There are two generated source files. The first file contains the 3D scene structure with its objects in VRML format. The second file contains the semantic information generated in the MPEG-7 format.

Conceptualising Interactions

NiMMiT (Notation for MultiModal interaction Techniques) is an alternative approach for designing semantic VEs with focus on semantics interactions (Vanacken

et al., 2008). The approach considers two main steps: conceptualising the static scene, and conceptualising the interactions.

At the conceptual phase, the approach in NiMMiT is partially similar to the VR-WISE and Grimaldo *et al.*'s approach (cf. Section 1.1.3). NiMMiT uses an ontology to map domain concepts to those of the ontology. The ontology allows to specify properties of objects related to interactions, such as selectable/non-selectable objects, or movable/non-movable objects.

Thereafter, the conceptualisation of the interactions in NiMMiT is done using a diagram-based notation intended to describe multimodal interactions between the user and the computer (Vanacken et al., 2007). An interaction primitive is described as a *task*, such as "select an object" or "move an object". A set of interactions is represented by a *task chain*. The interaction starts in the start-state, and ends with the end-state. An *event* is based upon the user's input that can be multimodal, such as speech utterances, gestures, pointing device events, or button clicks. A *label* is used to store values exchanged during the interaction. Finally, when the task chain has been completely executed, a *state transition* moves the state-diagram into the next state.

The main advantage is that the semantic model produced in the conceptual modelling phase can be used to enrich the interaction modelling phase. For instance, during a selection task, some objects should be selectable, but others not. Likewise, the moving task is only applied to several objects (e.g., tables and chairs of a room), but not all of them (e.g., walls and windows of the room). Thanks to the semantic description, the interactions are restricted to possible objects. Also, the conceptual modelling phase allows to specify interactions in a system-independent manner.

1.1.6 Discussion

We have analysed relevant approaches for the semantic modelling of VEs. The main existing techniques, frameworks, and methodologies in the literature for semantic modelling of VEs have been reviewed. In this section, we discuss the limitations of these approaches. As our ultimate focus is related to spatial semantics of VEs, we reformulate the main principles of the existing solutions from two different modelling perspectives: spatial entities and spatial relationships.

Limitations of the Related Approaches

For readability, Table 1.1 has previously summarised related approaches according to four criteria. These criteria can be revised as follows: 1) genericity (i.e., the semantic model should be domain-independent); 2) content-oriented semantic modelling (i.e., the semantic model supports access to attributes, relationships, and states of the scene's entities); 3) system-oriented semantic modelling (i.e., the semantic model provides access to the application's entities at runtime); and 4) the semantic model results from an explicit conceptual modelling phase. No previous approach presented so far satisfied all these criteria.

With regard to the genericity of the semantic models, both Informed VEs and semantic digital heritage models are limited to specific application areas (e.g., urban environments and cultural heritage).

Content-oriented semantic models (e.g., OntologyX3D, the SeVen platform, or XML-based ontology) are mainly obtained by coupling the scene content with ontologies. However, the coupling is carried out in an ad-hoc manner and in a manual way, because domain's and ontology's concepts are directly mapped onto low-level geometrical primitives.

Semantic models resulting from a conceptual modelling phase (e.g., VR-WISE and NiMMiT approach) overcome the manual mapping by using an explicit conceptual model. The conceptual model of a VE is commonly built upon graphical editors or diagram-based notations. Thus, it allows to introduce semantic information during the conceptual modelling. Nevertheless, a strong limitation of this approach is that the conceptual model only reflects a static view of the virtual world. Changes in objects' attributes and relationships are not updated at runtime upto the conceptual model.

System-oriented semantic models are mainly based on APIs or the concept of reflection. These paradigms enable meta-accesses to the application's entities during runtime. Thus, every update related to the scene content is observed.

From our point of view, ideally, the semantic modelling of VEs should combine both content-oriented and system-oriented modelling. Moreover, it should be based on generic metamodels representing abstract concepts of VEs, like the *Core Ontology* in Grimaldo *et al.*'s approach. It should provide a platform independent point of view on the VR application, built-up during the conceptual modelling phase of the development process. Based on these properties, spatial knowledge can be introduced into the semantic model of the VE and thus will extend the expressiveness of existing models. That way, spatial knowledge enriches the description of the content of the VE and, thanks to its formal representation, can also be used to specify constraints about the behaviour of the simulation.

On the Modelling of Spatial Entities

Most of the approaches presented previously have been built upon a partial conceptualisation of spatial entities within VEs. These approaches are summarised in Table 1.2. The common point is that the conceptual representation should be separated from the graphical representation of spatial entities. The reason is that, in VEs, some spatial entities may be associated with graphical resources, but this is not always true. The conceptual representation of spatial entities should contain not only information about their visual appearance, but also domain specific attributes and behaviours. It has also been observed that the common and basic semantic level constitutes a sound basis about spatial entities, such as location, orientation, width, height, and depth. Moreover, it is necessary to be able to update the evolution of spatial entities over the time, and to give access to their predefined services during runtime. Semantic reflection is

a relevant mechanism to achieve this goal.

Approach	Description
Smart objects (Kallmann and Thalmann, 1999)	Smart objects are virtual objects associated with a set of pre-programmed possible interactions with the user.
Semantic objects (Polys and Bowman, 2004)	Semantic objects are reusable scenegraph nodes encapsulating abstract information in the form of text, images, audio, video, or hyperlinked resources.
Semantic entities (Latoschik and Fröhlich, 2007)	Semantic entities are C++ objects providing semantic reflection about their capabilities (attributes and services).
Digital items (Gutierrez et al., 2005)	Digital items are virtual objects associated with semantic descriptors. A semantic descriptor decides the graphical representation of a digital item and defines the way to control it.
Ontology's concepts (Kalogerakis et al., 2006) (Grimaldo et al., 2006b) (Kleinermann et al., 2005)	Virtual objects are mapped into concepts of an ontology. Concepts may hold attributes and relations with other concepts. Ontology's concepts have common basic semantic defined in (Ibáñez and Delgado-Mata, 2006), such as object type, object identifier, location, orientation, width, height, or depth.

Table 1.2: Summary of related work regarding the modelling of spatial entities.

On the Modelling of Spatial Relationships

With a specific attention to semantic spatial constraints, we have summarised in Table 1.3 the related approaches that partially deal with the modelling of spatial relationships between objects within VEs. Evidence is that no previous work has dealt with the explicit modelling of semantic spatial relationships for VEs. Instead, these approaches merely specify relationships between spatial entities extracted from scene graphs. That might be "whole-part" relations, or special ones such as "in", "on", and "pickedBy". Whereas, semantic spatial relationships relating to direction, distance, or topology of the 3D space were not handled.

1.2 The MASCARET Approach

The previous section reviewed related approaches for the semantic modelling of VEs. This section focuses on a methodology and framework for semantic modelling of VEs named MASCARET, in which the work presented in this thesis takes place. In Part II, we will propose a model of spatial semantics of VEs that is generic and independent to the framework used for developing VEs. Then, we will show how our model of spatial semantics has been deployed into the MASCARET framework.

Approach	Description
Ontology-based approaches	By mapping the scene content to an ontology like in (Kalogerakis et al., 2006)(Otto, 2005b), it is possible to define relationships between concepts, such as "isPartOf" or "isGeneralisationOf".
Grimaldo et al. (Grimaldo et al., 2006b)	There are three further possible relations that can be defined between movable objects: "in", "on", and "pickedBy".
VR-WISE (Kleinermann et al., 2005)	During the conceptual modelling of the static scene of a VE, it is possible to specify relative positions (e.g., left, right, front, back, above, and under) to place an object with respect to another object in an ontology. However, these relations are not updated when some changes occur in the environment during the execution.

Table 1.3: Summary of related work regarding the modelling of spatial relationships.

The remainder of this section is structured as follows. Section 1.2.1 gives an overview of the approach. Section 1.2.2 describes how to develop an application using MASCARET. Section 1.2.3 presents the architecture of the MASCARET framework. Finally, Section 1.2.4 summarises the approach and discusses the limitations, regarding the modelling of spatial knowledge.

1.2.1 Presentation

MASCARET stands for *MultiAgent System for Collaborative, Adaptive & Realistic Environments for Training*. Initially, it was dedicated to the development of VR environments for human learning (Querrec, 2002; Chevaillier, 2006; Querrec, 2010). However, it has also been successfully applied for developing other types of VR applications, such as agent-based simulations (Septseault, 2007; Marion et al., 2007) or cultural heritage applications (Barange et al., 2011). The main goals of the MASCARET project are twofold:

1. to develop semantic-rich VEs in which knowledge about the environment is explicit to both users and agents,
2. to provide a methodology for designing and implementing such semantic-rich collaborative VEs.

To reach these goals, MASCARET is grounded on two original propositions:

- P1: to use the Unified Modeling Language (UML) as a generic language for specifying the different aspects of collaborative VEs (e.g., structure, behaviours, interactions, activities of users and agents),
- P2: to use a model-based process for developing and implementing VEs.

Proposition P1 is related to the use of UML as a common language for the semantic modelling of VEs. As discussed in Section 1.1, a variety of languages have been used for the semantic modelling of VEs. They can be divided into two categories. The first category is related to the definition of a new Domain Specific Language (DSL), dedicated to a specific family of VEs. This is the case of domain-dependent semantic modelling approaches. DSLs are mainly specialisations of markup languages (e.g., XML or DTD) that allow to specify concepts of a specific domain, such as urban environments or cultural heritage environments. The second category is concerned with the use of ontology languages (e.g., OWL or RDF). Ontology languages allow to model domain concepts that compose the static view of the VE. However, they have strong limitations in the modelling of the dynamics of the VE (e.g., interactions or behaviours). Often, textual scripting languages are integrated with ontology languages to enable the description of the dynamics of the VE. As stated in (Chevaillier et al., 2011), UML allows to overcome these difficulties. First, UML offers the ability to model both the static (or structural) view and the dynamical (or behavioural) view of the VE. Second, UML is an extensible modelling language. Extensions of UML are define using *profile*. Profiles allow to extend and/or specialise the semantics of UML, or existing profiles, for a specific domain of application. MASCARET is based on two UML profiles, specifically adapted to the modelling of collaborative VEs, namely VEHA and HAVE (Chevaillier et al., 2009).

Proposition P2 is concerned with the use of a model-driven approach for developing VEs. Thanks to its grounding on UML, the methodological approach in MASCARET follows the main principles of the MDA/MDE (Model-Driven Architecture/Model-Driven Engineering) framework, as envisioned by the Object Management Group (OMG, 2011). From this perspective, one of the interesting properties, that MASCARET shares with UML, is that it combines both graphical notations and textual formal expressions thanks to the Object Constraints Language (OCL), part of UML (OMG, 2006). The process to specify and develop the VE using MASCARET is detailed in the next section.

1.2.2 Development Process

Using MASCARET, the development process involves three roles: domain experts, graphical designers, and software engineers. The development process consists in three main phases: the *domain modelling* phase, the *instantiation* phase, and the *implementation and execution* phase (Querrec, 2010).

Domain Modelling Phase

First, the domain expert defines the conceptual model of the domain represented in the VE in the form of UML–MASCARET diagrams. This step can be completed using any UML modeller which supports modelling extensions as defined by UML profiles (it is the case for most of the commercial modellers). As we will see later on, the domain expert has to describe the structure of the VE using VEHA models. The behavioural models of the VE are based on state ma-

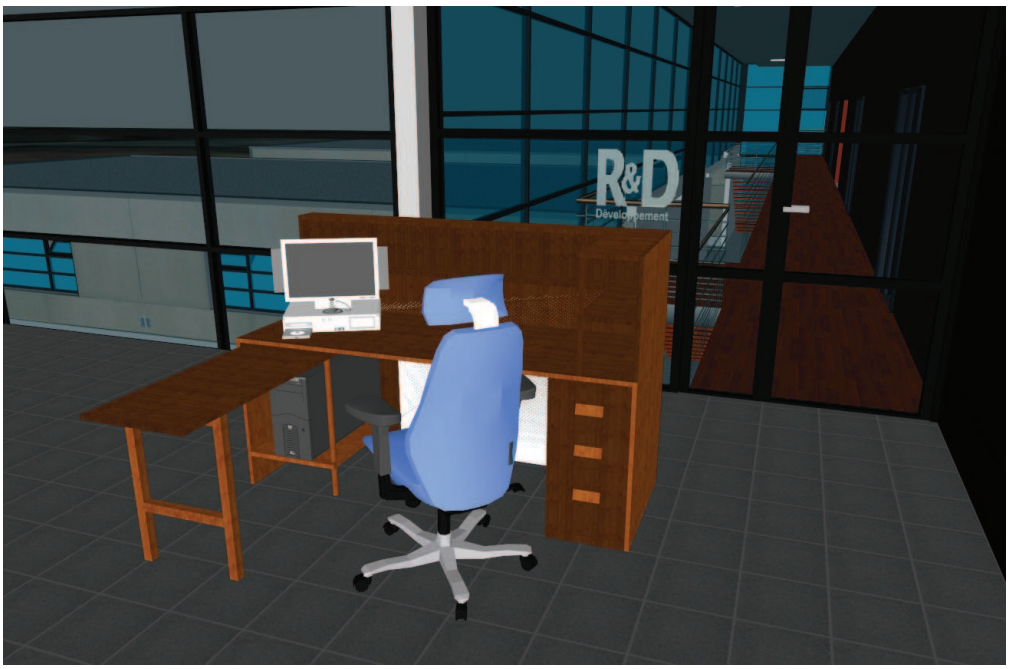


Figure 1.10: Example of a VE modelled using MASCARET: the graphical modelling view of the environment.

chines and activity charts. Thanks to HAVE, human activities can be specified using UML collaboration and activities diagrams. This domain model is then exported as an XMI[6] file. Here, the conceptual model of the VE is abstract. It is independent to any VR platform, 3D modelling formats, and programming languages.

However, it is important to notice the essential differences between MASCARET and other semantic modelling approaches. First, unlike approaches such as VR-WISE or ontology-based approaches, MASCARET does not aim to automatically generate the VE (i.e., in the form of VRML or X3D code). Instead, the domain model is used to conceptualise the environment and represent knowledge about the collaborative VE. For instance, Figure 1.13 and 1.14 illustrate the conceptual model partly corresponding to the VE graphically showed in Figure 1.10.

Instantiation Phase

Second, 3D designers have to design geometrical objects using a classical 3D modeller, see Figure 1.11. MASCARET provides plugins for several 3D modellers (e.g., 3DS Max and Blender) to refer to the conceptual model (the XMI file). These plugins allow the designer to add semantics to geometrical objects which are defined as instances of the conceptual model. Therefore, different VEs can be built, based on the same conceptual model. Those environments are then

[6]XML Metadata Interchange, http://www.omg.org/spec/XMI/

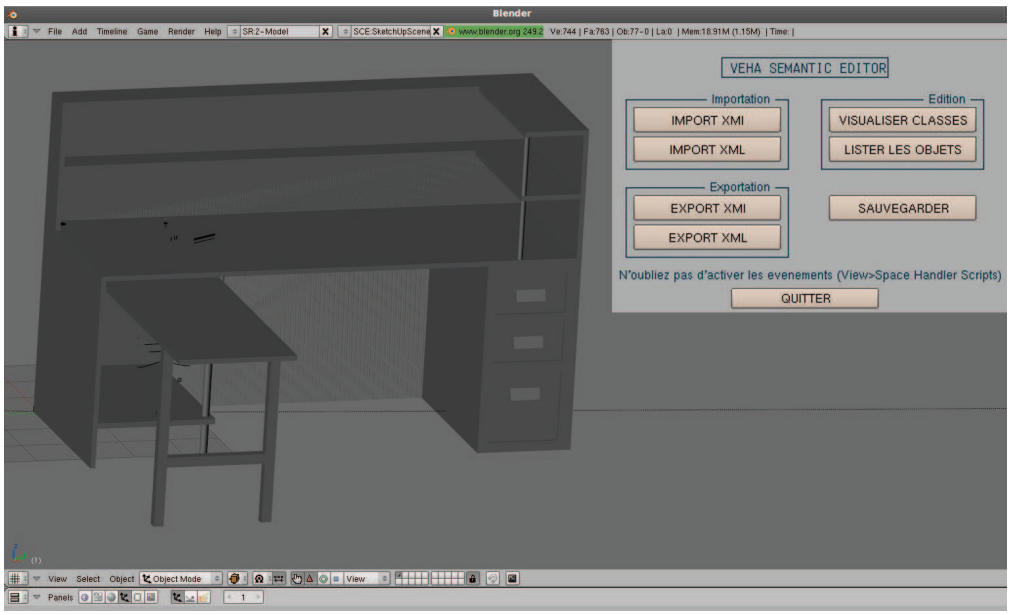


Figure 1.11: The graphical designer constructs the 3D scene using a 3D modeller.

exported into a file referencing (1) the conceptual model (Xmi file) (2) the links with the files containing the graphical models, and (3) the instances of the conceptual model.

Implementation & Execution Phase

Mascaret comes with packages for its implementation on various VR plaforms, namely ARéVi (Harrouet et al., 2006) and Ogre[7]. One of the novelties in Mascaret is that both the conceptual model and the model of the environment are loaded in the application. Mascaret defines a computational semantics for these two models. The models are interpreted by the Mascaret virtual machine and their execution on the VR platform is tractable. Therefore, it is possible to access the model at any time. Even if users' interactions or the internal dynamics of the environment lead to some changes in the environment (e.g., instantiation of new entities, or new role assignment for an agent), the consistency between the conceptual model and the graphical representation is *de facto* ensured. Besides, agents can make decision based on the execution of the model.

To conclude, Mascaret allows domain experts, graphical designers, and software engineers to work together sharing their knowledge on a same model. The architecture of Mascaret is detailed in the next section.

[7] http://www.ogre3d.org/

1.2.3 Model-based Architecture

The architecture of MASCARET conforms to the modelling architecture defined within the MetaObject Facility (MOF) framework (MOF, 2011). Therefore, MASCARET complies with the basic principles of the MDA/MDE approach. Figure 1.12 shows the multi-layer architecture of MASCARET, with respect to the MOF framework. For the sake of clarity, we briefly present in the following respectively the M1, M0, and M2 layer of MASCARET.

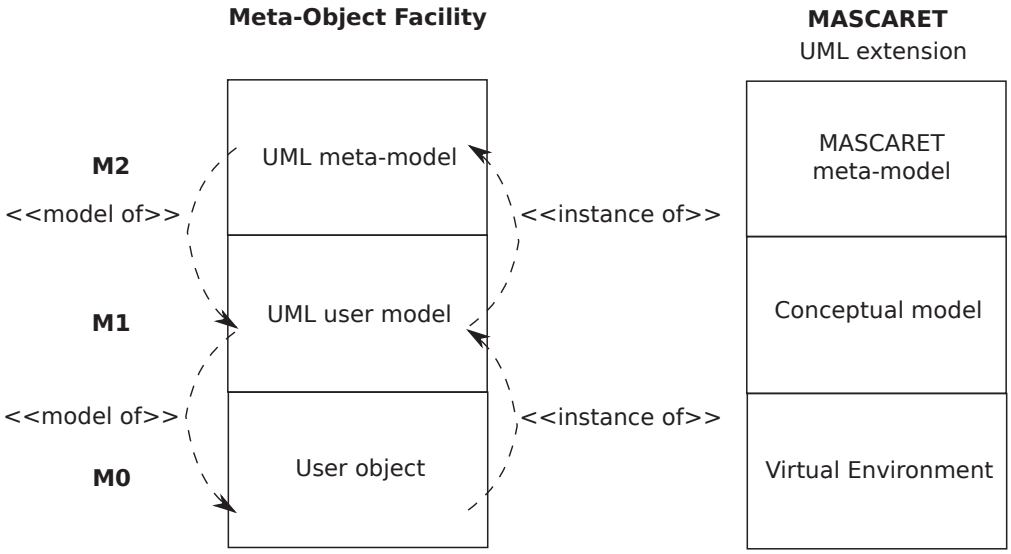


Figure 1.12: The multi-layer architecture of MASCARET (w.r.t MOF framework) for the semantic modelling of VEs.

The M1 Layer: The Conceptual Level

The M1 layer corresponds to the conceptual model of the VE. In MASCARET, the conceptual model is specified using VEHA and HAVE, two profiles defined to extend and specialise UML for the modelling of VEs (see Section 1.2.3). MASCARET does not come with additional types of diagrams. Thus, standard UML diagrams are used to draw the different modelling views, namely the environment as experienced by virtual and real humans, and the collaborative activities both virtual and real humans are supposed to perform within the environment. Main diagrams are:

- domain concepts and the structure of environment: class diagrams
- behaviours of entities: statecharts
- activities of users and agents: activity charts
- organisational structures: collaboration diagrams.

Regarding the scope of this thesis, we were merely interested in the modelling of structure and behaviours of the environment. For more information about how interactions, organisation, and collaboration are modelled in MASCARET, we refer the readers to (Querrec et al., 2004; Buche et al., 2004; Querrec et al., 2011).

Concepts of the Domain and Structure of the Virtual Environment

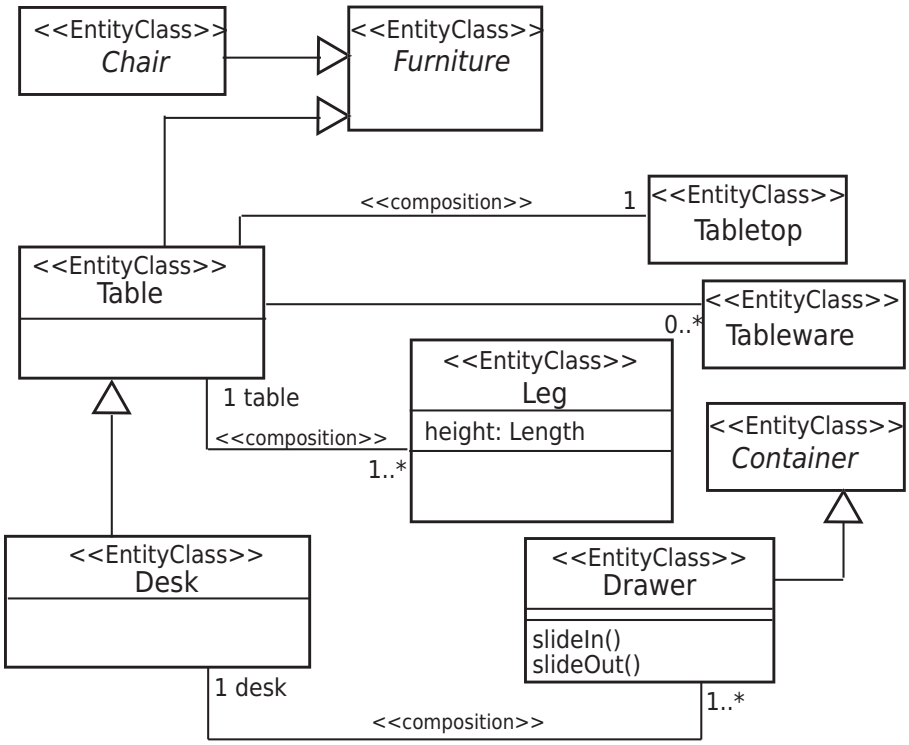


Figure 1.13: Conceptual modelling of a desk using MASCARET, from (Chevaillier et al., 2011). The vocabulary used is based on the WordNet lexical database.

The structure of the VE is represented using the VEHA meta-model in the form of class diagrams. In Figure 1.10, the environment is composed of different pieces of furniture (Desk and Chair), and other objects. Using MASCARET, these concepts, and how they are structured, are partly represented by the class diagram illustrated in Figure 1.13. Conceptual relationships between domain concepts are represented by stereotyped UML associations. Besides, Chair and Table are kind of Furniture, which is an abstract concept with no graphical representation. A table is composed of one TableTop, one or more Leg. It may have one or more Tableware. A Desk is a kind of Table. In addition to the predefined structure of the table, the desk has one or more Drawer. According to this model, the only two possible operations that one can performed on the drawer are slideIn() and slideOut().

Behaviours of the Entities

Some of the entities compounding the environment may have behaviours. MASCARET supports the modelling of *reactive* behaviours. That is, the behaviour of an entity results from the interactions with other entities, user's interactions on the entity, or the internal activity of that entity. MASCARET reuses the behavioural model of UML. Behaviours are modelled by means of statecharts which are based on the concepts of *state*, *transition*, and *event*. Contrary to UML, in MASCARET, state machines are explicitly associated to classes. When an event occurs, it may trigger the reaction of some entities. The corresponding signal is broadcasted to all the state machines of the entities sensitive to this type of event. The reception of the event may result in the firing of some transitions, with respect to the currently active states of the entities. When a transition is fired, the entity enters into another states. Otherwise, a reaction may be the execution of an operation, or a so-called *opaque behaviour*, i.e., a specific piece of code for which no semantic representation is given.

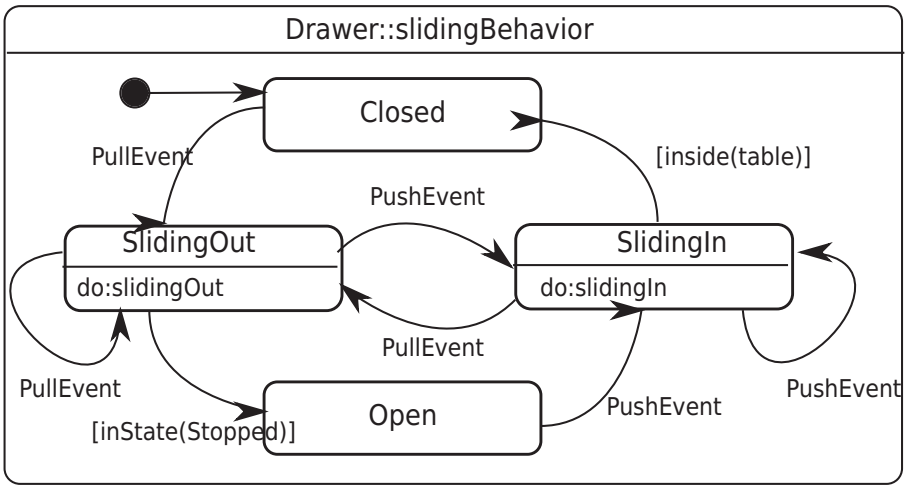


Figure 1.14: Conceptual modelling (partial) of the sliding behaviour of the drawer using MASCARET, from (Chevaillier et al., 2011).

Figure 1.14 illustrates the behavior of the drawer. The state machine specifies the sliding behaviour of the drawer (refer to the Drawer class from Figure 1.13). The drawer has here four possible states: Closed, Opened, SlidingOn and SlidingOut. The drawer is sensitive to interactions that trigger Pull or Push signals. The internal dynamics of the drawer can lead to state changes, depending on its position (inside(table)) or when entering into a state (inState(Stopped)). For the sake of simplicity, the state machine that governs the activation of these states is not represented here.

The M0 Layer: The Instance Level

Once the conceptual model of the environment is defined, the M0 layer describes one of the possible instantiations of the conceptual model. Each instantiation corresponds to a specific VE. Several VEs can be instantiated from the same

conceptual model. Every instance model represents a concrete description of the entities and the structure of the VE. Figure 1.15 illustrates the instantiation of a chair – an instance of the Chair class. Furthermore, it is possible to define the initial conditions for the state machine representing the behaviour of an entity. Also, it is possible to instantiate a specific organisation for users and agents, as well as their ongoing activities.

```
(1)<Entity class="Chair" name="cyanChair">
(2)  <Shape
          url="VRMLS/Chairs/cyanChair.wrl"/>
(3)  <Position
          x="-1" y="2" z="0"/>
(4)  <Rotation
          roll="0" pitch="0" yaw="-0.785398163"/>
(5)</Entity>
```

Figure 1.15: Example of an instantiation of an entity. Left: its graphical representation. Right: the value of some of the properties of that object, as defined into the conceptual model and the 3D model.

The M2 Layer: The Meta Level

Finally, the M2 layer corresponds to the meta-model of Mascaret. The meta-model itself is a model that defines the properties of the modelling elements that can be used to define the M1 model. For instance, concepts such as type, composition, state, behaviour, or operations are defined as classes (a.k.a. meta-classes) which holds their own properties (a.k.a. meta-attributes). In other words, the meta-model allows the reification and the introspection of the conceptual model. With respect to the Mof framework, this meta-model is defined using a subset of the Uml meta-model.

Mascaret is based on two complementary meta-models, namely Veha and Have (Chevaillier et al., 2009). Veha (a model for Virtual Environments supporting Human Activities) provides semantics about the structure and the dynamics of the VE, as it can be experienced by both natural and artificial agents during their activities. Whereas Have (a model of the Human Activities within Virtual Environments) provides the conceptual view of the collaborative activities the agents are supposed to perform within the virtual environment. In the following paragraphs, we briefly describe the elements of these meta-models the most closely related to the implementation of the conceptual spatial models we propose in the thesis. For a more detailed description about these meta-models, we refer the readers to (Chevaillier et al., 2009; Querrec et al., 2011).

Domain-Specific Concepts and Structure of the Virtual Environment

Figure 1.16 is a conceptual view of part of the MASCARET's meta-model representing domain concepts. The EntityClass is a meta-class, derived from the Class class of UML's meta-model. It represents a family of domain concepts. In turn, a particular domain concept is called an Entity. An entity is an instance of the EntityClass meta-class, derived from the InstanceSpecification class of UML's meta-model.

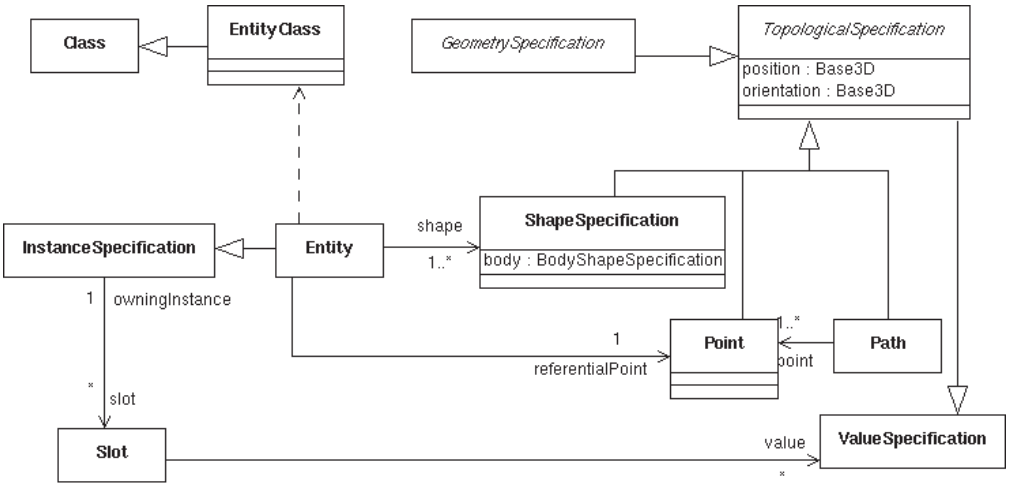


Figure 1.16: Conceptual view of the metamodel dedicated to the modelling of the concepts of the domain in MASCARET, from (Chevaillier et al., 2009).

Dedicated to the domain of VEs, MASCARET's meta-models have several important extensions compared to UML's meta-model. A domain concept (i.e., an Entity in MASCARET) has additional topological and geometrical properties. An entity can have one or more ShapeSpecification, they are graphical representations of an entity in a VE. An entity is located in a VE by a referential point (represented by the Point class).

Behaviour of the Entities

Figure 1.17 presents a part of the MASCARET's meta-model representing the behaviours of entities in VEs. Entities may have Operation. They can be associated with StateMachine. Each type of behavioural models are defined as a specialisation of the Behavior class.

1.2.4 Current Limitations for Specifying Spatial Constraints

In the previous sections, we have seen how MASCARET specialises the UML metamodel. Different UML–MASCARET diagrams have been used to model the VE. The class models represent the structure of the VE. An entity in the VE can be associated with behavioural models (i.e., state machines). The activities of human and artificial agents are defined using UML-like collaboration and activity diagrams.

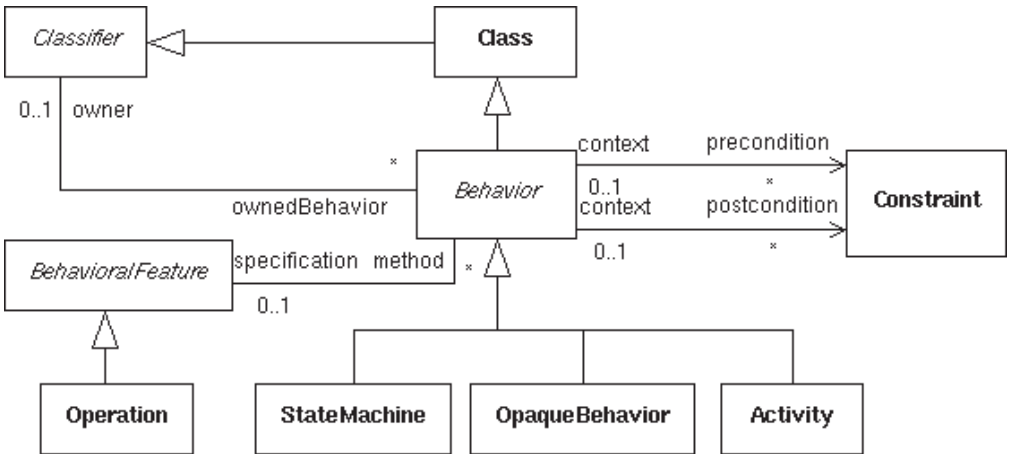


Figure 1.17: Conceptual view of the metamodel dedicated to the modelling of the behaviour of the entities in MASCARET, from (Chevaillier et al., 2009).

However, there exists many semantics that UML-based models are unable to convey by themselves. For example, considering the Desk model illustrated in Figure 1.13, a common understanding about the ontological model of the desk is that *"the height of a table is greater than the length of all of its legs"*. Similar semantic expressions can be found in the description of the structure of VEs, for instance the cardinality constraints (e.g., *"a table should have at least three legs"*). With regard to operational semantics, operations in VEs are often conditional, i.e., an operation can only be executed according to a precondition and is considered as completed when the post-condition will become true. For example, the operation SlidingOut of a drawer is terminated when there is no motion on the drawer: the post-condition of the operation is that the drawer is in the Stopped state. At last, spatial constraints among virtual objects, that are very important for the semantic description of the VE, can not be specified (e.g., *"the table is on the floor"* or *"initially, the drawer is inside the table"*).

To overcome the limitations of graphical modelling, it is necessary to integrate more formal and logical information into the UML-based conceptual models of MASCARET. In this thesis, we propose a straightforward solution which consists in extending the UML Object Constraint Language (OCL) into VRX-OCL, which stands for *Virtual Reality eXtension of OCL*.

1.3 Summary

In this chapter, we have presented the relevant approaches for the semantic modelling of VEs. Thereafter, we have described the MASCARET framework that forms the implementation context for our work on the modelling spatial semantics of VEs.

Regarding existing techniques for semantic modelling of VEs, we have been interested in two main questions: how to model spatial entities, and how to

model semantic relationships between them in VEs. An important conclusion is that, while most of the current approaches have raised to basic conceptual representation of spatial entities, none of them has dealt with a semantic model of spatial relationships between spatial entities within VEs.

MASCARET proposes a metamodel-based architecture for specifying and modelling VEs. It allows to build platform-independent semantic models, using an explicit phase of conceptual modelling. However, the main limitation of the current version of MASCARET is that it still lacks a semantic model for spatial relationships dedicated to VEs.

Chapter 2

Spatial Models

The previous chapter presented the related work in the domain of semantic modelling of VEs. The main conclusion is that there still lacks a model of spatial semantics of VEs. In this chapter, we focus on spatial models. We discuss the related approaches in the literature that are feasible for modelling spatial constraints between spatial entities in VEs.

This chapter is structured as follows. Section 2.1 places spatial constraints in different taxonomies, according to their inherent spatial characteristics, such as metric/nonmetric, quantitative/qualitative, or static/dynamic. Section 2.2 describes the relevant theoretical models to formalise spatial constraints. Section 2.3 discusses different solutions for conceptualising spatial constraints. The emphasis will be on high-level spatial languages and frameworks. Section 2.4 summarises the chapter.

2.1 Taxonomy of Semantic Spatial Constraints

To integrate semantic spatial constraints into VEs, it is necessary to identify the main families of spatial relationships existing in the real world and their inherent spatial characteristics. In this section, we carry out our review of the main taxonomies of constraints, in particular spatial constraints.

2.1.1 Metric vs. Non-metric Constraints

Spatial constraints are an important subject of study in research about spatial cognition. In (Waller et al., 2000), to study the place learning in humans, the authors categorised spatial relationships into two families: metric and non-metric spatial relationships. The main aim is to study the correlation and interrelationship between these two families of spatial constraints.

Metric relationships are based on distance and angular information. These relationships include the relative distances from the viewing locations and the object, as well as the angles formed by them.

Non-metric relationships are related to topological relationships (e.g., adja-

cency) and projective relationships (e.g., betweeness, leftside/rightside, or clockwise/counterclockwise).

One of the main conclusions obtained from this study is that spatial cognition (in this case, the memorisation of place in humans within a computer-generated environment) is affected not only by metric but also nonmetric relationships present during the learning process.

2.1.2 Quantitative vs. Qualitative Constraints

Spatial constraint can be quantitative (e.g., "45 degrees" or "5 meters") or qualitative (e.g., "front of" or "inside"). Nevertheless, research in the field of qualitative spatial representation and reasoning (QSR) aims to represent, and then reason about, spatial aspects of the world in a *qualitative* manner. An important motivation for such a QSR approach is that it is considered to be closer to how humans represent and reason about commonsense knowledge. Another motivation is that it is possible to deal with incomplete knowledge (Cohn and Renz, 2008). The most important aspects of space are qualitatively represented as the following:

- direction ("left", "above")
- distance ("far","near")
- topology ("touch", "inside")
- size ("large", "tiny")
- shape ("oval", "convex")

It is obvious that the expression of spatial aspects is purely symbolic and qualitative. The above spatial aspects are not independent. For example, the size of an object is dependent to the distance from which it is observed. Also, different points of view (i.e., directions) may yield different observations to the shape of an object.

2.1.3 Static vs. Transition Constraints

One of the first attempts that aims to put forward a taxonomy of spatial constraints was in (Cockcroft, 1997, 2004). The initial point of view was that, constraints should be divided into *static* constraints and *transition* (or *dynamic*) constraints. Static constraints must be satisfied at every single state of the environment (e.g., a salary cannot be negative). Meanwhile, transitional/dynamic constraints are applied when the environment changes from the current state to another state (e.g., on updating a salary, its amount should not decrease).

With regard to static spatial constraints, the author proposed a classification based on the distinction between topological, semantic, and user rules (cf. Figure 2.1). *Topological* constraints are concerned with geometrical properties

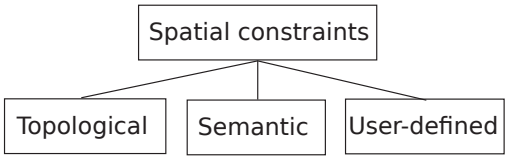


Figure 2.1: Cockcroft's taxonomy of static spatial constraints.

and spatial relations of spatial objects (e.g., all polygons must close). *Semantic* constraints are concerned with the meaning of geographical features (e.g., a road can not run through a lake). *User-defined* constraints are more esoteric in nature and not necessarily based on semantics (e.g., a nuclear power station must be located by a distance from residential areas).

Finally, a taxonomy based on two dimensions was presented. The first axis makes the distinction between static and transitional constraints. The second one classifies constraints in terms of topological/semantic/user-defined. This results in six combinations: topological static, topological transition, semantic static, semantic transition, user static, and user transition. Table 2.1 illustrates some examples for each type of spatial constraints.

1	Static semantic	The height of a mountain may not be negative.
2	Static topological	All polygons must close.
3	Static user	All streets wider than seven metres must be classified as highways.
4	Transition semantic	The height of a mountain may not decrease.
5	Transition topological	If a new line or lines are added making a new polygon the polygon and line tables must be updated to reflect this.
6	Transition user	Road of any type may not be extended into body of water of any type.

Table 2.1: Examples of a classification of spatial constraints, according to (Cockcroft, 1997).

2.1.4 A Fine-grained Classification of Constraints

In (Louwsma et al., 2006), the authors recognised the relevance of a two-dimensional taxonomy that allows the distinction between the static versus transitional aspect of constraints, as presented in Cockcroft's approach. However, as the spatial constraints classification proposed by Cockcroft was quite generic, the authors additionally refined the second axis of Cockcroft's taxonomy by proposing different criteria for the classification of constraints. These criteria were:

1. The number of objects/classes/instances involved in constraints. A constraint can be related to:

- a single instance (restrictions on attributes and relationships between attribute values of a single instance)
- two instances of the same class (binary relationship)
- multiple instances of the same class (aggregate)
- two instances of two different classes (binary relationship)
- multiple instances of different classes (aggregate)

2. The type of properties/relationships of the objects involved. They could be:

   - spatial features: direction, topology, distance.
   - temporal features (e.g., adjacent in time like "bushes may only be located after trees have been positioned")
   - thematic: semantics not related to spatial or temporal relationships (e.g., "a parcel must always be owned by at least one person").
   - complex: a combination of the above properties (e.g., quantity or aggregate constraints like "maximum of 10 planting objects in a specified area in the centre of the park").

3. The dimension of constraint:

   - temporal dimension - 1D
   - spatial dimension - 2D/3D
   - spatiotemporal dimension - 4D
   - thematic measurement scale

4. The manner of expression:

   - a constraint "must never" occur
   - a constraint must "always"occur

5. The nature of the constraint (physically impossible or domain dependent).

   - theorem-based or physically impossible (e.g., "a tree cannot float in the air")
   - design-based (e.g., "bush should be south of tree")

This novel taxonomy of constraints obviously includes the definition of previous constraint types as defined in Cockcroft's taxonomy (i.e., topological, semantic, and user-defined). Furthermore, this taxonomy deals with not only spatial constraints but also constraints in general (e.g., temporal constraints or cardinality constraints).

2.1.5 Discussion

We have reviewed a variety of taxonomies of constraints in general, with a special focus on spatial constraints. Evidence is that a qualitative approach is very well suited for representing and reasoning about spatial semantics in VEs. The qualitative representation of spatial semantics appears to be closer to how human perceives and communicates about spatial knowledge. Among the presented constraints taxonomies, the proposal of (Louwsma et al., 2006) is the most recent, relevant, and complete one. This taxonomy is generic and covers the previous classifications of constraints. Main aspects of space (e.g., topology, distance, and direction) are included. These main aspects can be further divided into metric and non-metric constraints.

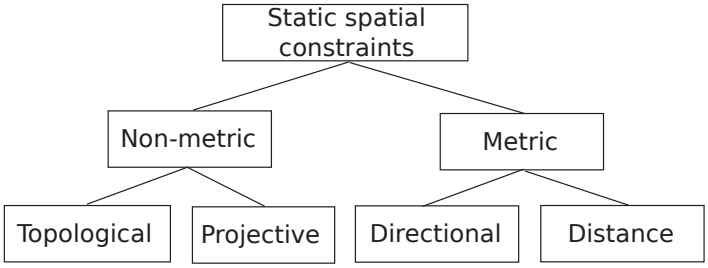


Figure 2.2: Summary of pure spatial constraints in 3D environments.

Based on the above critical remarks, Figure 2.2 outlines our perspective for a taxonomy of spatial constraints that should be represented in 3D environments. It is hierarchical and purely spatial. Other requirements related to representation issues of spatial constraints (e.g., the number of objects/classes/instances involved, or the manner of expression) should be treated during the design of spatial modelling languages.

In the next section, we will give a closer look at each type of spatial constraints. Existing models for topological, projective, directional, and distance relations will be presented respectively.

2.2 Spatial Models

2.2.1 Topology

Topology is a fundamental aspect of space. Topological relations are preserved under topological transformations like rotation, scaling, or translation. The two main approaches for modelling topological relations are the *Region Connection Calculus* (*RCC*) by Randell, Cui, and Cohn (Randell et al., 1992) and the *9-intersection* model proposed by (Egenhofer, 1991).

Based on different levels of connectedness between two regions, the *RCC* model distinguishes a set of eight base relations, including the following relations: `DC` (disconnected), `EC` (externally connected), `PO` (partially overlap), `TPP` (tangential proper part), $\texttt{TPP}^{-1}$ (tangential proper part inverse), `NTPP` (non-

tangential proper part), $\texttt{NTPP}^{-1}$ (non-tangential proper part inverse), and EQ (equal). These relations, noted as *RCC-8*, are considered as the smallest set allowing topological distinctions, as illustrated in Figure 2.3.

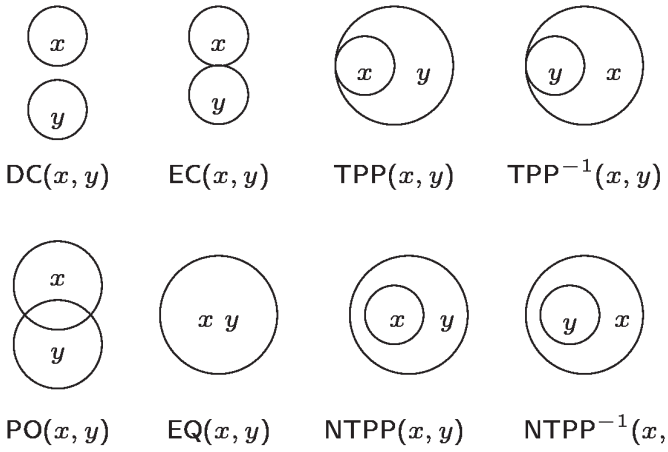


Figure 2.3: Eight basic topological relations in 2D, proposed in (Randell et al., 1992).

Alternatively, to model topological relations, (Egenhofer, 1991) proposes to use a 3×3 matrix corresponding to the nine possible intersections between the interior (^{o}), exterior ($^-$), and boundary (∂) of regions, so called *9-intersection* model. When dealing with 2D spatial regions, Egenhofer's model produces the same set of topological relations as in *RCC-8*. These relations are named as "disjoint", "meet", "overlap", "inside", "coveredBy", "contains", "covers", and "equal". Figure 2.4 illustrates the "meet" relation and the corresponding matrix in the *9-intersection* model.

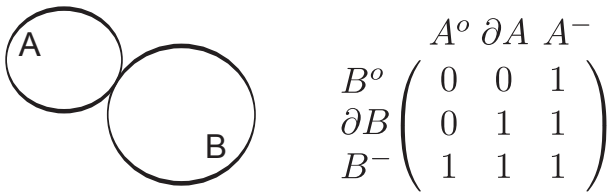
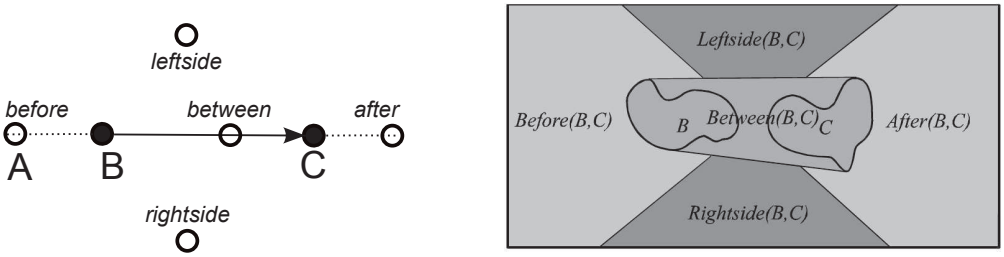


Figure 2.4: Example of the "meet" relation and the corresponding matrix in the *9-intersection* model, proposed in (Egenhofer, 1991).

More recently, in (Zlatanova, 2000; Zlatanova et al., 2004), the authors used the *9-intersection* model to investigate the number of topological relationships among four types of objects: points, lines, surfaces, and bodies in 3D. Overall, 69 possible relationships between them was identified. In particular, there exists 8 possible topological relations between two bodies in 3D.

2.2.2 Projection

Projective relations remain invariant under projective transformations. An example of projective invariants is the collinearity. That is, three collinear points A, B, C remain collinear after a projection. Based on the concept of collinearity, the *5-intersection* model allows to distinguish five possible relations of a point A with respect to the two given points B and C in 2D (Clementini and Billen, 2006). Figure 2.5a shows the five relations, i.e., "before", "after", "between", "leftside", and "rightside".



(a) (b)

Figure 2.5: 5 projective relations among three points and regions are defined in the *5-intersection* model (Clementini and Billen, 2006).

The point-based projective model thereafter was extended to regions and volumetric objects in 3D space (Billen and Clementini, 2006). The concept of collinearity between points is applied to regions. Figure 2.5b shows the five possible projective relations among three regions.

2.2.3 Direction

While topological and projective relations are pure non-metric and qualitative, directional and distance relations represent the metric aspects of space. However, research in direction modelling has converged on a qualitative representation of direction. That is, direction is described using qualitative prepositions such as "left of" instead of numerical terms such as "90 degrees". A rationale for a qualitative approach is that it is often used in daily communication about direction. It is also considered to be closer to the human mental representation of direction. Currently, directional models differ from each others by the dimension of space (i.e., 2D or 3D space). This leads to a further distinction between existing directional models based on how entities are represented in space. Most approaches merely consider 2D space, where entities are simplified as points or regions. Whereas, there exists very few models dealing with volumetric entities in 3D space. For each directional model below, we first describe the model in 2D and then discuss its extensions in 3D where appropriate.

The *double cross* model considered direction among three points in 2D, including a primary point and two reference points forming a vector (Freksa, 1992).

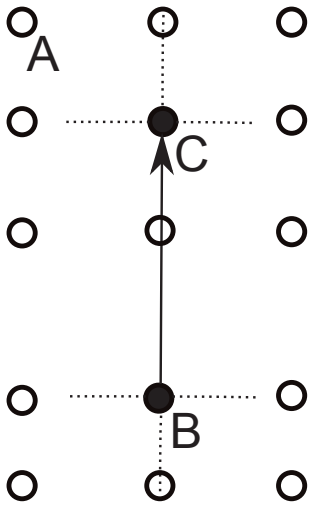


Figure 2.6: Direction among three points (a point A and a vector $\vec{BC}$) in 2D space: 15 relations are defined in the double cross model (Freksa, 1992)

Based on the two lines orthogonal with the vector, 15 relative positions of the primary point with regard to the vector could be distinguished (see Figure 2.6). Compared to projective relations, the *double cross* model can be considered as a more fine-grained representation of the *5-intersection* model presented in the previous section. The 15 basic directional relations in the *double cross* model cover the 5 basic projective relations among points. In (Pacheco et al., 2002), an extension of the *double cross* model was introduced that distinguishes 75 relations among three arbitrary points in 3D.

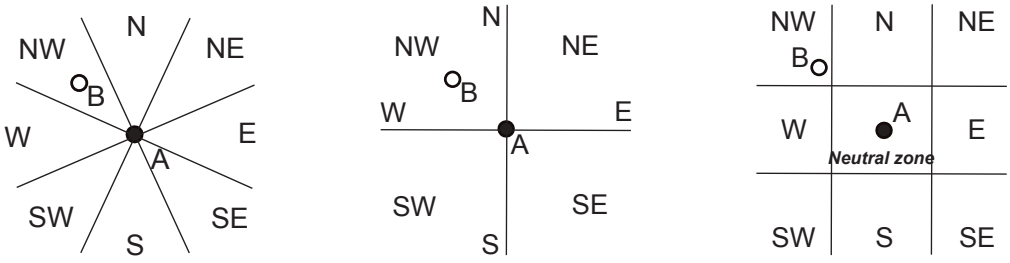


Figure 2.7: Cardinal directions between two points in 2D space (Frank, 1992): (a) cone-based model; (b) projection-based model; (c) projection-based model with neutral zone.

Alternatively, the *cardinal direction* model defined direction between two points: a primary point B and a reference point A in 2D (Frank, 1992). The model assumes the existence of a reference system, such as the intrinsic orientation of the reference object (e.g., "front", "back") or magnetic poles (e.g., "north", "south"). Such a reference system divides the space around the reference point

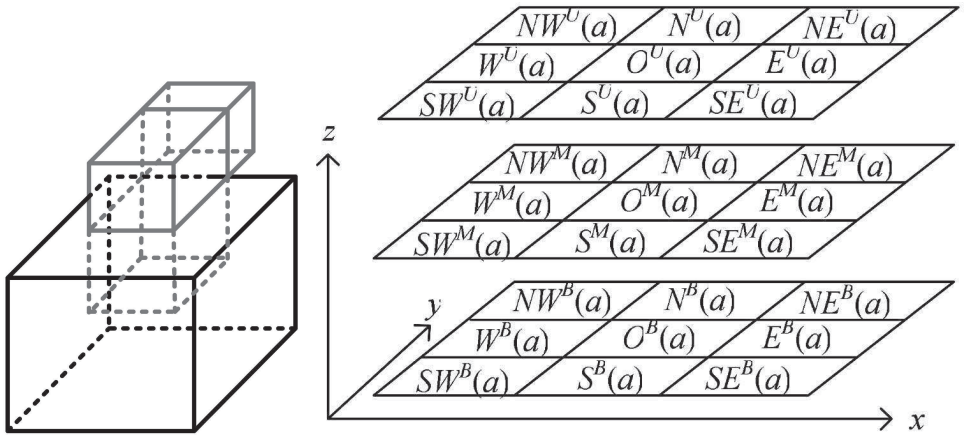


Figure 2.8: In the TCD model (Chen et al., 2007), 27 relations are defined around a reference object approximated by its axis-aligned bounding box.

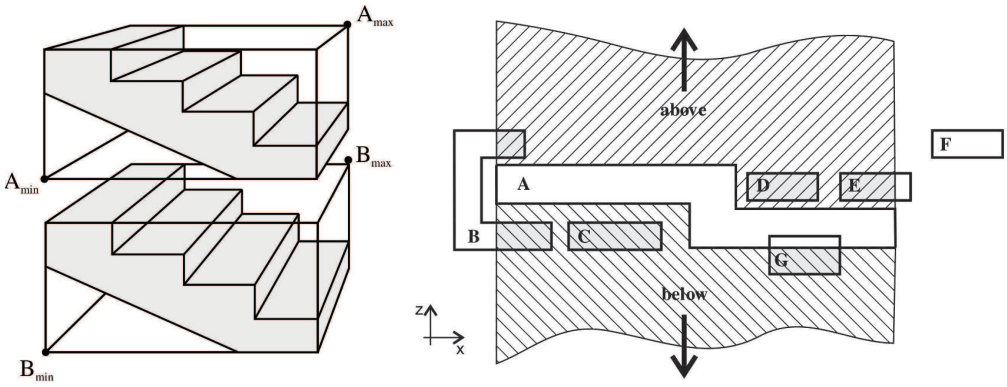


Figure 2.9: The model in (Borrmann and Rank, 2009) deals with direction between complex objects in 3D by projecting them into 2D planes.

A into 8 cone-shaped areas (see Figure 2.7a) or 4 partitions using a projection-based approach (see Figure 2.7b). Moreover, a neutral zone (i.e., an area around the reference object where no direction is defined) can be added to divide the 2D space into 9 areas (see Figure 2.7c). Later on, the author compared the three representations and showed that the projection-based representation with neutral zone was better than other ones in terms of spatial reasoning (Frank, 1996).

At a fine-grained level, the direction-relation matrix (Goyal and Egenhofer, 2001) dealt with direction between regions in 2D. Similar to the projection-based approach with neutral zone, this model approximates the reference object by its minimum bounding rectangle and thus 9 regions around it could be defined. Thereafter, the model uses a 3×3 matrix that allows to calculate the intersection of the primary region with the 9 regions. An algorithm in linear time for 2D vector-based regions is introduced in (Skiadopoulos et al., 2005).

With regard to 3D space, (Chen et al., 2007) extended the cardinal direction

model to 3D space, called TCD (Three-dimensional Cardinal Direction) model. The reference object is approximated by its axis-aligned minimum bounding box that partitions the 3D space around it into 27 directional relations according to 3 layers (upper, medium, and below) (see Figure 2.8). Because the TCD model did not take into account shapes of objects (e.g., concave and convex objects are treated in a same way using their bounding boxes), this may lead to wrong results in some situations. Some approaches tried to model directional relations among complex objects (such as stairs) by projecting them into 2D planes (Borrmann and Rank, 2009), as in Figure 2.9, or by calculating the intersection between cubic matrix (Chen and Schneider, 2010). However, these approaches have a big computational issue because they require to partition objects and the 3D space into cubic cells.

Orthogonal to the previous approaches, (Liu et al., 2005) considered the ICD (Internal Cardinal Direction) model. It is a special case in which the primary object is inside the reference object. Such a ICD model allows a distinction of directional relationships such as “Paris is in the north of France" and “England is to the north of France".

To summarise, the complexity of directional models is directly proportional with the multi-dimensionality of space. No directional model developed so far has efficiently dealt with direction among complex objects in 3D that remains an open research question. Instead, approximation structures such as bounding boxes or points are often used to simplify the specification of direction between objects.

2.2.4 Distance

Together with direction, distance is another important metric aspect of space. There are two main approaches to model distance relations: absolute distance and relative distance (Hazarika, 2005). In the first case, absolute distance between two spatial entities is directly computed based on their position. Examples of absolute distance are “A is 10 meters from B” or “A is close to B”. In the second case, relative distance between two spatial entities is obtained by compared to the distance to a third entity. An example of a relative distance is “A is closer to B than to C”.

As we can see through the examples, absolute distance can be represented quantitatively or qualitatively. A quantitative representation of absolute distance can be obtained by a simple computation when spatial entities are simplified as points (e.g., their centroid). Meanwhile, a qualitative representation of absolute distance relations is commonly obtained by dividing the space into several regions of different sizes. In (Hernandez et al., 1995), absolute distance is organised based on different levels of granularity, for example: a level with three distinctions (“close”, “medium”, and “far”), four distinctions (“very close”, “close”, “far”, and “very far”), or five distinctions (“very close”, “close”, “commensurate”, “far”, and “very far”), as showed in Figure 2.10a.

In contrast to absolute distance, relative distance is purely qualitative. There

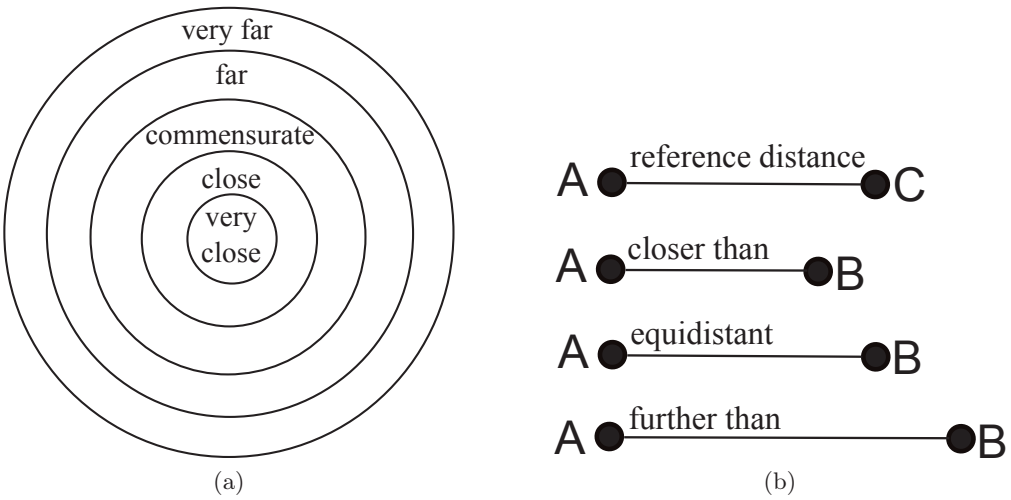


Figure 2.10: Illustration of absolute distance and relative distance.

are only three ternary qualitative relations: "closer than", "equidistant", or "further than", as illustrated in Figure 2.10b.

2.3 Conceptualising Spatial Constraints

As discussed in Section 1.1.6, existing approaches for semantic modelling of VEs have paid a less attention to spatial semantics. Current semantic models of VEs have focused on a high level representation of spatial entities. Nevertheless, spatial relationships between spatial entities have not been taken into account. This section investigates relevant approaches for conceptualising spatial relationships.

2.3.1 Object-oriented Approaches

Object-oriented approaches offer a natural way to conceptualise spatial relationships. In conceptual and object-oriented schemas, the definition of "class" (or "concept") naturally corresponds to the definition of spatial entities. Moreover, in a class diagram, it is possible to specify links between classes, such as composition or aggregation. As a consequence, the links between classes can be benefited to convey spatial relationships.

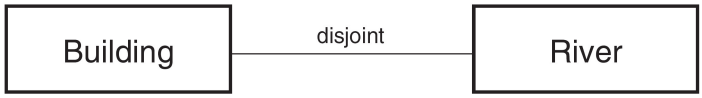


Figure 2.11: Illustration of object-oriented approaches for conceptualising spatial relationships. Spatial entities are mapped to classes. Spatial relations are represented by associations between classes.

GeoOOA is an object-oriented approach providing a conceptual framework to specify spatial relationships among geographical entities (Kosters et al., 1997).

Geographical entities (like streets, buildings, or rivers) are mapped into corresponding classes. GeoOOA only considers "whole-part" topological relationships among geographical entities, for example "contains", "part of", or "disjoint". These relations are conveyed by associations between classes. Figure 2.11 illustrates the representation of a spatial relation between two geographical entities.

Following the same object-oriented approach, MADS (Modeling of Application Data with Spatio-temporal features) is a framework to design spatio-temporal conceptual model (Parent et al., 1999). In MADS, spatio-temporal conceptual modes can be drawn using a graphical editor. Associations between classes are used not only to represent spatial relationships but also temporal relationships using Allen's temporal intervals (Allen, 1983). MADS provides a richer expression of spatial relationships. For example, cardinality of spatial relations can be described (e.g., "a park contains at least two trees").

In (Ber and Napoli, 2002), an object-based knowledge representation system, named Y3, is proposed for representing and classifying spatial structures. In Y3, there exists predefined association classes allowing a complete representation of topological relations defined in the RCC-8 model (see Section 2.2.1). That is, topological relations between objects are considered as instances of an association class, such as `DC` (disconnected), `EC` (externally connected), or `TPP` (tangential proper part) classes. Further, it is also possible to specify various quantifier such as "all", "none", "at least", or "at most" in topological relations.

Despite object-oriented paradigms provide a relevant way to represent spatial entities, the description of spatial relations by means of associations between classes however raises many issues. On the one hand, this type of representation may lead class diagrams to be overwhelming by associations. On the other hand, as pointed out in (Pinet et al., 2004), class diagrams are not sufficient to express spatial relations depending on a particular condition. For instance, Figure 2.11 implies that all the buildings must be disjoint with all the rivers. Particular spatial relations depending on the attributes of a building and a river have not been taken into account.

2.3.2 Spatial Languages

Another approach to conceptualise spatial relationships is to use spatial languages. In the object-oriented approaches, the definition of spatial relations was done by means of conceptual links between classes that led to a static view of spatial relations. Spatial languages provide a more flexible way for the specification of spatial relationships in conceptual schemas. Typically, spatial languages are strongly inspired from natural languages. Spatial languages can be logical languages, visual languages, or hybrid languages (i.e., a combination of natural, visual, and logical languages) (Salehi et al., 2007).

SQL-based Approaches

One of the first developed spatial languages is related to the use and ex-

tension of the Structured Query Language (SQL) to define spatial relations in a conceptual schema. *Spatial-SQL* (Egenhofer, 1994) is an example of such an extension. In *Spatial-SQL*, users can enter standard SQL queries to retrieve non-spatial data. As a spatial language, *Spatial-SQL* is incorporated with spatial operations to query topological relationships between geometrical objects. Moreover, it is possible to specify how to display spatial query results (e.g., the context, content, and visual properties of the graphical presentation of spatial queries). Inspired from *Spatial-SQL*, (Borrmann and Rank, 2009) extended this language to cover directional relationships.

OCL-based Approaches

An important family of spatial languages is concerned with the Object Constraint Language (OCL) (Warmer and Kleppe, 2003). OCL is an integral part of the Unified Modeling Language (UML) which is the standard language for modelling conceptual schemas. OCL is used to express constraints and expressions on any elements in the conceptual schemas represented in the form of UML models. It is both a constraint and a query language: an OCL expression returns a value or an object within the system, but the evaluation of expressions does not change the state of the system. The main advantage of OCL relies on its formal foundation and its expressiveness. OCL is able to cover various types of constraints related to attributes, operations, state transitions, or pre/post-conditions of a UML model.

Beyond the use of OCL to maintain constraints in the conceptual schemas, numerous researches have been conducted in extending this language to cover other complex constraint types. First, different propositions have been made to extend OCL for dealing with temporal constraints, see (Soden and Eichler, 2009) for a recent description. This makes OCL possible to cover the dynamic aspect of constraints over the time. Second, recent work has recognised OCL as the most relevant and pertinent approach to cover complex spatial constraints (Casanova et al., 2000). Several spatial extensions of OCL have been proposed. Similar to object-oriented approaches, *Spatial OCL* (Pinet et al., 2007) first aims at using UML for modelling spatial data, and then using OCL for specifying spatial constraint in Geographical Information Systems. *Spatial OCL* is limited to topological constraints. It provides eight topological operations: `overlaps`, `contains`, `isInside`, `areAdjacent`, `covers`, `isCovered`, `areDisjoint`, and `areEqual`. These operations correspond to Egenhofer's *9-intersection* topological model presented in Section 2.2.1. Similarly, *GeoOCL* (Werder, 2009) extended the traditional concept of UML class to geometrical class. A geometrical class is a UML class with an additional geometry attribute. Based on the concept of geometrical class, topological constraints between them can be specified using *GeoOCL*.

2.3.3 Geometrical Constraint-based Virtual Environments

From the point of view of applications, modelling spatial constraints among 3D objects has been addressed in several domains of application of VEs, notably in Computer-Aided Design (CAD) and Computer-Aided Manufacturing (CAM). In these domains, VEs are often used for assessing the conceptual design of mechanical systems.

However, it is important to note that spatial constraints in these VEs are purely geometrical constraints. The main goal of geometrical constraints-based VEs is to facilitate the accurate positioning of assembly objects in run-time, and the detection and maintenance of constraint consistencies during 3D manipulations. In these VEs, geometrical constraints are mainly modelled by means of scene graphs (Wang et al., 2003).

Figure 2.12: Design of a simple house using geometrical constraints, proposed in (Fernando et al., 1999).

(Fernando et al., 1999) proposed an architecture for geometrical constraint-based VEs. This approach first defines a set of possible assembly relationships between two objects. An assembly relationship depicts information such as the contact between the two objects (e.g., a surface, a plane, a line, or a point), or the possible motion permitted for the two objects (e.g., rotation or translation). Then, the assembly objects are loaded into VEs and represented by a scene graph. The user is free to grab and manipulate objects in the 3D space. When the objects collide, the contacting surfaces are identified through the scene graph. Based on information of the contacting surfaces and the colliding objects, it is possible to identify the geometrical constraints to be satisfied. Finally, only the valid motions on the assembly objects are allowed, without breaking the existing constraints. Figure 2.12 illustrates the design of a simple house using

geometrical constraints.

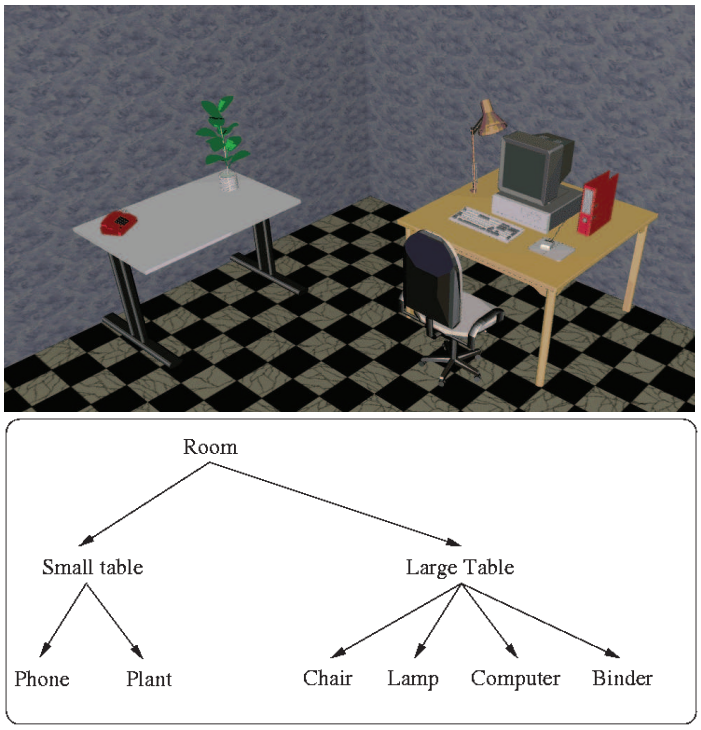


Figure 2.13: Maintaining geometrical constraints in VEs using scene graphs, proposed in (Stuerzlinger and Smith, 2002).

Geometrical constraints have been used to facilitate the manipulation of objects in Augmented Reality, and the interaction with objects in Virtual Reality applications (Smith and Stuerzlinger, 2001; Stuerzlinger and Smith, 2002; Smith and Willans, 2006). In these applications, similarly to Fernando *et al.*'s approach, geometrical constraints are conceptualised by means of scene graphs, see Figure 2.13. Virtual objects in the scene are associated with predefined constraint areas. A constraint area is defined as a convex polygon, and an orientation vector. A constraint area of an object can be an "offer area", or a "binding area". An offer area identifies locations on an object where other objects are allowed to contact. For instance, an offer area of a table is its top surface where the user can place objects on. In contrast, a binding area depicts locations on an object where this object can constrain to an offer area. For example, a binding area of a chair should be below its base, therefore the chair could be placed on the offer area of the floor of a room. When a binding area constrains itself to an offer area, the binding area has been "satisfied". Based on this mechanism, it allows a more realistic control while interacting with virtual objects. For example, when a chair has been constrained to a floor, the interaction with the chair is restricted such that the binding areas of the chair always remain in contact with the floor.

In the above approaches, geometrical constraints are useful to control direct

manipulations of objects in 3D spaces. However, they can not specify meaningful spatial constraints between virtual objects. For example, in Figure 2.13, geometrical constraints allow to describe that "the small table and the large table can be *moved* on the floor". Whereas, it is impossible to specify that "the small table is behind and on the left of the large table".

2.3.4 Discussion

We have reviewed the relevant approaches for specifying spatial relationships at a conceptual level. The emphasis has been on high level and formal spatial languages and frameworks. It has been observed that object-oriented paradigms provide a relevant basis for the spatial conceptualisation. In a conceptual schema, the definition of concepts (or classes) naturally meets the definition of spatial entities. Most of the approaches aimed to model spatial relationships in a conceptual schema by means of associations between concepts. Nevertheless, the main bottleneck of these object-oriented approaches is that they are unable to express conditional and dynamic spatial relationships between concepts.

To address this obstacle, spatial languages have been introduced into conceptual schemas. The syntax of spatial languages is often designed to be as close as possible to the natural language. Their semantics commonly ground on a logic formalism. In addition, spatial languages facilitate the expression of spatial relationships by means of graphical editors, so called visual languages. Therefore, spatial languages provide a more formal and flexible way to express spatial relationships in a conceptual model. For instance, the Structured Query Language (SQL) has been used and extended to model integrity constraints in spatial data models. Similarly, the Object Constraint Language (OCL) has been defined as an integral part of the conceptual models built upon the standardised modelling language UML. However, existing extensions accounting for spatial constraints are still very limited.

Finally, several approaches for the design of VEs based on geometrical constraints have been discussed. In these VEs, geometrical constraints are used to support an accurate manipulation of virtual objects during a virtual assembly, or to provide a more realistic interaction with virtual objects. Nevertheless, geometrical constraints are essentially different from the definition of semantic spatial constraints as being interested in our work. Geometrical constraints remain a low level description that is not expressive enough for human and agents understanding and reasoning.

2.4 Summary

In this chapter, we have respectively described i) taxonomies, ii) theoretical models, and iii) conceptual solutions that are relevant for representing spatial constraints among entities in VEs.

We have studied spatial constraints within different dimensions, based on spatial characteristics (i.e., metric vs. non-metric, quantitative vs. qualitative,

and static vs. dynamic). The main conclusion is that topological, projective, directional, and distance constraints stand for the most fundamental spatial aspects constituting a human mental description of space as well as enabling a high-level spatial reasoning for humans and agents.

Thereafter, the four main types of spatial constraints have been investigated. The main remark is that a qualitative approach has been proven as a pertinent approach for representing spatial constraints.

Finally, the related solutions for conceptualising spatial constraints, that are susceptible to be applied in VEs, have been discussed. The object-oriented paradigm appears to be a relevant basis for conceptual modelling of VEs. Furthermore, a conceptual schema could be integrated with a spatial language to provide a flexible management of spatial constraints in a conceptual level.

Part II

Contributions

Chapter 3

An Integrated Model of Spatial Constraints For Virtual Environments

In this chapter, we propose an integrated model of spatial constraints dedicated to VEs. First, we model the concepts of spatial entity and frame of reference. Then, we respectively tackle the four main families of spatial constraints, they are topological, projective, distance, and directional constraints. The integrated model serves as a theoretical and computational basis for our language and framework for conceptualising spatial semantics that are described in Chapter 4.

This chapter is organised as follows. Section 3.1 presents a simple but realistic VE, both to provide the examples of spatial constraints, and to illustrate our model later on. Section 3.2 and Section 3.3 respectively describe how spatial entities and frames of reference are represented in our model. Our model of nonmetric spatial constraints (i.e., topological and projective constraints) is respectively presented in Section 3.4 and Section 3.5. Next, our approach to modelling metric spatial constraints (i.e., directional and distance constraints) is respectively detailed in Section 3.6 and Section 3.7. Finally, Section 3.8 summarises and discusses our model.

3.1 Examples of Spatial Constraints In a Simple Virtual Environment

Figure 3.1 illustrates a simple but realistic VE. This VE represents a room. The room is composed of a floor, a ceiling, and several walls. In the room, there are one desk and three tables: a green triangular one, a red rounded one, and a blue rectangular one. There exists an important difference between a desk and a table. That is, the orientation of the desk is fully defined. One can state which are the front, left, and above sides of the desk. For the table, only the above side can be defined. In the room, there are also three chairs: a red one, a black one, and a yellow one. There are also other objects, such as a dice, a blue

Figure 3.1: A simple virtual environment illustrating a room. The labels were added to indicate the sides (i.e., front, left, and above) of the desk. This simple virtual environment is used to illustrate spatial constraints.

box, a green cone, or a red sphere. Figure 3.2 shows a closer look to the blue rectangular table and some other object such as the sphere and the cone.

A variety of spatial constraints can be defined in this simple VE. Constraint 1 exemplifies a spatial relationship between the sphere and the blue rectangular table.

Constraint 1. *The sphere must be on (meeting) the blue rectangular table.*

Similarly, the following constraint can be described.

Constraint 2. *The cone is disjoint with the sphere, and both of them are on the table.*

Constraint 3. *In the room, every table must be disjoint with other tables, and all of the tables must be on the floor.*

In the above examples, spatial constraints such as "meet" or "disjoint" are based on the connectedness among spatial entities. As we will see later in Section 3.4, these spatial constraints are called *topological constraints* because they are preserved under topological transformations like translation, scaling, or rotation of spatial entities.

From an other perspective, Constraint 1, Constraint 2, and Constraint 3 are examples of *binary spatial constraints*. A binary spatial constraint aims to locate

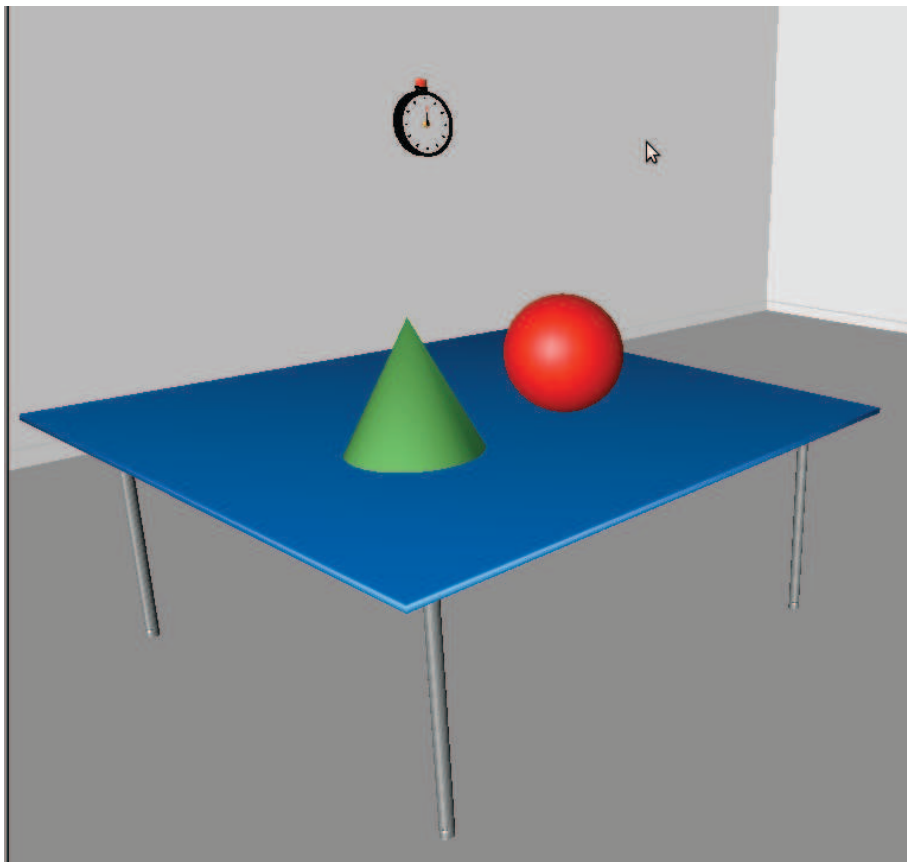


Figure 3.2: A closer view to the blue rectangular table of the room presented in Figure 3.1. In this case, an example of spatial relationships is that "the sphere is disjoint with the cone, and both of them are on (meeting) the table".

an entity, called *primary entity*, within a spatial relationship to another entity, called *reference entity*. Alternatively, spatial relationships can be established among three, four, or many spatial entities. In these cases, more than one entity is used as reference entities. Accordingly, spatial constraints can be *ternary*, *quaternary*, or *n-ary constraints*. Constraint 4 is an example of ternary spatial constraints.

Constraint 4. *In the room, the black chair must be between the red rounded table and the blue rectangular table.*

In this example, both the red rounded table and the blue rectangular table play the role of reference entities that allow to define the relative position of a primary entity – the black chair.

Similarly, Constraint 5 exemplifies a quaternary constraint. Here, all of the three tables are used as reference entities.

Constraint 5. *In the room, the red chair must be coplanar and surrounded by the three tables: the triangular green table, the rounded red table, and the*

rectangular blue table.

Constraint 4 and Constraint 5 aim to define spatial relationships such as "between", "surrounded", or "coplanar". The "between" relationship is based on the collinearity (i.e., alignment) among spatial entities. Meanwhile, the "surrounded" and "coplanar" relationships rely on the coplanarity among spatial entities. As we will see later in Section 3.5, these relationships are examples of *projective constraints* that are preserved under projective transformations.

In the room, an important difference between the desk and the other tables is that the desk has an *intrinsic direction*. The intrinsic direction allows to distinguish the left/right side, the front/behind side, or the above/below side of the desk. As we will see later in Section 3.6, the intrinsic direction of the desk can be modelled by the three vectors $\vec{front}$, $\vec{left}$, and $\vec{above}$. Based on these three vectors, other sides of the desk can be inferred. The following examples show spatial constraints found in the room based on the intrinsic direction of the desk.

Constraint 6. *In the room, the rounded red table must be in front of the desk.*

Constraint 7. *In the room, the triangular green table must be on the left of the desk.*

Constraint 8. *In the room, the rectangular blue table must be in front and on the left of the desk.*

Constraint 6, Constraint 7, and Constraint 8 are examples of another family of spatial constraints, called *directional constraints*. Here, direction is given from a first-person perspective (also called ego-centric direction). Alternatively, directional constraints can also be observed from a third-person perspective (also called allo-/exo-centric direction). Figure 3.3 gives a different view to the room that is centred on the viewpoint of the desk. Using the desk as the point of view, the following examples illustrate direction from a third-person perspective.

Constraint 9. *From the viewpoint of the desk, the rectangular blue table must be on the left of the rounded red table.*

Constraint 10. *From the viewpoint of the desk, the triangular green table must be behind of the rectangular blue table.*

Constraint 9 and Constraint 10 are binary, but they are viewed from a third-person perspective. In these cases, direction between a primary entity and a reference entity is only based on the intrinsic direction of the viewer (i.e., the desk). Meanwhile, the intrinsic directions of the primary entity and the reference entity (i.e., the other tables) have no impact on the description of direction.

Beside directional constraints, *distance constraints* provide a further important metric description of spaces. Our model of distance will be described in Section 3.7. Constraint 11 exemplifies a constraint about distances between spatial entities in the room.

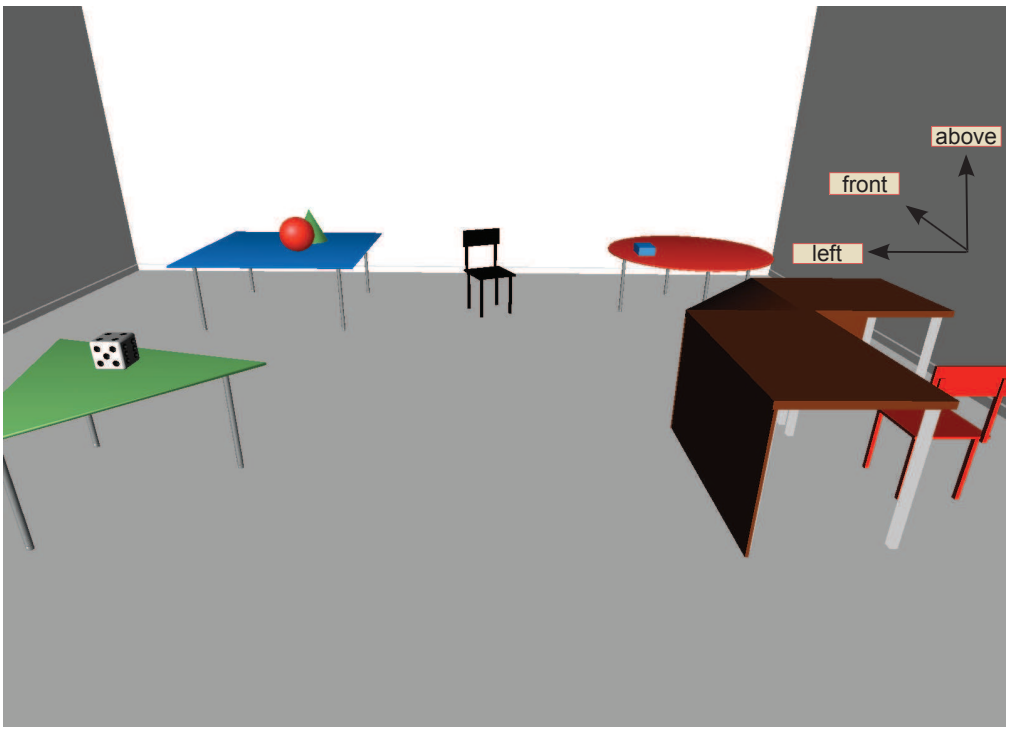


Figure 3.3: A different view to the room that is centred on the viewpoint of the desk. Using the desk as a reference frame, the following example of direction from a third-person perspective can be stated: "From the viewpoint of the desk, the rectangular blue table is on the left of the rounded red table".

Constraint 11. *In the room, the red rounded table is located closer than 1 meter from the desk.*

Up to now, every spatial expression described above merely illustrates a specific type of spatial constraints. Nevertheless, it is possible to combine different types of spatial constraints in a single spatial expression. Constraint 12 provides a combination of topological and projective constraints. Constraint 13 illustrates a combination of topological, directional, and distance constraint.

Constraint 12. *In the room, the black chair must be on the floor, between the red rounded table and the blue rectangular table.*

Constraint 13. *In the room, the red rounded table must be on the floor, in front of and closer than 1 meter to the desk.*

In the following sections of this chapter, we will show how each family of spatial constraints is represented in our model. Such a representation of spatial constraints is very useful in many contexts. For example, Figure 3.4 illustrates a scenario in which a user must realise a spatial task. That is, "the user must put the cone on the dice". Here, spatial constraints are very important. The satisfaction of spatial constraints allow to know whether the user's interaction

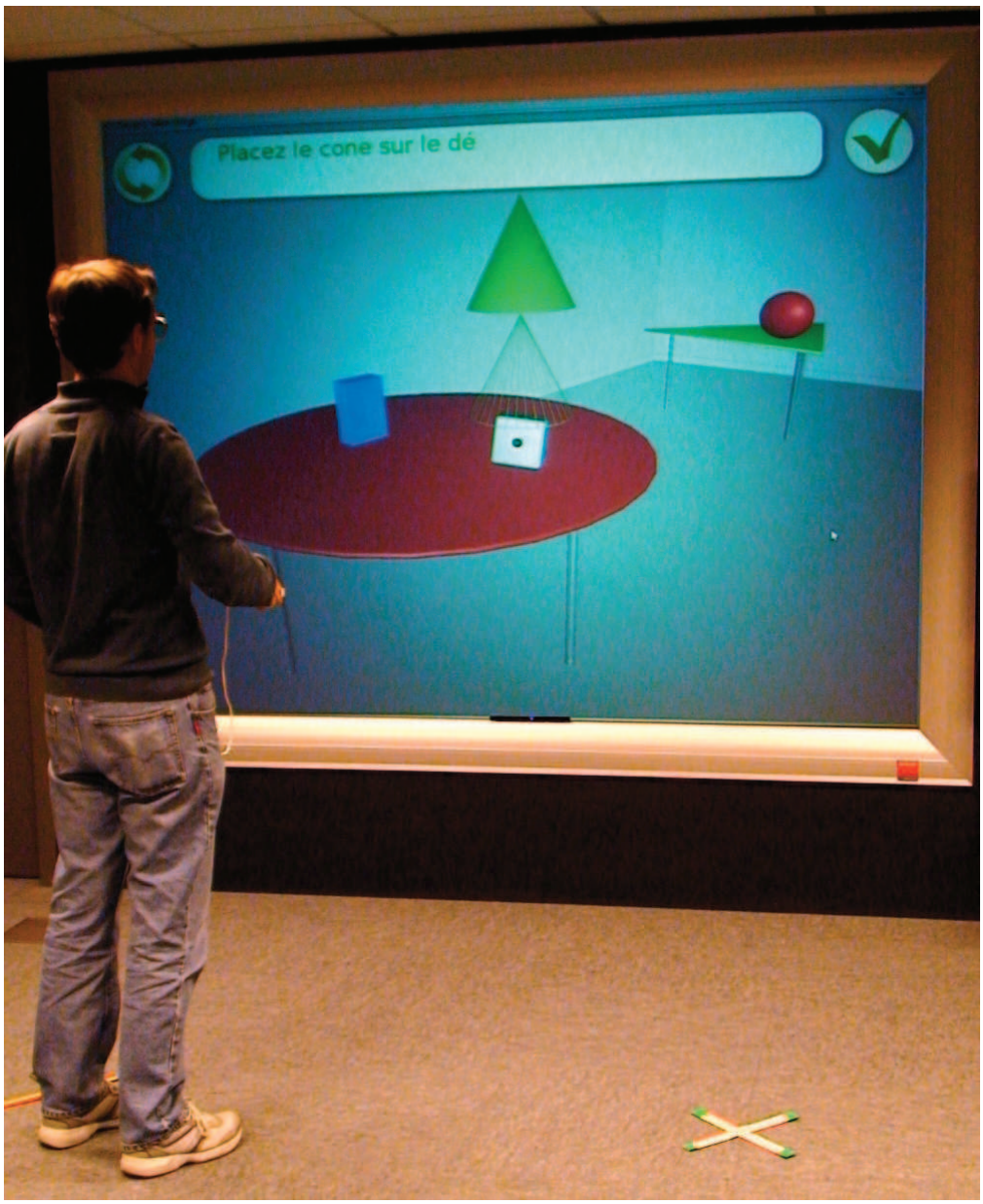


Figure 3.4: A user is manipulating objects in the room using a device (i.e., a wiimote). The goal is to satisfy a spatial configuration, such as "the user must put the cone on the dice".

is already accomplished or not. A more detailed description of the applications of spatial constraints in VEs will be given in Chapter 5.

3.2 The Notion of Spatial Entity

To model spatial relationships, it is primordial to model spatial entities. Our intention here is to identify the essential information of every spatial entity that is

required to model spatial relationships between them. This information should be generic and independent of any geometrical and conceptual representation of spatial entities, such as scene graphs or spatial ontologies. Therefore, spatial entities are studied as separated entities. Conceptual links between spatial entities (such as structural, logical, compositional, or whole-part links) are not discussed.

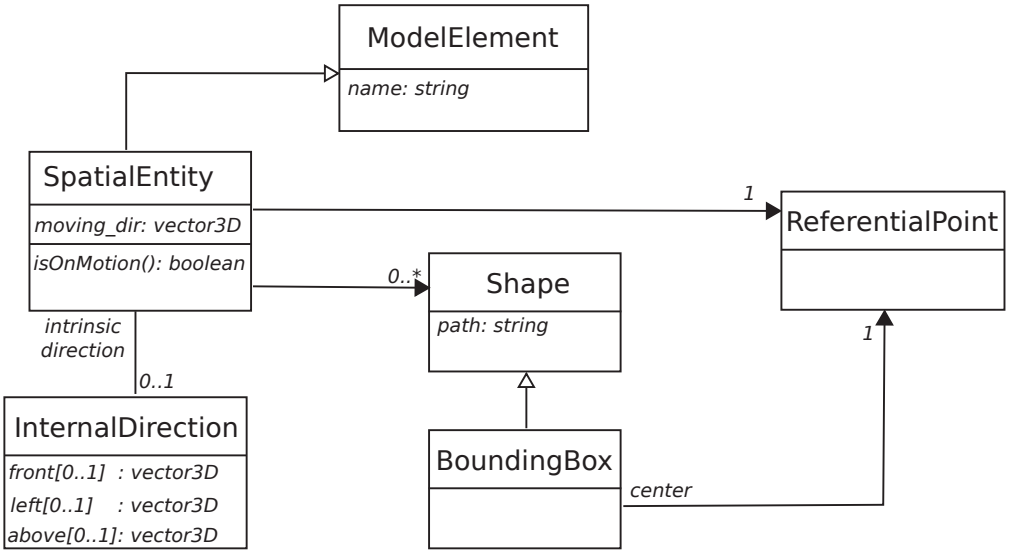


Figure 3.5: The conceptual definition (in the form of a UML class diagram) of spatial entity.

Figure 3.5 illustrates our conceptual definition of spatial entity in the form of a UML (Unified Modeling Language) class diagram. Within a VE, a spatial entity can be a spatial object (tangible or not). In this case, a spatial entity corresponds to a scene entity but with additional semantics. Otherwise, a spatial entity can be an artificial agent (e.g., a virtual human), or a user. In our approach, we abstract spatial entities as a kind of elements in a conceptual model of VE. A spatial entity is geometrically represented by some *shapes*. However, the geometrical representation of a spatial entity is separated from the further semantic information: *name*, *referential point*, *bounding box*, *intrinsic direction*, and *moving direction*. This added information is needed to specify spatial constraints among spatial entities. The elements of our conceptual model of spatial entity are explained as follows.

- Name: Every spatial entity is identified by a unique name. For example, "triangularTable", "roundedTable", and "rectangularTable" are respectively the unique names of the three tables in the room. Our motivation is that, spatial constraints may be given for a family of spatial entities in a very general manner (e.g., "all of the tables must be on the floor"), or for specific spatial entities with different types (e.g., "the red sphere must be on the blue rectangular table"). In the first case, spatial constraints do

not mention any spatial entity in particular. However, in the latter case, it is necessary to identify which spatial entities are actually included in a spatial constraint. In our model, a unique name is used to allow a uniform access to a concrete spatial entity involved in a spatial constraint.

- Shape: refers to the geometric structures of a spatial entity. It indicates the path to 3D file formats like VRML or X3D.

Since the geometrical structures of spatial entities are complex and thus not relevant to manipulate or calculate, they are often approximated using simpler structures. For instance, spatial objects in 2D spaces are represented by vector-based polygons, but to compute spatial relations among them, they are often simplified as points or minimum bounding rectangles (Papadias and Theodoridis, 1997). In our model, to facilitate the computation of spatial constraints, we hence use two levels of approximation of spatial entities.

- Referential point: Every spatial entity is associated with one *referential point*. A referential point conveys semantics about a spatial entity. In most cases, a spatial entity is referred as its position in space. The logic is that, for instance, a chair is considered as "on the left" of a table when the position of the chair is "on the left" of the table. Otherwise, a referential point can be an interaction point, or a marker that facilitates the navigation between spatial entities in VEs. For example, it is possible to define the referential point of a chair as the point of sitting in the chair, or the referential point of a door can be defined as its handle. In this case, a referential point is similar to the concept of *hand clues* of *Smart objects* (Kallmann and Thalmann, 1999) that defines the expected location in order to perform an interaction with an entity.

- Bounding box: A spatial entity is described by its *axis-aligned bounding box* (AABB). The use of AABB at the conceptual level makes spatial entities easy to compute, cheap to store (only two points are needed for their representation), and fast to test for intersection. Furthermore, this primitive structure provides a convex approximation that is detailed enough to deal with spatial constraints. Although algorithms in the rest of this chapter for computing spatial constraints are detailed using AABB, further implementations can be similarly derived for other levels of approximation based on bounding volumes.

Figure 3.6 illustrates some spatial entities in the room enhanced with their additional semantics. The chair and the table are displayed with their bounding boxes. The referential point of the chair is defined as its point of sitting. Meanwhile, the referential point of the table is defined as the centre of the top surface of the table.

With regard to direction and motion of spatial entities, they are represented in our model as follows:

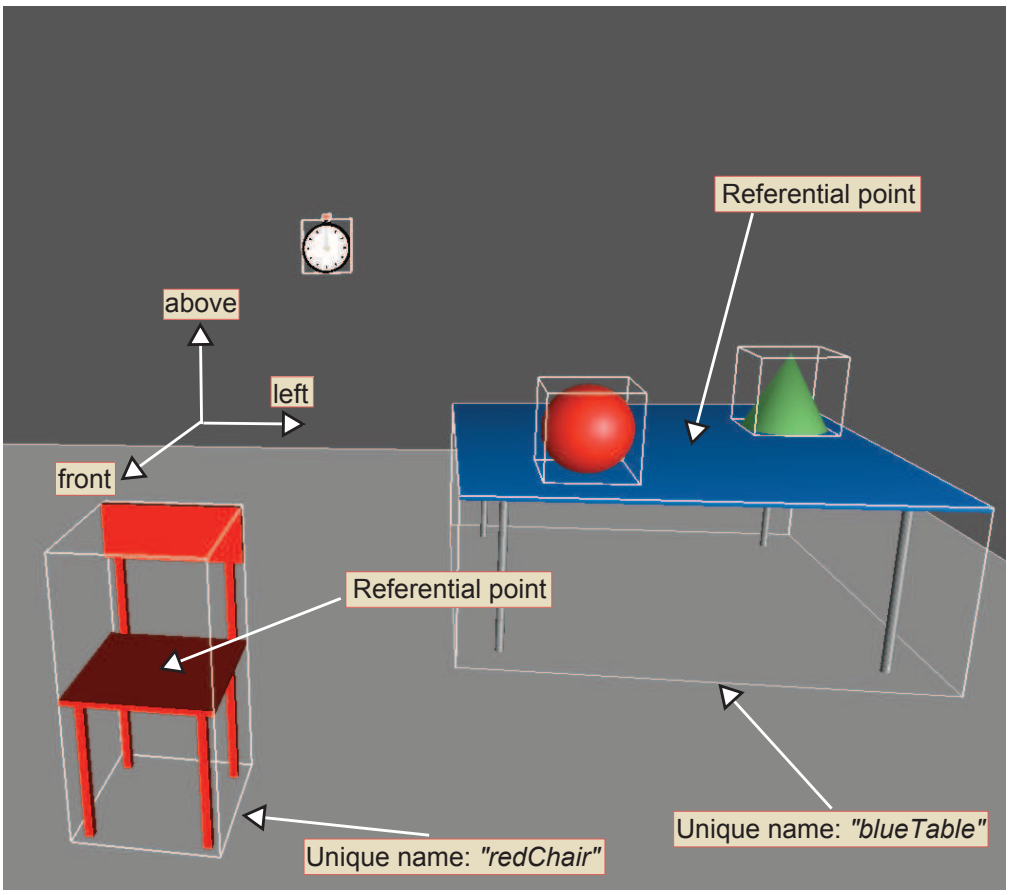


Figure 3.6: Illustration of the model of spatial entity. The *unique name* of the chair and the table are respectively "redChair" and "blueTable". They are displayed with their *bounding boxes*. The *referential point* of the chair is its point of sitting. The referential point of the table is the centre of its top surface. The chair has an *intrinsic direction*.

- Intrinsic direction: A spatial entity is *possibly* oriented, such as a desk, a ship, or a house. Oriented spatial entities have *intrinsic directions* that indicate their left/right, front/back, and above/below sides. We represent such intrinsic information by means of three unit vectors $\vec{front}$, $\vec{left}$, $\vec{above}$ that respectively define the front, left, and above side of an entity. Other intrinsic directions of an entity (i.e., "back", "right", and "below") can be inferred from these unit vectors. In Figure 3.6, the chair is oriented and its intrinsic direction is indicated in the figure. However, there also exists non-oriented or partially-oriented spatial entities. A sphere is an example of non-oriented entities. Whereas, a table is an example of partially-oriented entities, because one can easily identify the vertical axis (i.e., above/below sides) of a table, but it is difficult to know which are its front/back or left/right sides. As we have seen earlier in the examples, only directional constraints need intrinsic directions of spatial entities to

express direction among them. Other types of spatial constraints such as topological, projective, or distance constraints are defined among spatial entities without intrinsic direction.

- Moving direction: Spatial entities are not necessary still but possibly moving in 3D space. For example, an agent can walk in the room, a ship can move on the sea, a virtual human can walk from an initial place to a target, or a user is free to move in a VE. Here, we are interested in modelling the motion of spatial entities. A motion of an entity is represented by the *moving_dir* attribute that defines the front, left, and above direction of the motion.

Compared to the prior work, our model provides a more abstract representation of spatial entity with regard to the related approaches such as *smart objects*, *semantic objects*, or *basic semantic level* (cf. Table 1.2 on page 30). In our model, a *unique name* plays the role of an *object identifier*, meanwhile a *referential point* itself contains the *position* or *navigation point* of an object. Based on the *bounding box*, it is possible to extract information about the *width*, *height*, and *depth* of an object. Most of the elements of our model can be automatically computed. Our definition of *intrinsic direction* provides a more user-oriented semantics of direction of a spatial entity than the geometrical *rotation* or *orientation*. Also, the introduction of *moving direction* enables to model the spatial dynamics of spatial entities that were neglected in the previous models. Finally, both the definitions of *intrinsic direction* and *moving direction* allow to formalise the important concept of *frame of reference* detailed in the following section.

3.3 Modelling Frames of Reference

Most spatial relations must be given with respect to a *frame of reference* (FoR). A FoR defines the context in which a spatial relation is given. FoRs are often referred as different perspectives to spatial relations. That is, a FoR describes the *point of view* from which a spatial relation is observed. Different points of view can yield different results to the same spatial relation. Among the four families of spatial constraints, topological, projective, and distance constraints are relatively independent from FoRs. Whereas, directional constraints require an explicit definition of FoR.

There exists various types of FoRs. A FoR can be *absolute* or *relative*. In the first case, spatial relations are described using absolute landmarks, such as magnetic poles (e.g., north, south, west, or east). In the later case, spatial relations are relative to intrinsic properties of the user's point of view. More precisely, relative FoRs can be divided into *first-person perspective* or *third-person perspective*. As we have seen in Section 3.1, examples of first-person perspective are Constraint 6, Constraint 7, and Constraint 8. Whereas, Constraint 9 and Constraint 10 illustrate third-person perspectives.

Previous studies in the field have raised the need to support different FoRs in VEs. Salamin et al. (2006) showed that, in VEs, users prefer a third-person

perspective in some tasks such as navigation, displacement actions, or interaction with moving objects. Whereas, a first-person perspective is preferred in other tasks, such as manipulating an object with the hands. With regard to distance, users better evaluate the distances, anticipate and extrapolate the trajectory of moving objects using a third-person perspective. Later on, the authors showed that a switching between the two views is also needed (Salamin et al., 2009). Alternatively, Schafer and Bowman (2004) studied how different FoRs could be combined in collaborative VEs. Users are embedded into VEs with two different roles (i.e., director and actor) to perform tasks. A director can provide instructions based on his perspective (e.g., "in front of me" or "go to my left") or the actor's perspective (e.g., "beside you"). The results showed that a combination of different FoRs yields a better collaboration in some tasks. Although the benefits of FoRs in VEs have been recognised, previous studies merely provide experimental and evaluational models of the impact of FoRs in VEs. No previous work in the field has dealt with the modelling and computing aspects of such an important concept.

To overcome this limitation, we propose a model of FoR for VEs. The model of FoR would clarify the contextual information of spatial constraints. Spatial constraints are thus unambiguous and consistent. Our intention is to keep FoRs generic and domain-independent such that FoRs can be specified at the conceptual level. According to Hernández (1994), in general, three types of FoR can be distinguished:

(i) intrinsic – spatial relation is given by inner properties of the reference entity. A typical example of this kind of FoR is directional constraints from a first-person perspective. Direction is given based on the intrinsic direction – an inner property – of the entity.

(ii) extrinsic – spatial relation is imposed by external factors on the reference entity. An example of extrinsic FoRs is when talking about the direction of a moving entity. Considering a car that is moving backward, the description of direction with regard to the car should be based on its motion direction, not its intrinsic direction.

(iii) deictic – spatial relation is based on a third-person view from which the reference object is seen.

However, in the context of VEs, spatial constraints are sometimes defined on reference entities without mentioning any explicit FoR. An example could be "in the room, the round table is on the left of the square table", whereas it has been noted earlier that a table has not an intrinsic direction. We thus propose that a VE may have an implicit FoR. It can be a spatial entity assumed as the default viewer. Otherwise, an implicit FoR can be based on common sense landmarks, such as magnetic poles. Such an implicit FoR is often used in (but not limited to) indoor spaces. For example, within a room, it is commonly accepted that the entrance door is the point of view by default to the room. In this case, the entrance door plays the role of the implicit FoR.

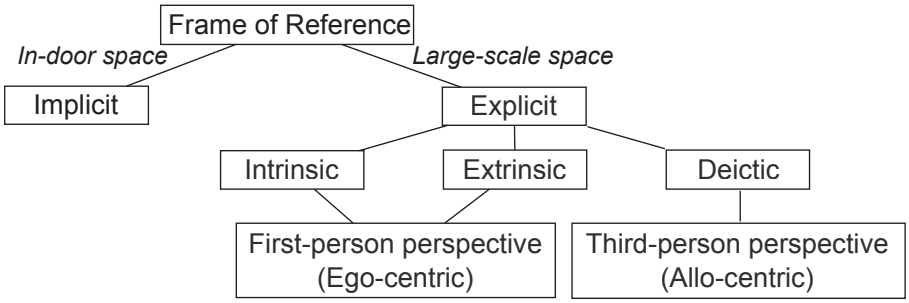


Figure 3.7: Taxonomy of different FoR in VEs.

Figure 3.7 illustrates our model of FoR. Using the concept of spatial entity presented above, the use of each type of FoR is as follows.

- When a spatial constraint is given with respect to a viewer (e.g., "*A sees that B is on the left of C*"), this is the case of **deictic FoR** . The spatial constraint is called **third-person perspective** constraint (also called allo-/exo-centric constraint).
- Otherwise, when no viewer is given (e.g., "*B is on the left of C*"), the spatial constraint is based on the reference object (entity C in the example), so called **first-person perspective** or ego-centric constraints. There are three possibilities.
  - First, if the reference object C is moving (i.e., it is impacted by external factors), the direction of motion (defined by the *moving_dir* attribute) will be used as a reference system to compute spatial constraints. This is the case of **extrinsic FoR** .
  - Second, if the reference object C is not moving and has an intrinsic direction, this intrinsic direction will be used. This is the case of **intrinsic FoR** .
  - Finally, if the reference object C is not moving and has not an intrinsic direction, the **implicit FoR** defined in the VE will be used.

Obviously, our model allows a unambiguous selection of FoR. In the following sections, we will show how our model of FoR is applied to specify the context of spatial constraints when necessary.

3.4 Modelling Topological Constraints

In this section, we aim at modelling topological constraints. Topological constraints allow to specify different levels of intersection between two spatial entities. A spatial entity can be completely separated with another one (e.g., "in the room, all the tables must be disjoint"). Otherwise, an entity can have a tight intersection with another one (e.g., "the user must put the sphere on the rectangular table").

The two well-known models of topological relations are *RCC-8* (Randell et al., 1992) and *9-intersection* (Egenhofer and Franzosa, 1991). These models defined eight possible topological relations between two regions in 2D, they are named as "disjoint", "meet", "overlap", "coveredBy", "covers", "inside", "contains", and "equal". Each relation corresponds to a level of intersection between two 2D regions, as illustrated in Figure 3.8. The main advantage of topological relations to VEs is that they are preserved under topological transformations (i.e., translation, rotation, and scaling). Also, topological relations are independent from FoRs.

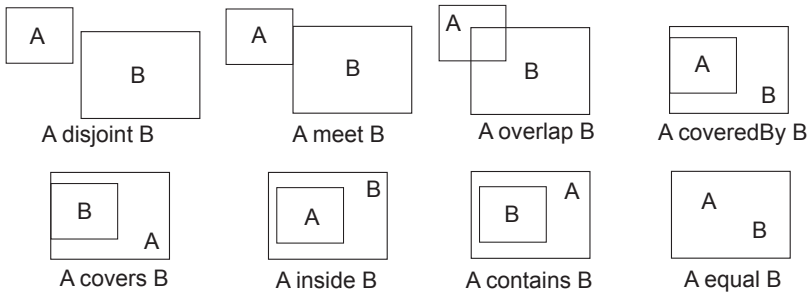


Figure 3.8: Illustration of the eight basic topological relations in 2D according to the *RCC-8* and *9-intersection* models.

However, applying the *RCC-8* and *9-intersection* models to VEs raises many computational and interaction issues. Both of the two models are based on an exact computation of the intersections between objects with sharp boundary. Whereas, in VEs, users manipulate objects using interface devices. As seen in Figure 3.4, the user was required to use a wiimote to accomplish a spatial interaction, such as "the user must put the cone on the table". Such a spatial interaction is a procedure lacking in precision. That makes the manipulation *"put the sphere on the table"* become difficult if objects involved must have common sharp boundaries.

To overcome the above issues, we decided to approximate topological constraints. Our main inspiration was the previous models that attempt to deal with *uncertain* (or *fuzzy*) topological relations in 2D spaces. In these models, the main assumption was that, 2D regions have not shape boundary but vague shape, such called *broad boundary region*. Figure 3.9 provides an example of

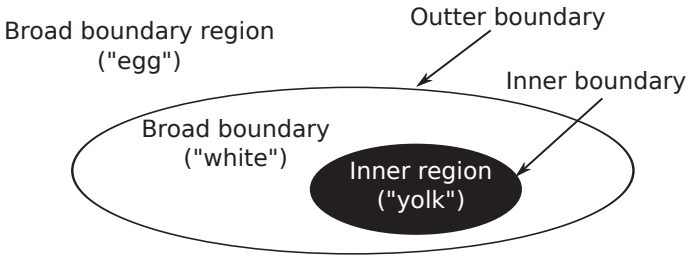
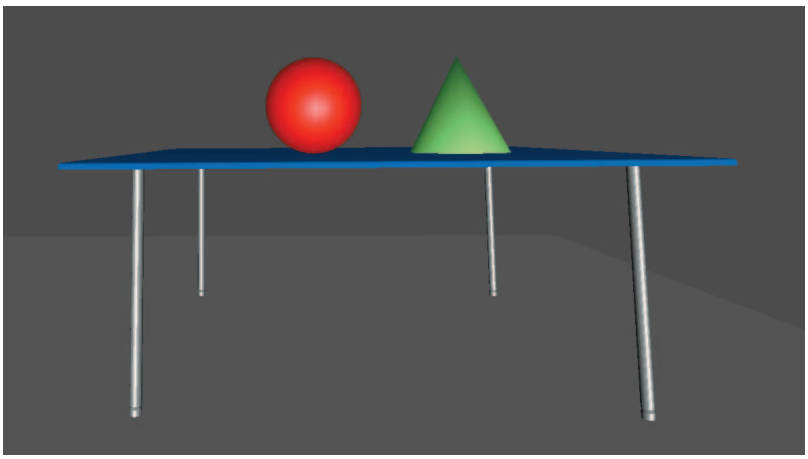
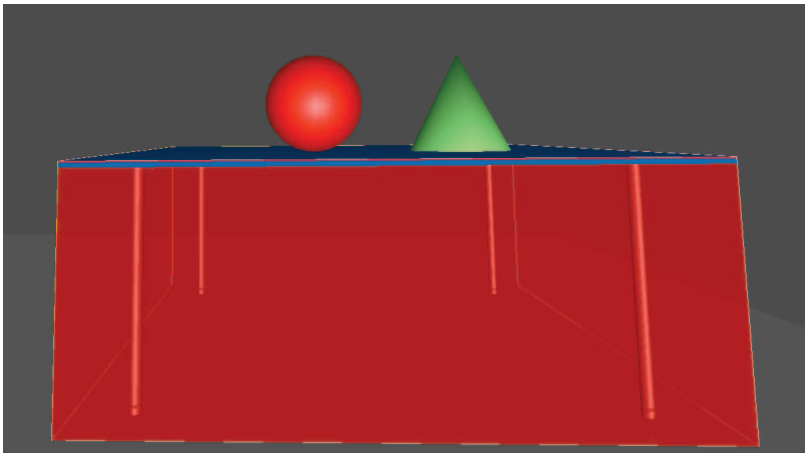


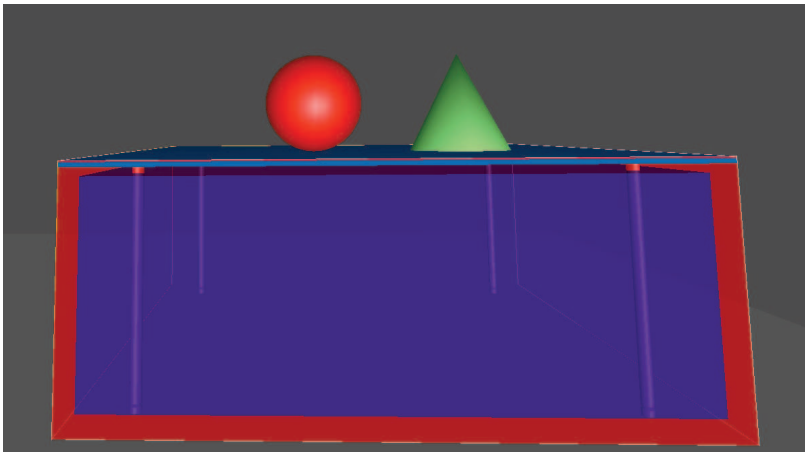
Figure 3.9: Example of a broad boundary region.



(a) A spatial entity (e.g., a table).



(b) A spatial entity with its bounding box.



(c) A spatial entity with its thick boundary.

Figure 3.10: Example of a spatial entity without and with thick boundary.

broad boundary regions. Based on the *RCC* model, the "egg-yolk" model identified 46 possible topological relations between broad boundary regions (Cohn and Gotts, 1996). Alternatively, based on the *9-intersection* model, the "approximate topological model" identified 44 topological relations between broad boundary regions (Clementini and Di Felice, 1997). There exists a difference with regard to the number of relations between the two models. The reason is that the "approximate topological model" model simplifies several special relations of the "egg-yolk" model into a single relation. Later on, the Qualitative Min-Max (*QMM*) model proposed in (Bejaoui et al., 2009) enumerated in an exhaustive manner all of 242 possible topological configurations between broad boundary regions. The main difference between the *QMM* model and the two previous models is related to how a broad boundary region is formalised. In the "egg-yolk" and "approximate topological model" models, the inner region (the *yolk*) is a proper part of the broad boundary region (the *egg*). Whereas, it is not the case in the "QMM" model in which the inner region is either a proper part or a tangential proper part of the broad boundary.

In our model, based on the concept of broad boundary region, we define the concept of *thick boundary objects* (TBOs). A TBO is a spatial entity represented by its AABB whose surfaces have a thickness $\varepsilon > 0$. We also suppose that spatial entities are large enough so that every dimension of their AABBs is larger than 2ε. For example, Figure 3.10 illustrates a table whose thick boundary is the buffer zone inside the object, between the red and the blue parts. Accordingly, every TBO can be decomposed into three parts: the exterior (the set of points not contained in AABB), the boundary (the set of points contained in the thick surfaces), and the interior (the set of points contained in AABB but not in the thick surfaces). With regard to a TBO A, its exterior, boundary, and interior are denoted as A^-, A^ε, and A^o respectively. Similar to the *9-intersection* model, topological relations between the two TBOs A and B are defined by a 3x3 matrix representing the nine intersections between their six parts.

$$I_{TBO}(A,B) = \begin{pmatrix} A^o \cap B^o & A^o \cap B^\varepsilon & A^o \cap B^- \\ A^\varepsilon \cap B^o & A^\varepsilon \cap B^\varepsilon & A^\varepsilon \cap B^- \\ A^- \cap B^o & A^- \cap B^\varepsilon & A^- \cap B^- \end{pmatrix} \tag{3.1}$$

However, until now, the number of uncertain topological relations between TBOs in 3D spaces has not yet been defined. To do so, we decided to project TBOs into 2D planes. The reason is that, uncertain topological relations between 2D objects have been identified, as in the "approximate topological" model or the "egg-yolk" model discussed above. By doing an orthographic projection of the two TBOs A and B to one of the three planes of the Cartesian coordinate system, the result is two *rectangles* with *broad boundary* A_{Oth}, B_{Oth}. The broad boundary rectangles (BBRs) A_{Oth}, B_{Oth} are special cases of broad boundary regions. Their interiors, boundaries, and exteriors are respectively noted as A^o_{Oth}, A^ε_{Oth}, A^-_{Oth} and B^o_{Oth}, B^ε_{Oth}, B^-_{Oth}. Consequently, the intersection matrix

between two BBRs is easily defined, called $I_{BBR}(A, B)$.

$$I_{BBR}(A,B) = \begin{pmatrix} A^o_{Oth} \cap B^o_{Oth} & A^o_{Oth} \cap B^\varepsilon_{Oth} & A^o_{Oth} \cap B^-_{Oth} \\ A^\varepsilon_{Oth} \cap B^o_{Oth} & A^\varepsilon_{Oth} \cap B^\varepsilon_{Oth} & A^\varepsilon_{Oth} \cap B^-_{Oth} \\ A^-_{Oth} \cap B^o_{Oth} & A^-_{Oth} \cap B^\varepsilon_{Oth} & A^-_{Oth} \cap B^-_{Oth} \end{pmatrix} \tag{3.2}$$

Based on the properties of orthographic projections, there exists an important link between BBRs (i.e., projection rectangles) and TBOs (i.e., AABBs). That is, two projection rectangles enable the reconstruction of a unique AABB. With regard to topological relations, it is intuitive that, the two AABBs A and B are "disjoint" if and only if their projection rectangles are also "disjoint" under two orthographic projections. This leads to the following theorem that identifies the number of topological relations between two TBOs.

Theorem 1. *The number of topological relations between TBOs described by the I_{TBO} matrix equals the number of topological relations between BBRs described by the I_{BBR} matrix.*

Proof. Taking an arbitrary point $p \in A^-$, we have $p \in B^-$ iff $p \in B^-_{Oth}$ in two orthographic projections (easily given by the characteristic of orthographic projection). Thus, $A^- \cap B^- \neq \emptyset$ (sharing a common point) iff $A^-_{Oth} \cap B^-_{Oth} \neq \emptyset$ in two orthographic projections. If we perform the same operation with the other elements in the $I_{TBO}(A, B)$ matrix, it is obvious that each $I_{TBO}(A, B)$ matrix corresponds exactly to an $I_{BBR}(A, B)$ matrix. In other words, the number of topological relations between TBO equals the number of topological relations between BBR. □

Let us now consider topological relations between BBRs. In the previous uncertain topological models, the drawback is that the topological relations are not symmetric, i.e., "A meets B" does not imply "B meets A". To us, the thickness ε is identical to every TBO. It is chosen by application designers as a global parameter depending on the accuracy of interface devices. The symmetry of the relations is consequently conserved. This condition leads to the following restrictions with regard to BBRs.

$$\left.\begin{array}{l} A^-_{Oth} \cap B^\varepsilon_{Oth} \neq \emptyset \\ A^\varepsilon_{Oth} \cap B^\varepsilon_{Oth} \neq \emptyset \\ A^o_{Oth} \cap B^\varepsilon_{Oth} \neq \emptyset \end{array}\right\} \Rightarrow \left\{\begin{array}{l} B^-_{Oth} \cap B^\varepsilon_{Oth} \neq \emptyset \\ B^o_{Oth} \cap B^\varepsilon_{Oth} \neq \emptyset \end{array}\right. \tag{3.3}$$

$$\left.\begin{array}{l} A^-_{Oth} \cap B^\varepsilon_{Oth} = \emptyset \\ A^\varepsilon_{Oth} \cap B^-_{Oth} = \emptyset \end{array}\right\} \Leftrightarrow \left\{\begin{array}{l} A^\varepsilon_{Oth} \cap B^o_{Oth} = \emptyset \\ A^o_{Oth} \cap B^\varepsilon_{Oth} = \emptyset \end{array}\right. \tag{3.4}$$

Note that these restrictions are symmetric with A_{Oth} and B_{Oth}. Compared to 44 uncertain relations in the "approximate topological" model (similarly, the "egg-yolk" model), these restrictions exclude 27 cases, keeping 17. Figure 3.11 illustrates 17 possible topological relations between TBOs. Figure 3.12 illustrates 27 topological relations that are impossible, and thus excluded, between

TBOs with identical thickness. Figure 3.13 shows the transition between the 17 possible topological configurations. We however still keep the same numbering and the name of relations as defined in the previous models for compatibility and comparison with the previous work. As the intersection matrix between the TBO is similar to the BBR, it makes calculating topological relations between TBO in VEs a trivial task.

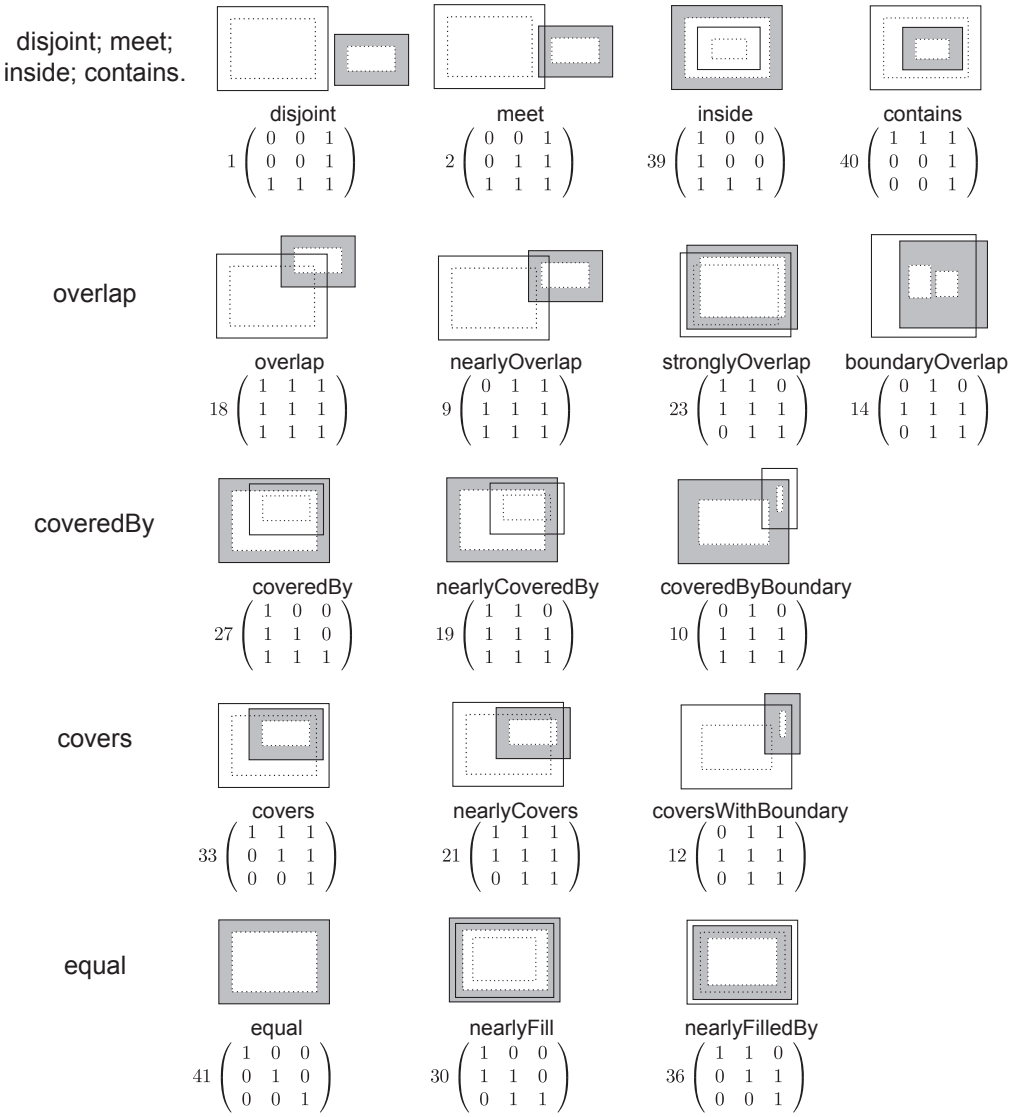


Figure 3.11: Illustration in 2D of 17 topological relations between thick boundary objects.

Some further remarks can be made regarding the TBO. For simplicity, we have defined above the thick surfaces as a buffer zone inside a spatial object. They however may be outside, or both inside/outside the object. In particular, as spatial entities are often large, it seems relevant to suppose that the interior is always bigger than the thickness ε. This restriction eliminates all the configura-

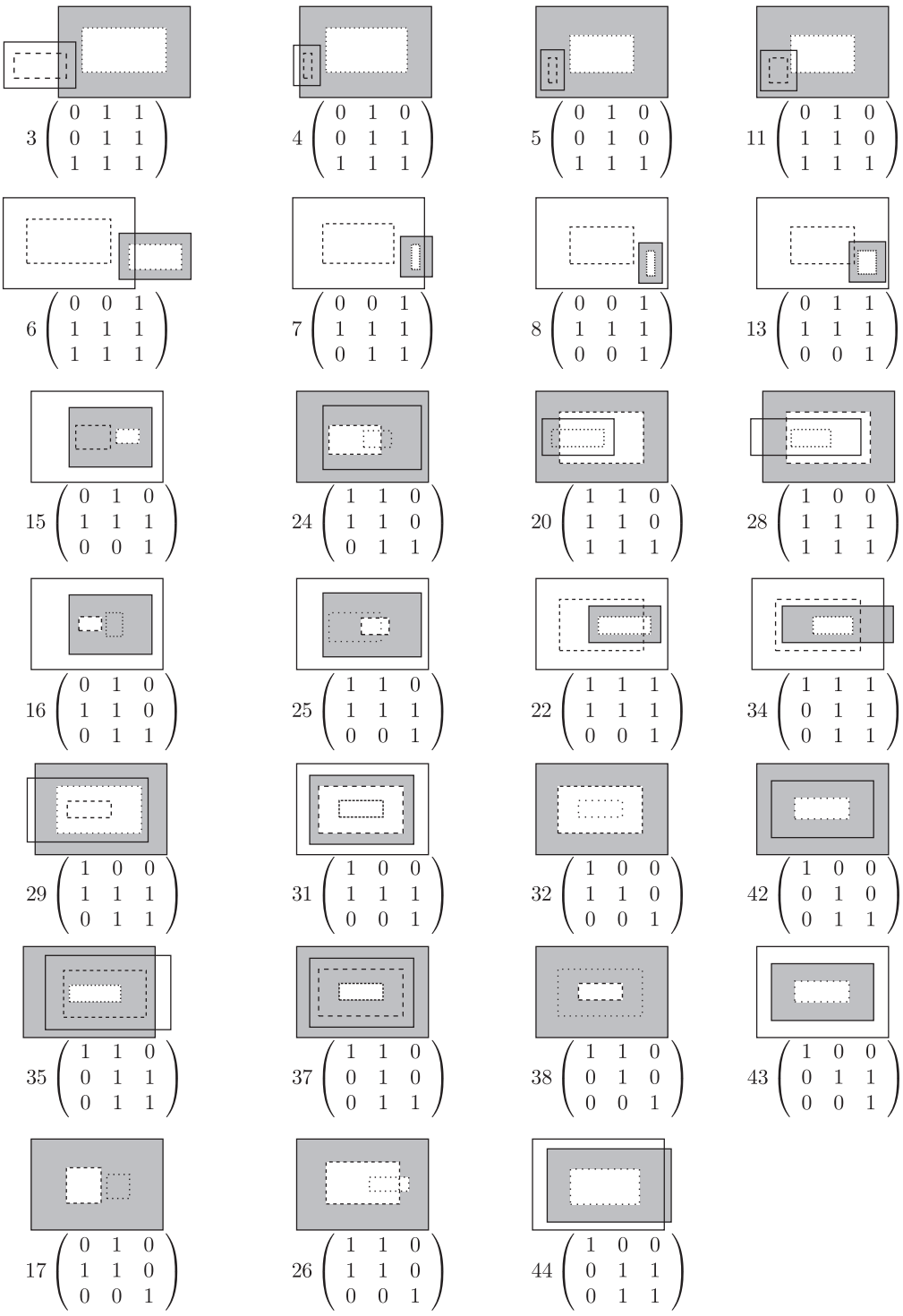


Figure 3.12: 27 topological relations that are excluded between thick boundary objects with identical thickness.

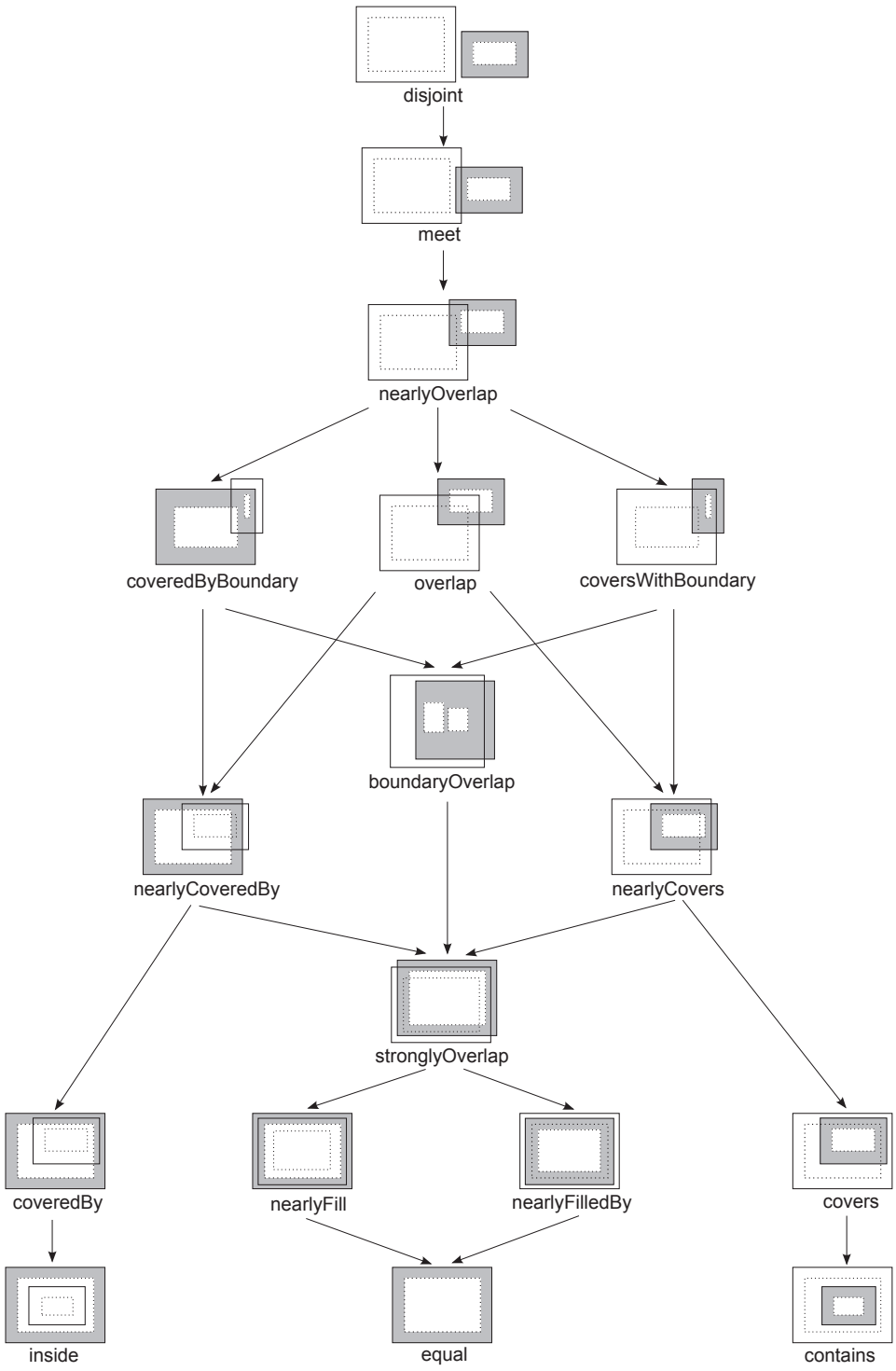


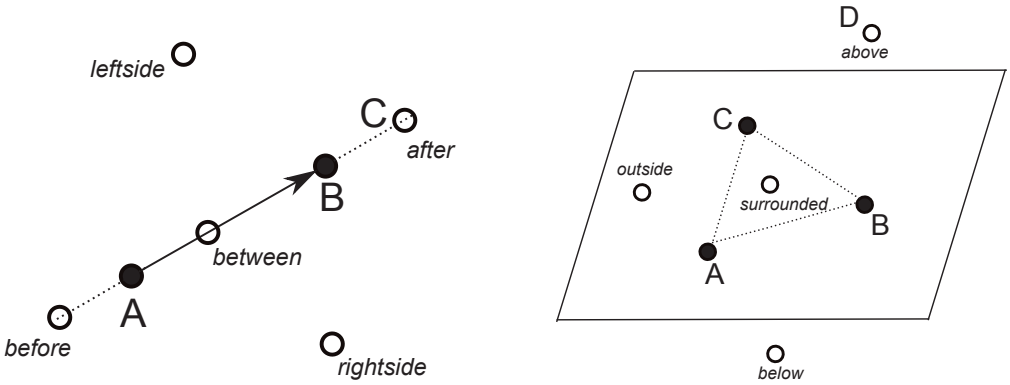
Figure 3.13: The transitions between 17 possible topological configurations between TBO.

tions between two TBOs A and B in which A^o is inside B^ϵ (relations numbered 14, 10, and 12), keeping 14 relations.

To sum up, there are 17 possible binary topological constraints between a primary spatial entity and a reference spatial entity, taking into account their thick boundaries. In Figure 3.11, we gather them in 8 basic groups, corresponding to the original *RCC* and *9-intersection* models:"disjoint", "meet", "overlap", "inside", "coveredBy", "contains", "covers", "equal". For each group, there are several possible spatial configurations. As we will see later, every group of topological constraints corresponds to a topological operator in our spatial language.

3.5 Modelling Projective Constraints

In this section, we aim at modelling projective constraints. We are mainly inspired by projective invariants introduced by Billen and Clementini (Billen and Clementini, 2006; Clementini and Billen, 2006) that are based on the two main concepts: *collinearity* and *coplanarity*.



(a) Ternary projective constraints based on the concept of collinearity.

(b) Quaternary projective constraints based on the concept of coplanarity.

Figure 3.14: The principle of projective constraints, according to (Billen and Clementini, 2006). Spatial entities are simplified as points.

- Collinearity: Initially, projective constraints are formalised based on the *collinearity* among three points that is preserved under any projection. That is, the three collinear points in a 3D space remain collinear after a projective transformation. Based on the concept of collinearity, it is possible to specify spatial relationships such as "between", "leftside", or "rightside", as illustrated in Figure 3.14a. In this case, projective constraints are *ternary*, because there needs three spatial entities to formalise a relationship. Our model of ternary projective constraints is presented in Section 3.5.1.

- Coplanarity: Furthermore, projective constraints can be formalised based on the concept of *coplanarity* among four points. Given the three points

that form a plane, a fourth point can be "coplanar", "above", or "below" with regard to the plane, as showed in Figure 3.14b. In this case, projective constraints are *quaternary*. Section 3.5.2 describes our model of quaternary projective constraints.

3.5.1 Ternary Projective Constraints

To facilitate the presentation of ternary projective constraints, we observe them under two modes: projective views and immersive views.

- Projective views: Figure 3.15 provides a top-view of the room that results from an orthographic projection of the 3D scene to a plane. Given a VE, there are three projective views that can be obtained by projective transformations. They are top-view, bottom-view, and side-view. Projective views are a natural way to observe projective constraints. They are specially useful to give to the user a global view of the environment (e.g, a navigation map of a large-scale environment). Constraint 14 depicts an example of ternary projective constraints observed from the top-view of the VE.

  Constraint 14. *From the top-view of the room, the triangular green table*

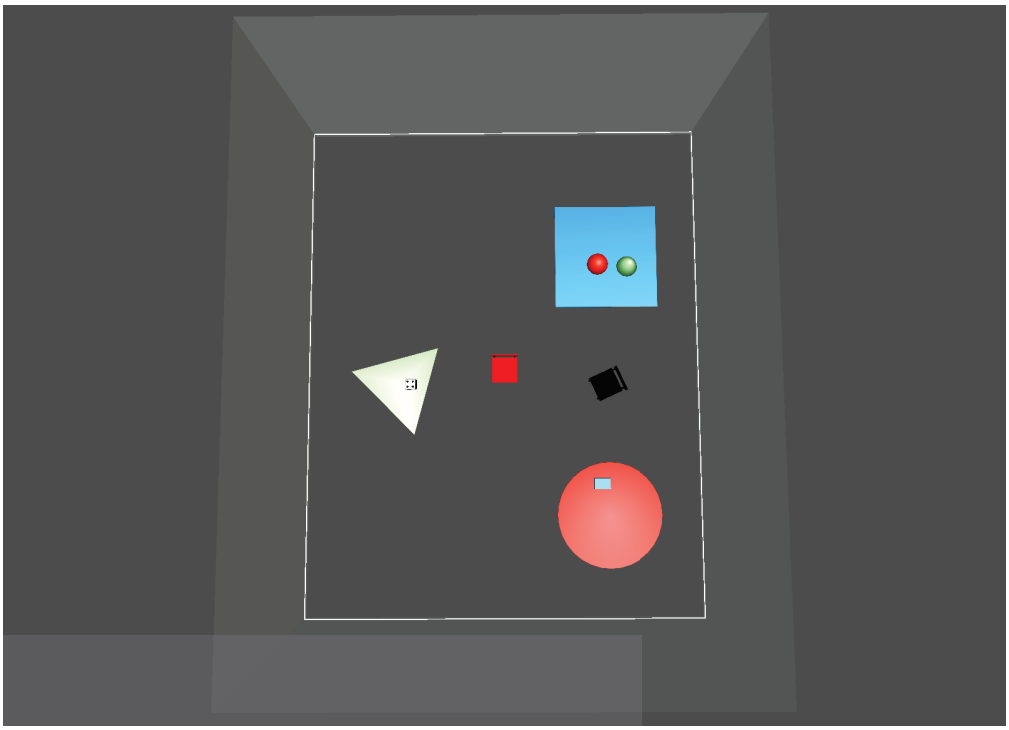


Figure 3.15: A top-view of the room. In this figure, an example of spatial relationships is that "from the top-view of the room, the triangular green table is on the left side of the rounded red table and the rectangular blue table".

must be on the left side of the rounded red table and the rectangular blue table.

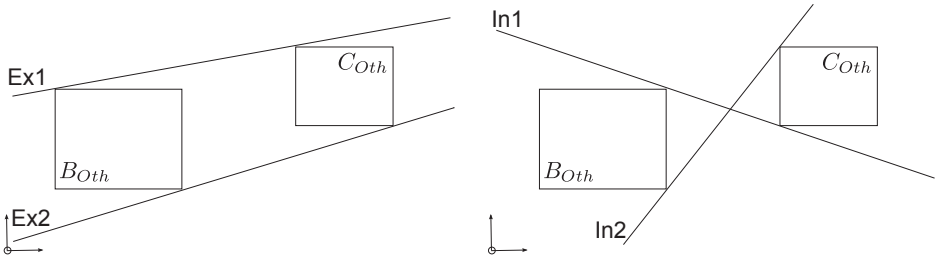
- Immersive views: Otherwise, Constraint 4 depicted earlier another example of ternary projective constraints. That is, "in the room, the black chair must be between the rounded red table and the rectangular blue table". This spatial constraint is called ternary projective constraint, but does not result from a projection. We term these constraints as ternary projective constraints under immersive views, to differentiate from ternary projective constraints under orthographic views.

Orthographic View Mode

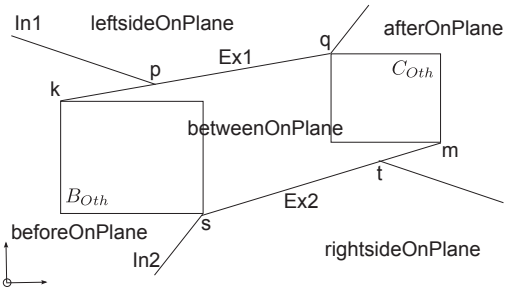
Under an orthographic view, spatial entities are 2D regions, not 3D bodies. To specify spatial constraints under orthographic views, we propose to refine and integrate into VEs the *5-intersection* model whose aim is to describe the collinearity among 2D objects (Clementini and Billen, 2006). In our model, spatial entities are defined either as referential points or bounding boxes. Accordingly, our goal is to specify spatial constraints at the two level of details.

- In the case that spatial entities are simplified as their referential points, ternary projective constraints are described based on the concept of collinearity among three points, as seen in Figure 3.14a.
- Otherwise, given three spatial objects represented by their bounding boxes, an orthographic view (i.e., top view, front view, or side view) of a 3D scene results in three rectangles respectively noted as A_{Oth} (primary object), and B_{Oth}, C_{Oth} (reference objects). In the case that the reference objects are disjoint, based on the *5-intersection* model, the relative position of A_{Oth} to B_{Oth} and C_{Oth} is described thanks to the intersections between the external tangent lines (see Figure 3.16a) and the internal tangent lines (see Figure 3.16b) which constitute five areas (see Figure 3.16c). The central area is called `betweenOnPlane` corresponding to the convex hull of B_{Oth} and C_{Oth}. Based on this area, four other areas are defined that allow one to decide whether the primary object is left/right of or before/after the reference objects.

Nevertheless, it is important to note that there may exist ambiguities in ternary projective constraints under orthographic views. That is, different orthographic views may yield different observations of a concrete relation. In Constraint 14, the relative position of the triangular green table to the rounded red table and the rectangular blue table is inverted (i.e., rightside instead of leftside) under the opposite view (the bottom-view in this case). To disambiguate ternary orthographic projective constraints, we propose that such spatial relationships must be formalised in the context of a *frame of reference*. In our model, frames of reference are different viewpoints (i.e., front view, top view, and side view).



(a) External tangent lines. (b) Internal tangent lines.



(c) Five acceptance areas.

Figure 3.16: Ternary projective relations in an orthographic view.

Let us now consider the computation of ternary projective constraints. The previous algorithms for computing ternary projective relations between regions did not allow to lift ambiguities occurred under different orthographic views. Moreover, since the results of the orthographic projection of AABB are rectangles instead of regions, we decided to build a customised algorithm, which is also used later to compute ternary projective relations in an immersive view. The main steps of our method is illustrated in Algorithm 1.

Algorithm 1: Computing a ternary projective constraint under an orthographic view.

1. Find the internal tangent lines (noted as $In1$, $In2$) and the external tangent lines (noted as $Ex1$, $Ex2$) of B_{Oth} and C_{Oth}.
2. Divide the projection plane into five acceptance areas.
3. Based on the position of the observer, decide whether A_{Oth} (or one of its semantic points) intersects with a particular area from the viewpoint of the observer.

Algorithm 1 is intuitively and geometrically explained as follows.

- The Step 1 is carried out based on an observation that the external tangent lines connect two vertices of B_{Oth} and C_{Oth} such that all the remaining vertices are in the same half-plane. The internal tangent lines are those that connect two vertices of B_{Oth} and C_{Oth} such that all the remaining vertices B_{Oth} and C_{Oth} are in different half-planes.

- In Step 2, it is essential to define the convex hull of B_{Oth} and C_{Oth}. We

denote p, q, s, t as the intersection points between the external tangent lines $Ex1$, $Ex2$ and the internal tangent lines $In1$, $In2$; k and m are the intersection points of the external tangent lines with B_{Oth} and C_{Oth}. It is obvious that the convex hull of B_{Oth} and C_{Oth} is the union of B_{Oth}, C_{Oth}, and the quadrangle made by k, q, m, s.

- In Step 3, it is needed to compute whether A_{Oth} intersects with one of the five acceptance areas. This leads to the problem to decide whether a point of A_{Oth} belongs to a specific acceptance area. Not lost generality, suppose that the situation in Figure 3.16 is obtained from a top view projection. In this case, our algorithms for computing the five relations "between", "leftside", "rightside", "before", and "after" are respectively detailed in Algorithm 2, Algorithm 3, Algorithm 4, Algorithm 5, and Algorithm 6.

Algorithm 2: Point v belongs to the area `betweenOnPlane` if one of the following conditions holds:

(i) v is contained in B_{Oth} or C_{Oth}.
(ii) v is contained in the quadrangle made by k, q, m, s.

Algorithm 3: Point v belongs to the area `leftsideOnPlane` if all the following conditions hold:

(i) v does not belong to the area `betweenOnPlane`. This test is carried out thanks to Algorithm 2.
(ii) v and s are in different half-planes divided by $In1$.
(iii) v and t are in different half-planes divided by $In2$.
(iv) v and s (resp. t) are in different half-planes divided by $Ex1$.

Algorithm 4: Point v belongs to the area `rightsideOnPlane` if all the following conditions hold:

(i) v does not belong to the area `betweenOnPlane`. This test is carried out thanks to Algorithm 2.
(ii) v and q are in different half-planes divided by $In1$.
(iii) v and p are in different half-planes divided by $In2$.
(iv) v and q (resp. p) are in different half-planes divided by $Ex2$.

Algorithm 5: Point v belongs to the area `beforeOnPlane` if all the following conditions hold:

(i) v does not belong to the area `betweenOnPlane`. This test is carried out thanks to Algorithm 2.
(ii) v and q are in different half-planes divided by $In1$.
(iii) v and t are in different half-planes divided by $In2$.

Algorithm 6: Point v belongs to the area `afterOnPlane` if all the following conditions hold:

(i) v does not belong to the area `betweenOnPlane`. This test is carried out thanks to Algorithm 2.
(ii) v and s are in different half-planes divided by $In1$.
(iii) v and p are in different half-planes divided by $In2$.

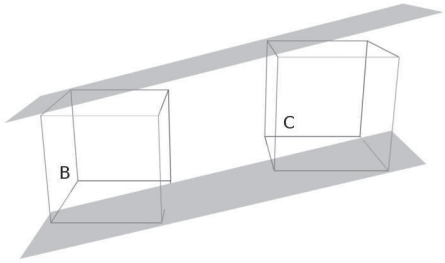
The above algorithms were given from a top-view. To overcome ambiguities, in the case that the observer yields a bottom-view of the environment, the results of Algorithm 3 and Algorithm 4 are inverted. That is, the point v that belongs to the area `leftsideOnPlane` from a top-view is in the `rightsideOnPlane` area from a bottom-view, and vice versa. This enables a unambiguous distinction of ternary projective relations observed under orthographic views. Our algorithms are in constant time, compared to that mentioned in (Clementini and Billen, 2006) which runs in $O(n\log\, n)$ where n is the number of vectors to represent 2D regions.

Immersive View Mode

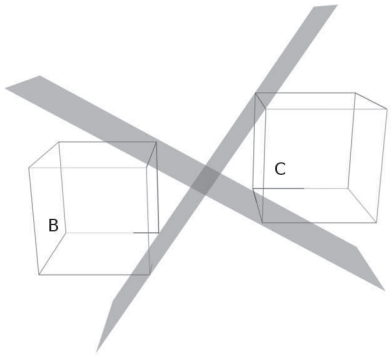
In an immersive view, spatial entities are 3D shapes, not 2D regions. To represent ternary projective constraints between 3D shapes, we are based on the projective model between 3D bodies proposed in (Billen and Clementini, 2006). Given two disjoint reference objects B and C, the localisation of the primary object A can be retrieved from the intersections between the external tangent planes (see Figure 3.17a) and the internal tangent planes (see Figure 3.17b).

However, the initial Billen and Clementini's model remains very formal. The number of tangent planes are undefined. In our context, the use of bounding boxes as convex approximations of spatial entities allows us to argue that there are exactly four external tangent planes and four internal tangent planes of reference objects. Based on the intersections between the tangent planes, there are four acceptance sub-spaces, they are "between", "before", "after", and "aside". Figure 3.17c shows the four acceptance sub-spaces. In an immersive view, ternary projective constraints share three common relationships with those under an orthographic view, they are "between", "before", and "after". However, it is impossible to identify whether the primary entity is leftside/rightside of the two reference entities.

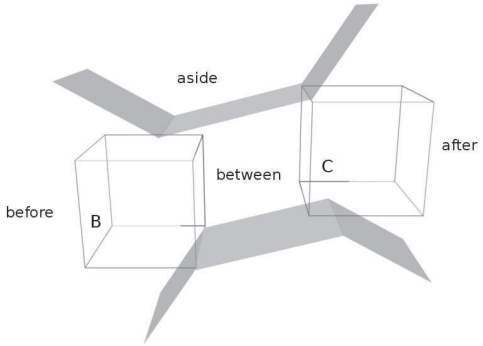
Let us now detail our method to compute ternary projective relations in 3D spaces that is presently lacking in the previous models. The most interesting aspect is that the method is based on the reconstruction of the 3D view from orthographic views, as previously described in Theorem 1. For instance, one can note that the primary object A is "between" the reference objects B and C in an immersive view if and only if the image rectangles of A are between the corresponding ones of B and C in two orthographic projections. It leads to the following algorithm for calculating ternary projective relations between



(a) Two of four external tangent planes.



(b) Two of four internal tangent planes.



(c) Four sub-spaces made up from the tangent planes.

Figure 3.17: Ternary projective relations in an immersive view.

volumetric objects.

Algorithm 7: The primary object A is between the reference objects B and C if one of the following conditions holds:

(i) A_{Oth} is `betweenOnPlane` B_{Oth} and C_{Oth} in the two planes XY and XZ.
(ii) A_{Oth} is `betweenOnPlane` B_{Oth} and C_{Oth} in the two planes XY and YZ.
(iii) A_{Oth} is `betweenOnPlane` B_{Oth} and C_{Oth} in the two planes YZ and XZ.

This algorithm is similarly expanded to other sub-spaces. Overall, the algorithms to compute ternary projective constraints in immersive views are in constant time, because they reuse the previous algorithms to compute ternary projective constraints in orthographic views that were also in constant time.

It is also important to note that, all our algorithms in this chapter to compute spatial constraints will be used in our framework for conceptualising and visualising abstract spatial constraints. A detailed description of the framework

will be given in the next chapter. For instance, based on our model of ternary projective constraints in an immersive view, it is possible to compute and visualise semantic areas formed by three spatial entities in 3D spaces. Figure 3.18 shows the area "between" of the rounded red table and the rectangular blue table. One can see that the black chair is "between" the rounded red table and the rectangular blue table. Constraint 4 is thus satisfied.

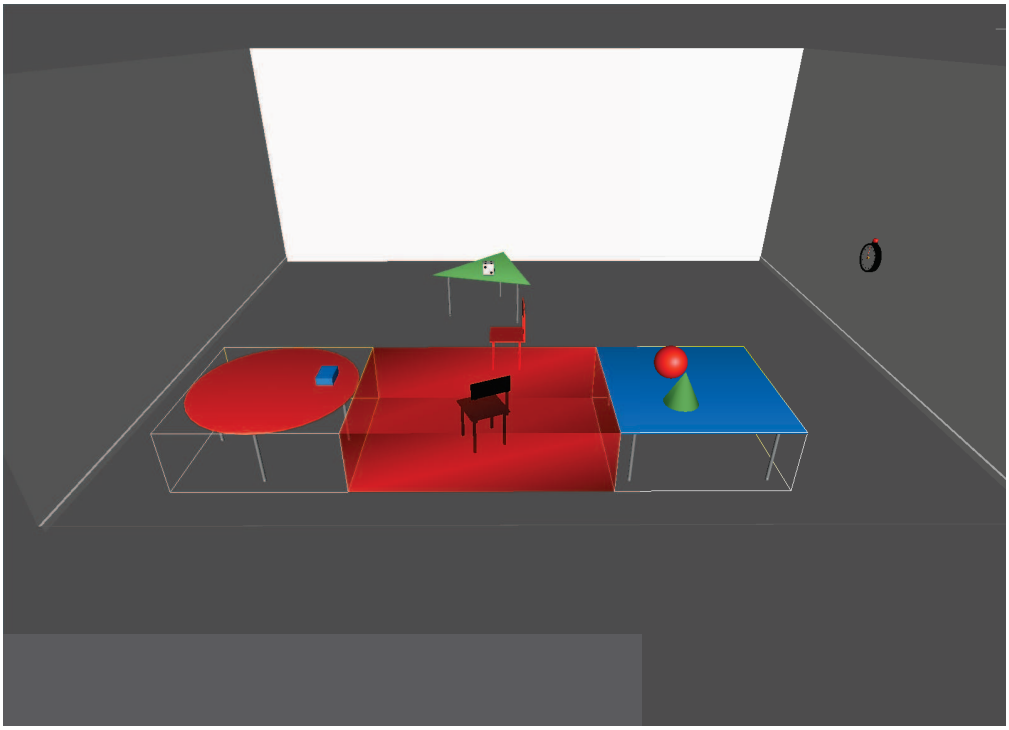


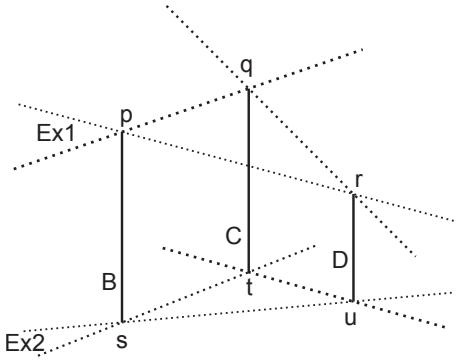
Figure 3.18: The area "between" the rounded red table and the rectangular blue table.

3.5.2 Quaternary Projective Constraints

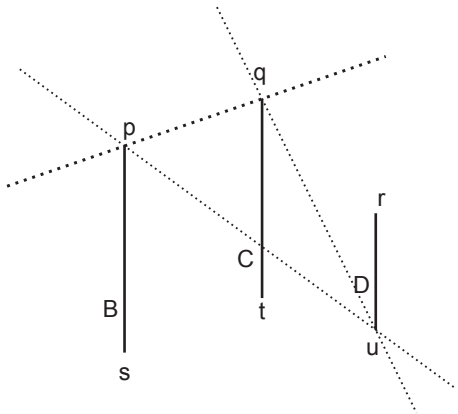
While ternary projective relations are based on the collinearity between three spatial entities, quaternary projective relations deal with the coplanarity among four spatial entities.

In the case that spatial entities are considered as their referential points, the relative position of a fourth point can be defined with regard to the three given reference points, as illustrated in Figure 3.14b. Based on the concept of coplanarity among three points, the possible relationships are "above", "below", "inside" (or "sourrounded"), and "outside".

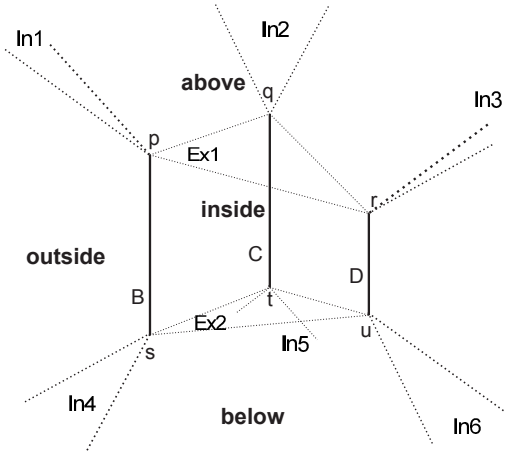
Furthermore, the concept of coplanarity was extended to volumetric objects in a 3D space (Billen and Clementini, 2006). However, no method has been given that allows to compute quaternary projective constraints. Consequently, to specify and compute quaternary projective constraints in VEs, we propose



(a) Two external tangent planes.



(b) One (of six) internal tangent planes.



(c) Four sub-spaces made up from tangent planes.

Figure 3.19: Projective relations between four objects.

a specialised quaternary projective model, taking into account our conceptual definition of spatial entities described previously.

Similar to ternary projective constraints, quaternary projective constraints are geometrically formalised by means of tangent planes. Given three disjoint reference objects B, C, and D, the intersections between the external tangent planes (see Figure 3.19a) and the internal tangent planes (see Figure 3.19b) result in four sub-spaces named *above*, *below*, *inside*, *outside* (see Figure 3.19c, in which full bounding boxes are not showed).

Let us now consider the method to compute quaternary projection relations.

Algorithm 8 illustrates the main steps of our method.

Algorithm 8: Computing a quaternary projective constraint.

1. Find the internal external tangent planes of B, C, and D.
2. Compute the intersections among tangent planes and divide the 3D space into the four acceptance sub-spaces *above*, *below*, *inside*, *outside*.
3. Decide whether A (or one of its semantic points) intersects with a particular sub-space.

Every step of Algorithm 8 is explained in the following.

- In Step 1, the main requirement consists of finding internal and external tangent planes. In the context of AABB, we argue that there are two external tangent planes (noted as $Ex1$ and $Ex2$) and six internal tangent planes (respectively noted as $In1$,..,$In6$). Figures 3.19a and 3.19b intuitively show that the external tangent planes connect three vertices from the reference objects such that all the remaining vertices are on the same half-space. In contrast, the internal tangent planes connect three vertices from the reference objects but gather two objects in a same half-space whereas the last one is in another half-space.

- In Step 2, in addition to the intersections of tangent planes, it is needed to define the convex hull of the three reference objects. Let p, q, r and s, t, u be the vertices that respectively make up the two external tangent planes $Ex1$ and $Ex2$, Figure 3.19c shows that the convex hull is built from the union of B, C, D, $Ex1$, $Ex2$, and three planes, termed as *complementary tangent planes*. Informally, an example of a *complementary tangent plane* is the one made up from p, q, s, and t that connects vertices from two of three reference objects such that all remaining vertices are in the same half-space.

- In Step 3, let us now consider the algorithm to test whether the primary entity A intersects with one of the four sub-spaces. This leads to the problem of deciding whether a point belongs to a specific sub-space. Our the methods for checking the "above" and "below" relations are detailed respectively in Algorithm 9 and Algorithm 10.

Algorithm 9: Point v is above the reference objects B, C, and D if all of the following conditions hold:

(i) v is not inside the convex hull of three reference objects.
(ii) v and s, u, t are in different half-spaces divided by $Ex1$.
(iii) v and s (respectively t, u) are in different half-spaces made by $In1$ (respectively $In2$, $In3$).
(iv) v and p (respectively q, r) are in different half-spaces made by $In4$ (respectively $In5$, $In6$).

Algorithm 10: Point v is above the reference objects B, C, and D if all of the following conditions hold:

(i) v is not inside the convex hull of three reference objects.
(ii) v and p, q, r are in different half-spaces divided by $Ex2$.
(iii) v and p (respectively q, r) are in different half-spaces made by $In4$ (respectively $In5$, $In6$).
(iv) v and s (respectively t, u) are in different half-spaces made by $In1$ (respectively $In2$, $In3$).

The integration of quaternary projective constraints into VEs allows one to express and query semantic constraints related to the coplanarity of a quadruplet of objects. An example of quaternary projective relations was given in Constraint 5, that is "in the room, is the red chair coplanar with the three tables? If yes, is the red chair surrounded by the three tables?". Here, there are three reference entities that are needed to localise a primary entity. Later on, using our model of quaternary projective constraints, it is possible to visualise abstract spatial constraints in VEs. For instance, Figure 3.20 illustrates the visualisation of the area "above" of the three tables in the room.

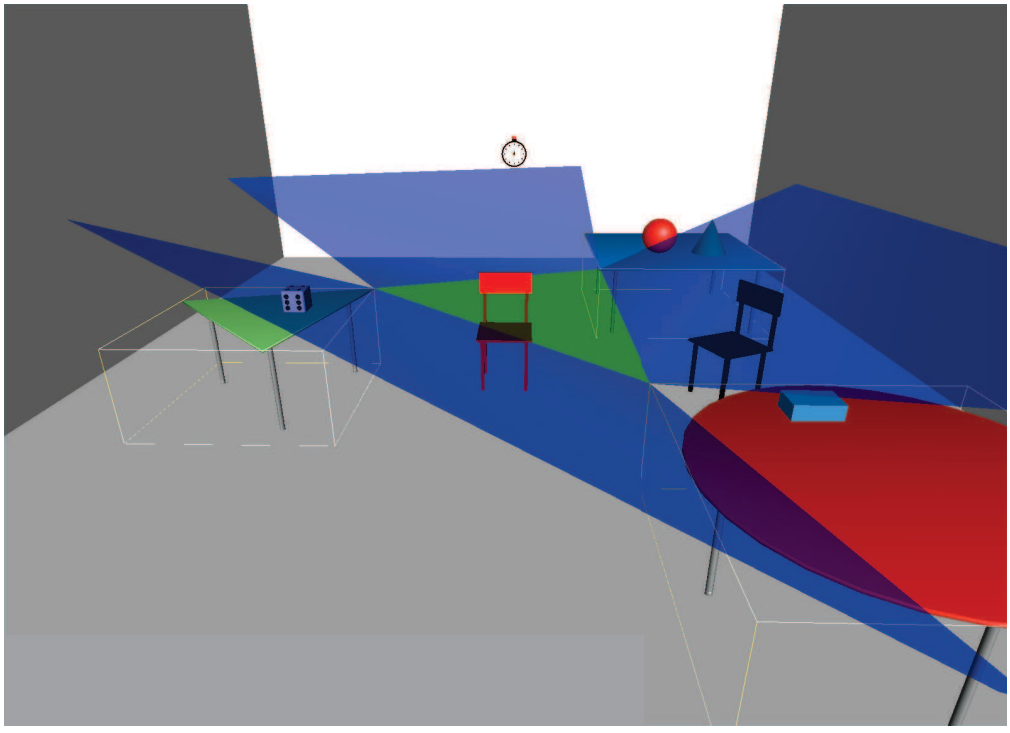


Figure 3.20: The area "above" of the three tables in the room.

3.6 Modelling Directional Constraints

Directional constraints are strongly dependent on FoRs. According to our model of FoR, we model directional constraints using the two main types of FoR: first-person perspective and third-person perspective.

3.6.1 Direction From a First-person Perspective

In this case, direction is based on a reference object, so called object-based or ego-centric direction. According to our model of FoR, to compute directional constraints, there are three possibilities:

1. If the reference object is moving, the motion direction of the reference object is used.
2. If the reference object is still, the intrinsic direction of the reference object is used.
3. Otherwise, if the reference object has not an intrinsic direction, the implicit direction of the environment is used.

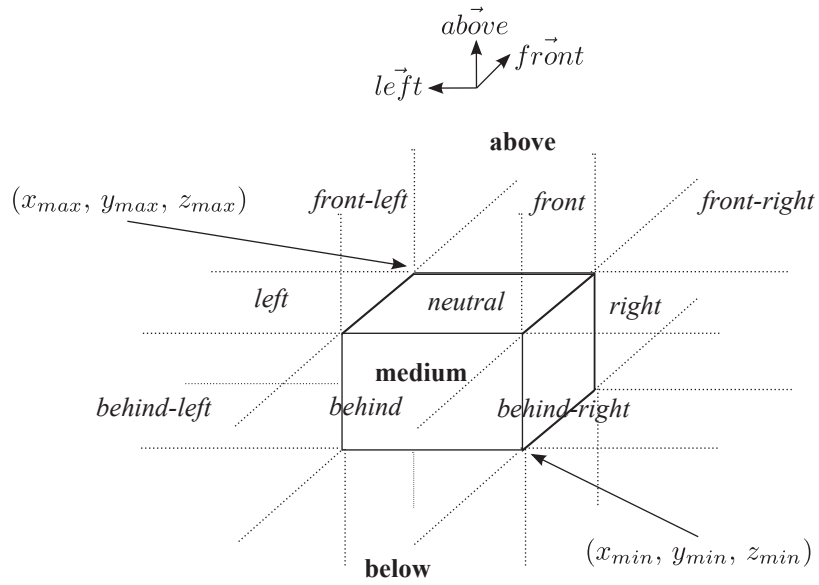
In our model, the motion direction, the intrinsic direction, and the implicit direction of the environment have the same representation (i.e., using the three vector $\vec{front}$, $\vec{left}$, $\vec{above}$). Consequently, not lost generality, we assume that the reference object is still. Thus, directional constraints are computed based on the intrinsic direction of the reference object.

Based on our concept of spatial entity, Figure 3.21 illustrates our model of direction. The model is formalised based on the two levels of representation of spatial entities: bounding box and referential point.

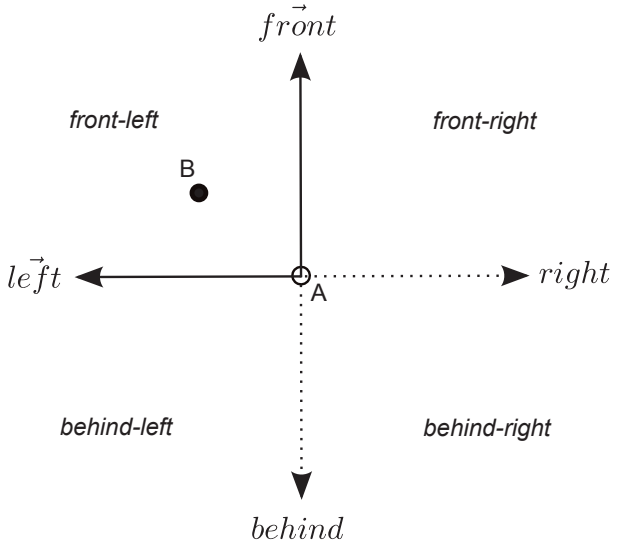
Bounding Box-based Direction

In the case that the reference entity is approximated by its bounding box, we define 27 directional relations divided into 3 layers (above, medium, and below). For each layer, the 9 possible directional relations are "front", "front left", "front right", "left", "right", "behind", "behind left", "behind right", and "neutral", as illustrated in Figure 3.21a. These relations have equivalent terms in geographical direction: "north", "north west", "north east", etc. Our model of object-based direction shares several ideas with the previous models such as the 2D cardinal direction (Frank, 1992) or the Three-dimensional Cardinal Direction (Chen et al., 2007). Nevertheless, based on our conceptual definition of spatial entities and frames of reference, this allows a more formal representation of object-based direction. Given $(x_{min}, y_{min}, z_{min})$ and $(x_{max}, y_{max}, z_{max})$ as two extreme points of the bounding box, a formal definition of the 9 directional relations between a point (x, y, z) and the "medium" layer of the reference entity is as follows:

"front": $x > x_{max}$ and $y_{min} < y < y_{max}$ and $z_{min} < z < z_{max}$.



(a) Direction based on the bounding box of a spatial entity.



(b) Direction based on the referential point of a spatial entity.

Figure 3.21: The proposed model of direction.

"front left": $x > x_{max}$ and $y > y_{max}$ and $z_{min} < z < z_{max}$.
"front right": $x > x_{max}$ and $y < y_{min}$ and $z_{min} < z < z_{max}$.
"left": $x_{min} < x < x_{max}$ and $y > y_{max}$ and $z_{min} < z < z_{max}$.
"right": $x_{min} < x < x_{max}$ and $y < y_{min}$ and $z_{min} < z < z_{max}$.
"behind": $x < x_{min}$ and $y_{min} < y < y_{max}$ and $z_{min} < z < z_{max}$.
"behind left": $x < x_{min}$ and $y > y_{max}$ and $z_{min} < z < z_{max}$.
"behind right": $x < x_{min}$ and $y < y_{min}$ and $z_{min} < z < z_{max}$.

"neutral": $x_{min} < x < x_{max}$ and $y_{min} < y < y_{max}$ and $z_{min} < z < z_{max}$.

These definitions can be easily extended to the "above" and "below" layers. In the case of the "above" layer, the last condition should be changed to $z > z_{max}$. Whereas, in the case of the "below" layer, the last condition should be changed to $z < z_{min}$.

Point-based Direction

In several cases, spatial entities can be considered as points. It is specifically true when spatial entities are virtual humans or users. In these cases, direction is given from their viewpoint (e.g., eyes), represented as a point in a 3D space. Whereas, the body representation of a virtual human or a user is not often taken into account for the computation of directional relationships. Our model of direction based on the referential point of a spatial entity is illustrated in Figure 3.21b. Based on the vertical direction represented by the $\vec{above}$ vector of the reference entity, the space around the entity is divided into three layers: above, medium, and below. For each layer, based on the horizontal direction of the spatial entity represented by the $\vec{front}$ and $\vec{left}$ vectors, the model defines 9 possible directional relationships, they are "front", "front left", "front right", "left", "right", "behind", "behind left", "behind right", and "neutral". It is obvious that the point-based directional model has the same level of granularity as the bounding box-based directional model. This allows a consistent representation of direction, regardless of different levels of detail of representation of spatial entities.

Let us now consider the question how to compute directional constraints between spatial entities simplified as referential points. In fact, there exists several methods enabling such a computation. In our context, we decided to lay our model on the vector-based directional algebra proposed in (Shekhar et al., 1999). The reason is that this algebra is also based on the concept of directed object (i.e., object with internal orientation) that is similar to our concept of spatial entity. To compute the relative position of the primary entity B to the reference entity A, the algebra consists of computing the dot-product between the vector $\vec{AB}$ and the three intrinsic directions of the reference entity A (i.e., $A.\vec{front}$, $A.\vec{left}$, $A.\vec{above}$). First, it is needed to identify whether B belongs to the "above", "medium", or "below" of A. The guaranteed conditions are as follows:

B belongs to the "above" layer of A: $\vec{AB} \odot A.\vec{above} > 0$.

B belongs to the "medium" layer of A: $\vec{AB} \odot A.\vec{above} = 0$.

B belongs to the "below" layer of A: $\vec{AB} \odot A.\vec{above} < 0$.

Next, for each layer of A, it is necessary to identify one of the 9 areas to which B belongs. Considering Figure 3.21b, assume that B belongs to the "medium" layer of A (i.e., $\vec{AB} \odot A.\vec{above} = 0$), the checking conditions for a specific area are the following.

"front": $\vec{AB} \odot A.\vec{front} > 0$ and $\vec{AB} \odot A.\vec{left} = 0$.

"front left": $\vec{AB} \odot A.\vec{front} > 0$ and $\vec{AB} \odot A.\vec{left} >$.

"front right": $\vec{AB} \odot A.\vec{front} > 0$ and $\vec{AB} \odot A.\vec{left} < 0$.
"left": $\vec{AB} \odot A.\vec{front} = 0$ and $\vec{AB} \odot A.\vec{left} > 0$.
"right": $\vec{AB} \odot A.\vec{front} = 0$ and $\vec{AB} \odot A.\vec{left} < 0$.
"behind": $\vec{AB} \odot A.\vec{front} < 0$ and $\vec{AB} \odot A.\vec{left} = 0$.
"behind left": $\vec{AB} \odot A.\vec{front} < 0$ and $\vec{AB} \odot A.\vec{left} > 0$.
"behind right": $\vec{AB} \odot A.\vec{front} < 0$ and $\vec{AB} \odot A.\vec{left} < 0$.
"neutral": $\vec{AB} \odot A.\vec{front} = 0$ and $\vec{AB} \odot A.\vec{left} = 0$.

These checking conditions are similarly applied to the "above" and "below" layers of the reference entity A.

3.6.2 Direction From a Third-person Perspective

In this case, direction is observed from the perspective of a third-person. It is also referred to as viewer-based or alo-/exo-centric direction. Figure 3.22 is a simplified illustration in 2D of the room presented in Figure 3.1. It is used to illustrate several directional constraints from a third-person perspective presented so far. For instance, Constraint 9 showed that "from the point of view of the desk, the rectangular blue table must be on the left of the rounded red table". Constraint 10 indicated that "from the point of view of the desk, the triangular green table must be behind the rectangular blue table".

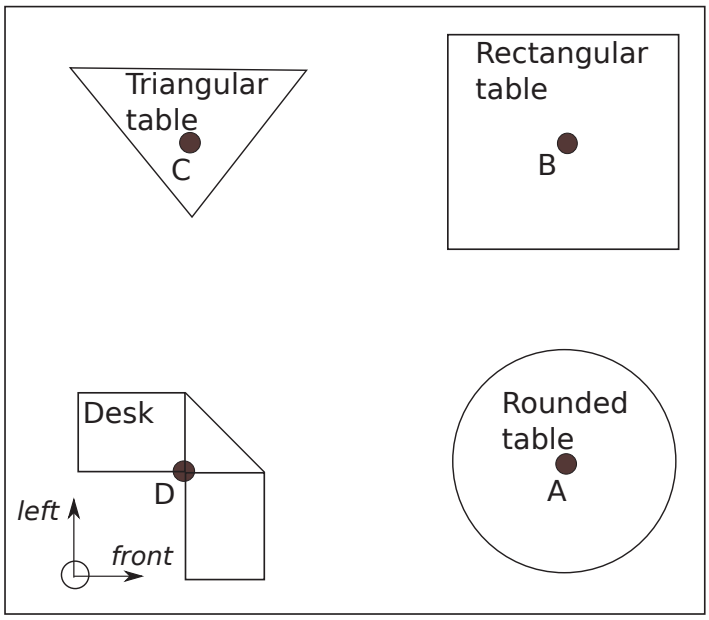


Figure 3.22: Example of direction from a third-person view: "From the viewpoint of the desk (D), the rectangular blue table (B) must be on the left of the rounded red table (A)".

The most important point is that direction given from a third-person perspective is completely based on the intrinsic direction of the viewer. Whereas,

both primary and reference entities may not be oriented. For example, using Figure 3.22, let us consider the directional relation between the rectangular blue table (B) and the rounded red table (A), from the perspective of the desk (D). Here, both of the primary entity B and the reference entity A do not have intrinsic direction. Instead, the viewer D has an intrinsic direction, it is used as the frame of reference to describe the directional relation.

Based on this important remark, this leads us to the ability to specify third person-based direction by using the first person-based model presented in the previous section. Our main idea is to apply the intrinsic direction of the viewer to the reference entity. Then, we use the reference entity, with an already defined intrinsic direction, to specify the direction of the primary entity. Applying our model to Figure 3.22, A has the same intrinsic direction with D. Using our model of direction from a first-person perspective, it is easy to calculate that $\vec{AB} \odot D.\vec{front} = 0$ and $\vec{AB} \odot D.\vec{left} > 0$ and $\vec{AB} \odot D.\vec{above} = 0$. Thus, it is concluded that B is on the left of A from the viewpoint of D, that is a correct result. We have the same result when A, B, and D are approximated by their bounding boxes. Because our model of direction from a third-person perspective reuses the model of direction from a first-person perspective, there are 27 possible directional relationships that can be specified from a third-person perspective.

As a summary, both the two models of direction from a first- and third-person perspective are mainly based on the concept of spatial entity. We enable the modelling of a variety of frame of reference. Our model of direction from a third-person perspective shares the same principle with our model of direction from a first-person perspective. This leads to several advantages. On the one hand, the models allow to specify directional constraints using both the two levels of representation of spatial entities, with the same level of granularity. All of the computation of direction are built upon spatial entities, and thus are simple to perform. On the other hand, they still enable a consistent representation of direction.

3.7 Modelling Distance Constraints

In this section, we aim to model distance constraints in VEs. In addition to direction, distance provides important metric information about space. So far, Constraint 11 described an example of a distance constraint in the room, that is "the rounded red table must be closer than 1 meter to the desk". In fact, distance is often combined with other spatial knowledge, notably directional knowledge. An example of such a combination was given in Constraint 13, that is "the rounded red table must be in front of and closer than 1 meter to the desk".

As we have discussed in Section 2.2.4 of Chapter 2, there are two main approaches to distance models in the literature. The first one is based on absolute distance, whereas the second one uses relative distance. Absolute distance is directly computed between two spatial entities (e.g., "A is 10 meters from B"

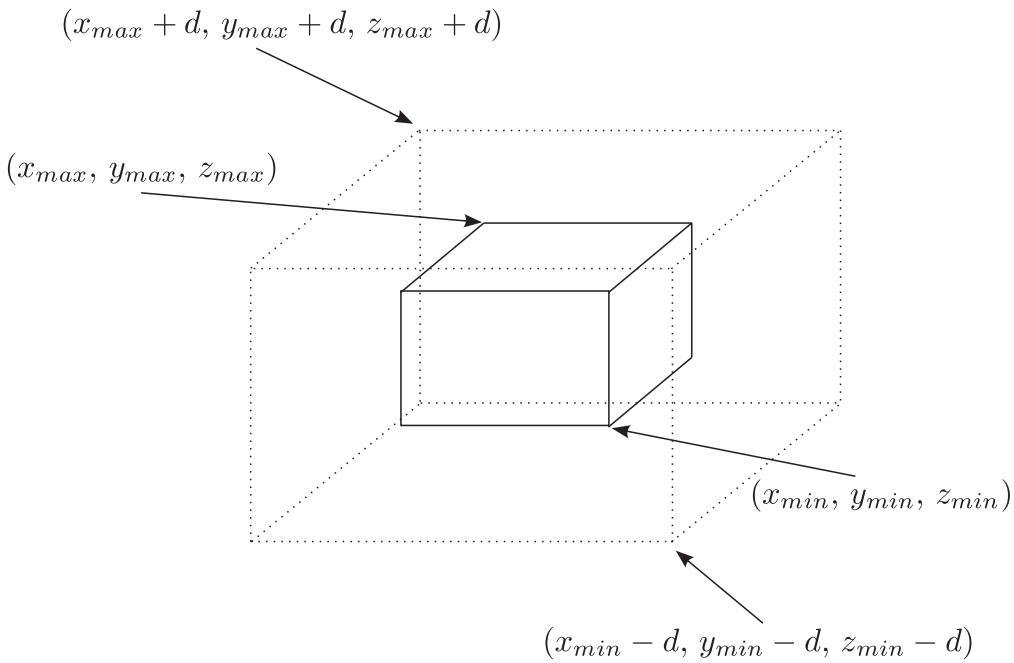


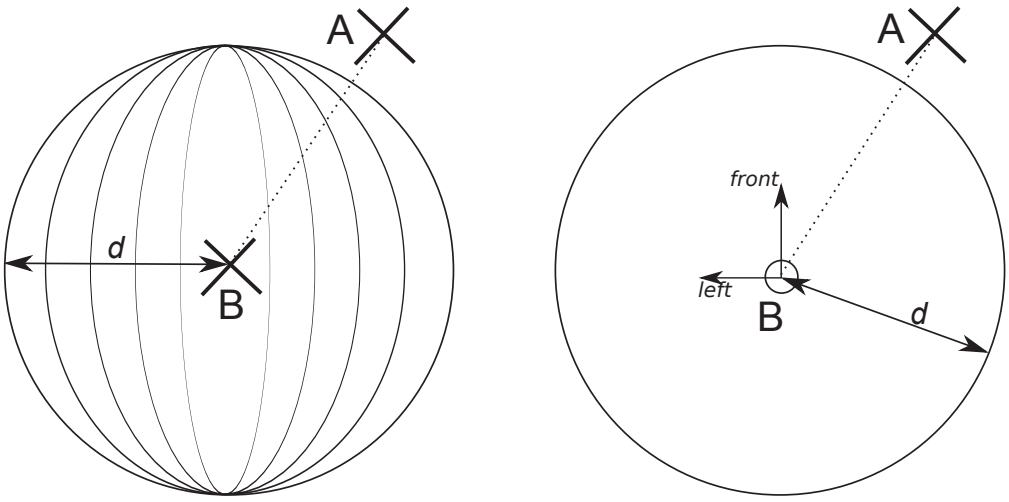
Figure 3.23: The proposed model of distance based on the bounding box of a spatial entity.

or "A is close to B"). Whereas, relative distance is based on a comparison with a reference distance (e.g., "A is closer to B than to C"). Our goal is to support both the two types of distance in VEs.

Absolute Distance

With regard to absolute distance, it can be represented both in a quantitative or qualitative way. Using a quantitative approach, distance between two spatial entities is measured in a metric manner. In contrast, a qualitative approach divides the space around the reference entity into different levels of granularity, for example "close", "medium", and "far". While qualitative distance models are interesting, they are however very dependant to a specific domain. In fact, like directional models, qualitative distance is mostly specified in the context of frame of reference. Frames of reference for distance are strongly associated to a domain. To qualitatively measure the concepts of "close", "medium", and "far", distance is defined based on the number of bus stops, metro stations, travelling time, or travelling cost.

For the above reasons, we propose a semi-qualitative approach to distance in VEs. That is, in our model, absolute distance is modelled in a quantitative manner. Thereafter, the model supports a qualitative representation of relative distance. Our model of absolute distance is illustrated in Figure 3.23 and Figure 3.24, corresponding to the two levels of specification of spatial entities. Our model of relative distance will be discussed later in the end of this section.



(a) Distance from the referential point.

(b) Distance from the referential point, combining with the intrinsic direction of the spatial entity.

Figure 3.24: The proposed model of distance based on the referential point of a spatial entity.

When a spatial entity is represented by its bounding box, the sub-space surrounding the spatial entity with a distance d is illustrated in Figure 3.23. Given $(x_{min}, y_{min}, z_{min})$ and $(x_{max}, y_{max}, z_{max})$ as the two extreme points of the bounding box, a point (x, y, z) has a distance equal or less than d to the spatial entity if and only if $x_{min} - d \leq x \leq x_{max} + d$ and $y_{min} - d \leq y \leq y_{max} + d$ and $z_{min} - d \leq z \leq z_{max} + d$. As a result, when one of the conditions does not hold, this leads to the fact that the point (x, y, z) has a distance further than d to the spatial entity.

Interestingly enough, the formalisation of distance can be extended to take into account directional information. In the previous section, direction was defined by dividing the space around a spatial entity into three layers: above, medium, and below. For each layer, there are nine possible directional constraints "front", "front left", "front right", "left", "right", "behind", "behind left", "behind right", and "neutral". Within the medium layer, each directional relationship is subsequently extended by combining with distance information (e.g., equal or less than d).

"front" with a distance d:

$x_{max} \leq x \leq x_{max} + d$ and $y_{min} \leq y \leq y_{max}$ and $z_{min} \leq z \leq z_{max}$.

"front left" with a distance d:

$x_{max} \leq x \leq x_{max} + d$ and $y_{max} \leq y \leq y_{max} + d$ and $z_{min} \leq z \leq z_{max}$.

"front right" with a distance d:

$x_{max} \leq x \leq x_{max} + d$ and $y_{min} - d \leq y \leq y_{min}$ and $z_{min} \leq z \leq z_{max}$.

"left" with a distance d:

$x_{min} \leq x \leq x_{max}$ and $y_{max} \leq y \leq y_{max} + d$ and $z_{min} \leq z \leq z_{max}$.

"right" with a distance d:

$x_{min} \leq x \leq x_{max}$ and $y_{min} - d \leq y \leq y_{min}$ and $z_{min} \leq z \leq z_{max}$.

"behind" with a distance d:

$x_{min} - d \leq x \leq x_{min}$ and $y_{min} \leq y \leq y_{max}$ and $z_{min} \leq z \leq z_{max}$.

"behind left" with a distance d:

$x_{min} - d \leq x \leq x_{min}$ and $y_{max} \leq y \leq y_{max} + d$ and $z_{min} \leq z \leq z_{max}$.

"behind right" with a distance d:

$x_{min} - dx \leq x_{min}$ and $y_{min} - d \leq y \leq y_{min}$ and $z_{min} \leq z \leq z_{max}$.

"neutral":

$x_{min} \leq x \leq x_{max}$ and $y_{min} \leq y \leq y_{max}$ and $z_{min} \leq z \leq z_{max}$.

In the case of the "upper" layer (respectively the "below" layer), the last condition is changed to $z_{max} \leq z \leq z_{max} + d$ (respectively $z_{min} - d \leq z \leq z_{min}$ for the "below" layer). The above formalisations serve as the basis for visualising spatial constraints in VEs. Figure 3.25 illustrates the combination between distance and directional constraints. Given a spatial entity like a desk, the semantic areas "front of" with a distance d (e.g., 1.5 meter) to the desk are visualised.

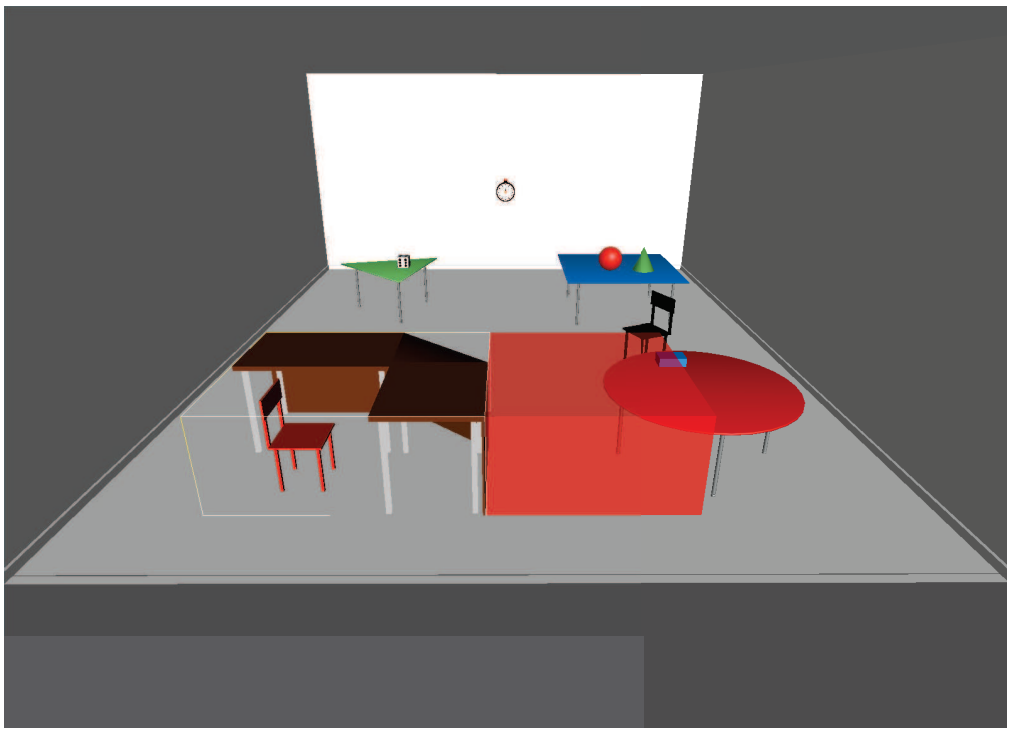


Figure 3.25: The area front of and 1.5 meters from the desk.

Let us now consider the case that spatial entities are represented by their referential points. Our model of distance is illustrated in Figure 3.24a. Given the reference entity B represented by its referential point (B_x, B_y, B_z), a point

(x, y, z) has a distance equal or less than d to B if and only if:

$$\sqrt{(B_x - x)^2 + (B_y - y)^2 + (B_z - z)^2} \leq d$$

Moreover, this point-based distance model can be combined with the directional model. In the previous section, we have seen that point-based direction is computed using a vector-based directional algebra. Considering Figure 3.24b, a distance and directional constraint such as "B is front-right of A, and B has a distance further than d to A" can be formalised by combining the conditions of both the distance and directional models, explained as follows:

"B front-right of A":

$\vec{BA} \odot B.\vec{front} > 0$ and $\vec{BA} \odot B.\vec{left} < 0$ and $\vec{BA} \odot B.\vec{above} = 0$

"B has a distance further than d to A":

$$\sqrt{(B_x - A_x)^2 + (B_y - A_y)^2 + (B_z - A_z)^2} > d$$

Relative Distance

Already defined our model of absolute distance, let us now consider relative distance. So far, we have defined relative distance between two spatial entities as a comparison to the distance to a third entity. Figure 3.26 provides an example of relative distance. In this example, B and C have "equal" distances to A. Whereas, the distance between D and A is "greater" than the distance between B and A. In other words, the distance between B and A is "shorter" than the distance between D and A. In general, spatial expressions of relative distance take the forms as follows.

"A is closer to B than to C"
"A is equal to B than to C"
"A is further than B from C"

In our model, relative distance is computed by comparing two absolute distances between two spatial entities. Given Constraint 11 "the rounded red table is closer than 1 meter from the desk" and the fact that "the rectangular blue table has a distance more than 2 meters to the desk", it is possible to conclude that "in the room, the rounded red table is closer to the desk than the the rectangular blue table". In other words, it can be expressed that "the rectangular blue table is further than the rounded red table, from the desk".

3.8 Summary

In this chapter, we have presented a general model of spatial constraints for VEs. An in-depth definition of both nonmetric spatial constraints (i.e., topological and projective constraints) and metric spatial constraints (i.e., directional and distance constraints) was presented. We also detailed the methods that allow to compute spatial constraints in VEs. This general model provides a theoretical basis for our framework to conceptualise spatial constraints in VEs that will be presented in the next chapter.

Regarding topological constraints, to tackle the lack of precision when manipulating 3D objects within VEs, we introduced the concept of *thick boundary*

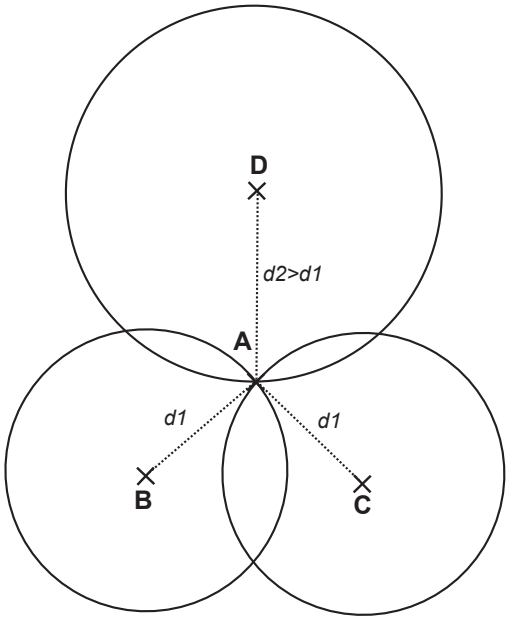


Figure 3.26: Example of relative distance.

objects and then defined 17 topological relations between them. Our topological model allows to specify 8 basic topological relationships between spatial entities, they are "disjoint", "meet", "overlap", "inside", "coveredBy", "contains", "covers", "equal".

Our model of projective constraints allowed one to define spatial constraints among three objects (i.e., "before", "between", "after", "leftside", "rightside") and four objects (i.e., "above", "below", "coplanar"), both in orthographic and immersive views.

Our directional model enables the specification of meaningful relationships such as "left", "right" or "north", "south" in VEs. We proposed a formal model for direction in VEs. The model is based on a qualitative description of direction that has proven to be closer to how human represents the world. We defined a clear semantics of reference systems used for describing direction. Our directional model allows to specify direction both from a first- and third-person perspective.

While our model of directional constraints is qualitative, our approach to distance constraints is semi-qualitative. We modelled two types of distance: absolute distance and relative distance. First, absolute distance is quantitatively computed between two spatial entities. Next, relative distance is obtained by comparing the quantitative distances between two spatial entities to a third spatial entity. Relative distance is purely qualitative. There are three possible constraints: "closer than", "equal to", and "further than".

Through a simple but realistic VE illustrating a room, we showed how each family of spatial constraints can be used to express spatial knowledge to users. Furthermore, we argued that topological constraints, projective constraints, di-

rectional constraints, and distance constraints can be combined together to define complex spatial constraints.

Chapter 4

A Language for Specifying Spatial Semantics

The previous chapter described our formal model of spatial constraints in VEs. We modelled and computed the four main types of spatial constraints (i.e., topological, projective, distance, and directional constraints). In this chapter, we show how the formal model of spatial constraints is applied into a spatial language and framework, named Vrx-Ocl, to specify spatial semantics of VEs at the conceptual level.

This chapter is structured as follows. Section 4.1 positions the Vrx-Ocl language into the overall architecture of the Mascaret framework. The syntax and meta-model of Vrx-Ocl are described in Section 4.2. Sections 4.3, 4.4, 4.5, and 4.6 respectively present how topological, projective, directional, and distance constraints are specified using the Vrx-Ocl language. Finally, Section 4.7 summarises the chapter.

4.1 Positioning

4.1.1 Architecture

To conceptualise semantic spatial constraints in VEs, the Vrx-Ocl approach is composed of two main steps.

1. Building the conceptual model of the VE
2. Using the Vrx-Ocl spatial language to specify spatial constraints in the conceptual model

The overall approach follows our critical remarks of the related solutions for conceptualising spatial relationships, as previously discussed in Section 2.3. We have stated that a conceptual schema, completed with a spatial language, appears to be a promising approach for conceptualising spatial relationships. The first step aims to define the conceptual schema of the VE. In the context of the Vrx-Ocl approach, we use the Mascaret framework to build the conceptual

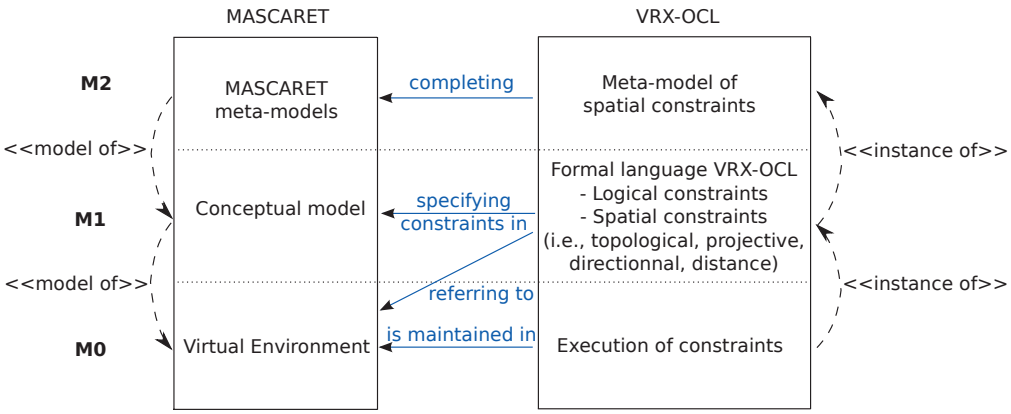


Figure 4.1: Positioning the VRX-OCL approach into the MASCARET approach.

model and the semantic model of the VE. However, the semantic conceptual model merely provides a description of the structure, behaviours, and conceptual links among spatial entities that compose the VE, not semantic spatial relationships among them.

Therefore, in the second step, our aim is to specify spatial constraints among spatial entities. To do so, we propose the VRX-OCL language that allows to specify spatial constraints and other types of constraints at the conceptual model.

Figure 4.1 situates the VRX-OCL approach into the overall architecture of MASCARET. Similar to MASCARET, VRX-OCL uses a model-based approach.

- First, because the modelling of spatial constraint require additional concepts, we had to extend the meta-model of MASCARET (i.e., the M2 layer) to cover the concepts of spatial entity, frame of reference, and spatial constraints.
- Next, the VRX-OCL language is proposed to specify spatial constraints at the conceptual model (i.e., the M1 layer).
- Finally, the execution of constraints (e.g., checking constraints satisfaction or constraints violation) is carried out in a concrete VE (i.e., the M0 layer).

4.1.2 Fitting into the Overall Development Process

Modelling of spatial constraints is fitted into the overall development process of MASCARET. To take into account the phase of modelling of spatial constraints, the development process of MASCARET must be adapted.

In the first phase (i.e., the domain modelling phase), the designer of the VE uses the VRX-OCL language to add spatial constraints to the conceptual model of the VE produced by MASCARET. Initially, to conceptualise a VE, the designer uses MASCARET diagrams such as class diagrams, statechart diagrams, or activity diagrams. For instance, Figure 4.2 shows the MASCARET class diagram representing the structure (i.e., domain concepts and their conceptual links) of a

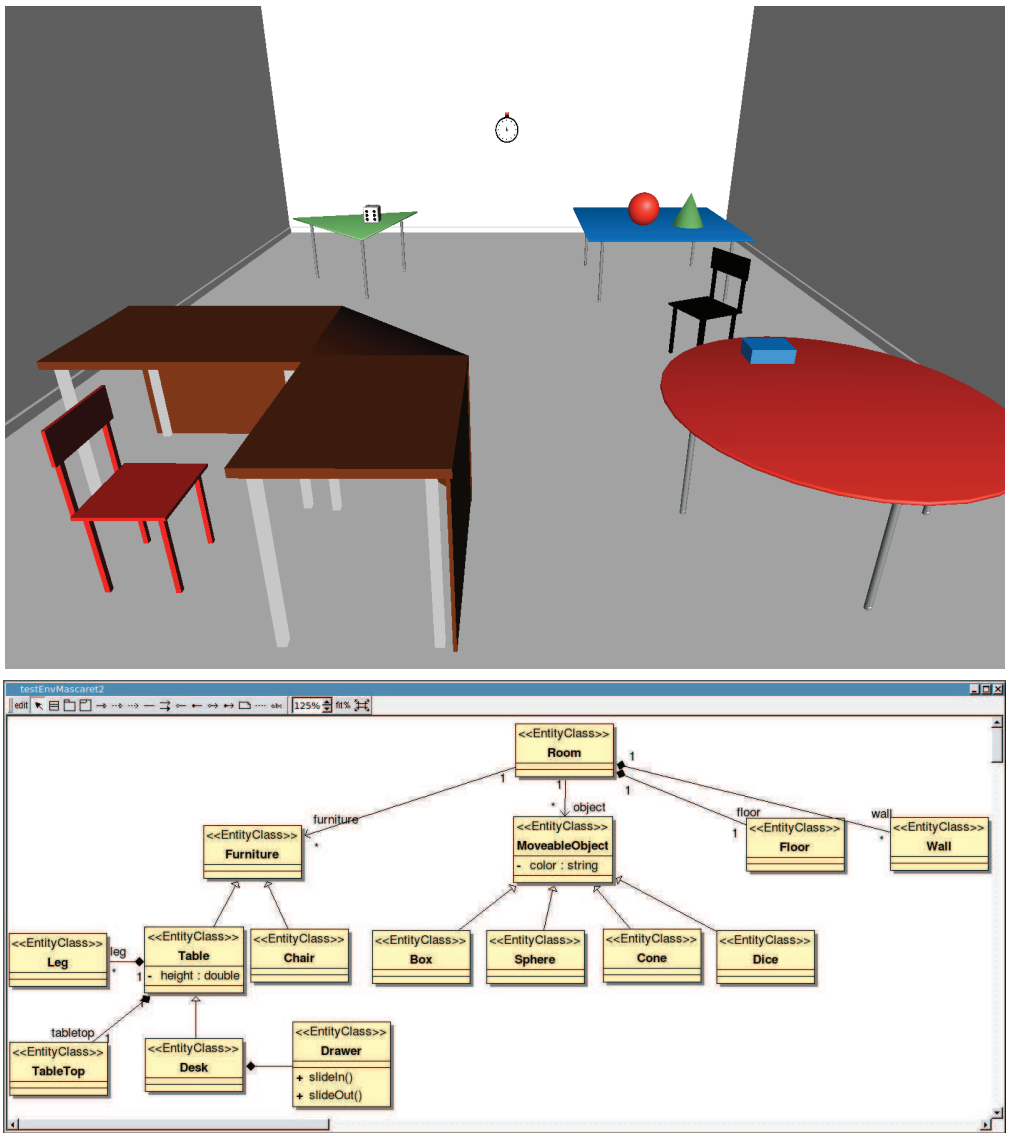


Figure 4.2: Conceptual modelling of virtual environment. Top: graphical representation of a room. Bottom: conceptual model of the room. Here, a MASCARET class diagram is used to represent the structure of the room, without any specification of spatial constraints.

virtual room. However, these conceptual diagrams are unable to convey spatial constraints. Instead, it is done by means of textual expressions in the VRX-OCL language, as illustrated in Figure 4.3. The specification of spatial constraints is performed using UML-supported modellers that allow to define constrains on any element of MASCARET models.

In the second phase (i.e., the instantiating phase), there needs several adaptations taking into account additional spatial information of spatial entities. Figure 4.4 shows an example of such an adaptation. A desk is an oriented spa-

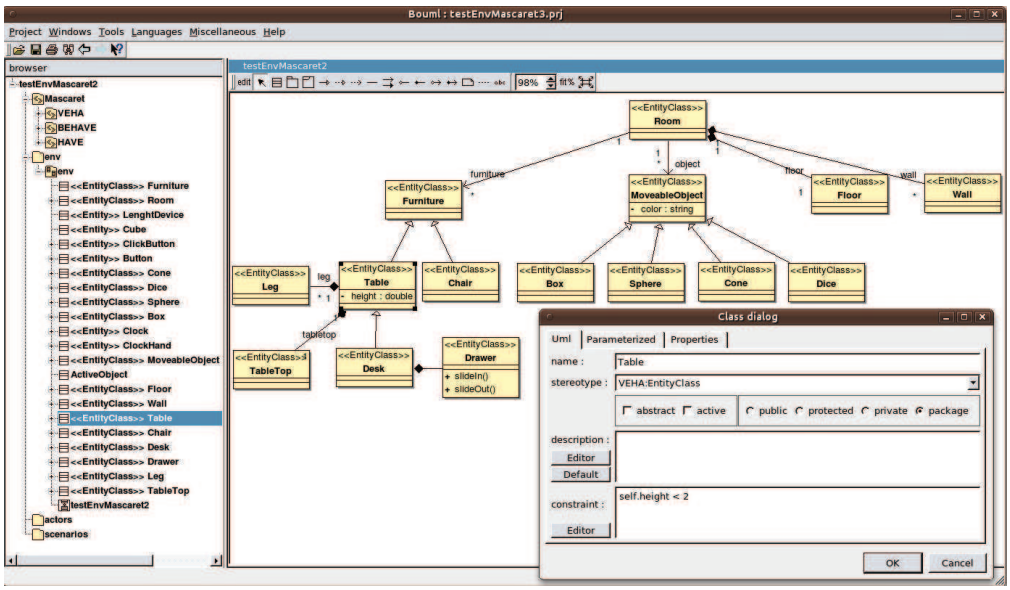


Figure 4.3: Adding the specification of constraints into the conceptual model, by means of textual expressions in the VRX-OCL language.

tial entity. However, it was impossible to specify the intrinsic direction of the desk in UML models. In our approach, the instantiation of spatial entities is enriched with additional information used for modelling spatial constraints, such as intrinsic direction of a spatial entity.

4.2 The VRX-OCL Language

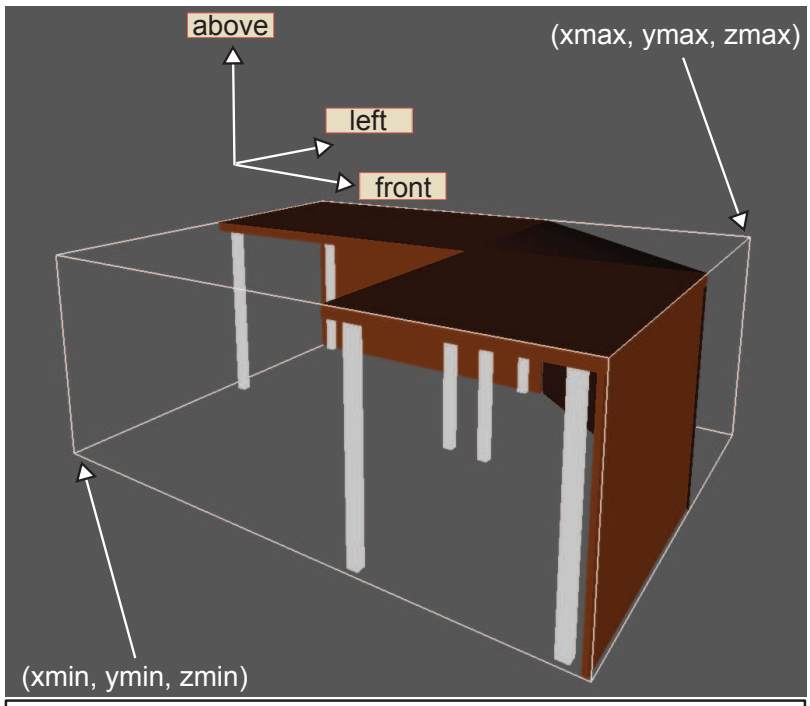
To specify spatial constraints at the conceptual model of VEs, we define the VRX-OCL language. In this section, we first give a general presentation of the language. Then, we present the extended spatial syntax of VRX-OCL. Finally, we describe the meta-model of VRX-OCL that enables the specification of spatial entities and spatial constraints at the meta level.

4.2.1 Presentation

VRX-OCL is defined as a Virtual Reality eXtension of the Object Constraint Language (Warmer and Kleppe, 2003). Initially, OCL is a standardised formal language to specify constraints in a UML-based conceptual model. By extending OCL, VRX-OCL is able to cover both spatial constraints as well as other types of constraint at the conceptual model of a VE.

Our motivations to define VRX-OCL as a spatial extension of OCL are based on the following characteristics of OCL.

- Both query and constraint language: OCL allows to express both constraints and queries at the conceptual model. OCL defines a constraint as a restriction on the values or properties of an object-oriented model. For




```
(1)<Entity class="Desk" name="desk1">
(2)  <Inline url="VRMLS/Desk/desk1.xml"/>
(3)  <Position x="-2.5" y="2" z="0"/>
(4)  <Rotation roll="0" pitch="0" yaw="-1.11529828327"/>
(5)  <IntrinsicDirection>
          < frontVector x="1" y="0" z="0"/>
          < leftVector  x="0" y="1" z="0"/>
          < aboveVector x="0" y="0" z="1"/>
     </IntrinsicDirection>
(6)</Entity>
```

Figure 4.4: Adaptation at the instantiating phase. Top: a desk in the application. Bottom: the instantiation of the desk with additional information used for computing spatial constraints.

example, "the length of the table must be less than 2 metres" is a constraint applied into the concept of table in the conceptual model. Additionally, OCL is also used as a query language within a UML model (Akehurst and Bordbar, 2001). An OCL expression returns a value or an object within the system, but it has no side-effects, i.e., the evaluation of an expression does not change the state of the system. It is specially relevant to semantic modelling of VEs. OCL can be used to perform high-level queries for contents of the conceptual model of VEs.

- Formal foundation: OCL is an integral part of the UML standard. OCL's semantics can be expressed using UML's meta-model, or using mathematical set theory and predicate logic (OMG, 2006). However, the syntax of

OCL does not use mathematical symbols. Instead, OCL uses a notation close to natural language.

- Expressiveness: OCL is a textual, formal, and declarative language to specify expressions on any elements in a UML model. OCL expressions convey many semantics that UML diagrams can not represent by themselves, such as cardinality constraints or logical constraints.

- Extensibility: Beyond initial formalizations that led to the OCL standard specification[1], numerous researches have been conducted in extending this language to cover other constraint types, namely, temporal and spatial constraints. First, OCL can be extended to cover temporal constraints, see (Soden and Eichler, 2009) for a recent description. Alternatively, several spatial extensions of OCL were proposed to deal with spatial constraints, such as SOCL (Duboisset et al., 2005) or GeoOCL (Casanova et al., 2000; Werder, 2009). However, these spatial extensions of OCL were only related to topological constraints, not all types of spatial constraints.

- Flexibility: OCL is flexible enough to express complex spatial constraints, compared to the fine-grained classification of constraints previously presented in Section 2.1.4. OCL allows to express constraints including multiple classes and multiple instances. In addition, constraints expressed in OCL can be used in many contexts, e.g., to indicate an invariant, a guard condition, or a pre-/post-condition. For instance, a spatial constraint such as "the cube must be on the table" can be used as the post-condition of the user's spatial interaction.

Using the UML class model presented in Figure 4.2, the following examples illustrate the expressiveness of OCL.

Example 1. The constraint "the height of the table must be less than 2 metres" is formally specified as follows.

```
context Table inv:
self.height < 2
```

In this example, the element taken as the context is the `Table` class whose instance is denoted by `self`. The constraint is applied on the `height` attribute of the contextual class. It is expressed in term of an invariant, i.e., a condition that must be true for all instances of a class and during all their lifespan.

Example 2. In OCL, the `allInstances()` function returns the collection of all the instances of a class. Universal and existential quantifiers are denoted as `forAll` and `exists`. Logical relations and implication are also supported using logical operators such as `and`, `or`, `xor`, and `implies`. The following expression exemplifies that "different tables must have different positions".

[1] Current version available in http://www.omg.org/spec/OCL/

```
context Table inv:
Table.allInstances()->forAll(t1, t2 |
 t1<>t2 implies t1.position<>t2.position)
```

Example 3. To express complex constraints with multi-classes, OCL allows the navigation between classes using associations. Such a cardinality constraint as "a room has only one floor but several walls" is expressed as follows.

```
context Room inv:
self.floor->size()=1 and self.wall->size()>=1
```

In this example, `floor` and `wall` are names of association ends between the `Room`, `Floor`, and `Wall` classes. The expression `self.wall` returns the collection of all walls associated to a room, while the function `size()` counts the number of elements in a collection.

Example 4. OCL expressions can be used to form a pre- or post-condition of an operation. For instance, the prototype of the constraint "Before the *slideOut()* operation of the desk, verify that the drawer is in the *Closed* state" is as follows.

```
context Table::slideOut():Boolean
pre: self.oclInState(Drawer::slidingBehavior::Closed)
```

Thus, whenever the operation *slideOut()* on the `Desk` is launched by the user, the `pre` expression is verified to enable/disable this interaction.

4.2.2 Extended Spatial Syntax

Through the above examples, we can see that OCL facilitates the specification of constraints in a formal way. However, the original syntax of OCL is limited in expressing spatial constraints. First, let us briefly review and generalise the syntax of OCL. The central element in an OCL expression is the access to properties of an object. A property is either an attribute, an association end, or an operation. To refer to a property of an object, the general syntax is as follows.

$$\underbrace{\underbrace{self}_{object} \quad . \quad \underbrace{height}_{property} \quad < 10}_{OclExpression}$$

Using this syntax, constraints in the original OCL are unary. Whereas, spatial constraints might be binary, ternary, quaternary, or n-ary in general. Another drawback of OCL is related to the expression of *frame of reference* (FoR). Previously, we have shown that most spatial relations must be given with respect to an implicit or explicit FoR. We have modelled different types of FoR that can be: intrinsic (the relation is given by inner properties of the reference object), extrinsic (the relation is imposed by external factors on the reference object), and deictic (the relation is given by the point of view from which the

reference object is seen). In the third case, it is necessary to introduce the FoR in the context of a spatial constraint.

As a consequence, to express spatial constraints, we propose Vrx-Ocl as an extended spatial syntax of Ocl. We distinguish two types of spatial expressions: those are independent of FoR (`SpatialExpWithOutFoR`), and those are dependent on FoR (`SpatialExpWithFoR`). A complete description of the grammar of the Vrx-Ocl language can be found in Appendix B. In general, the syntax of spatial expressions in Vrx-Ocl takes the form as below.

```
SpatialExp ::= SpatialExpWithOutFoR | SpatialExpWithFoR
SpatialExpWithOutFoR ::=
   po '.'  rel_name '(' ro (',' ro)* (',' arg)?  ')'
SpatialExpWithFoR ::=
   SpatialExpWithOutFoR '@' 'viewpoint' '(' obs ')'
```

The elements of the extended syntax are explained in the following.

- `po` is the primary object
- The primary object must be associated with one or more reference objects, noted as `ro`. Thus, one `po` and several `ro` allow to represent binary, ternary, quaternary or n-ary constraints
- FoR is represented using the new keyword `@viewpoint`. `obs` is a spatial object that plays the role of observer from which the spatial constraint is seen
- `arg` is a list of arguments (e.g., a projection plane or an approximation level)
- Finally, `rel_name` stands for the name of spatial operators

Using the extended syntax of Vrx-Ocl, a binary spatial constraint without a FoR takes the form as follows.

$$\underbrace{\underbrace{object1}_{primary\,object} \quad . \quad \underbrace{disjoint}_{constraint} \quad \underbrace{(object2)}_{reference\,object}}_{SpatialExpWithOutFoR}$$

Similarly, below is the general form of ternary spatial constraint expressed in Vrx-Ocl.

$$\underbrace{\underbrace{object1}_{primary\,object} \quad . \quad \underbrace{between}_{constraint} \quad \underbrace{(object2, object3)}_{reference\,objects}}_{TernarySpatialExpWithOutFoR}$$

Alternatively, a spatial constraint with an associated FoR takes the form as follows.

$$\underbrace{\underbrace{object1}_{primary\,object} \;.\; \underbrace{leftOf}_{constraint} \; \underbrace{(object2)}_{reference\,object} \; \underbrace{@viewpoint(object3)}_{observer}}_{SpatialExpWithFoR}$$

It is important to note that the primary object and the reference objects may be resulted from a different VRX-OCL expression, for instance, a `select` expression. For example, a different way to express quaternary spatial constraints is as follows.

$$\underbrace{\underbrace{object1}_{primary\,object} \;.\; \underbrace{above}_{constraint} \; \underbrace{(a\,VRXOCL\,expression\,returns\,three\,objects)}_{reference\,objects}}_{QuaternarySpatialExpWithOutFoR}$$

4.2.3 Meta-model of Spatial Constraints

To express spatial constraints, the VRX-OCL language extensively uses new concepts that have not yet been defined at the meta-model. In this section, we describe the VRX-OCL's meta-model that is able to cover new concepts used in spatial constraints. Furthermore, we show how the meta-model of spatial constraints is integrated into the MASCARET's meta-models, and thus spatial constraints can be used as invariants, pre-/post-conditions at the conceptual model.

General View of the Meta-model

Figure 4.5 presents a general view of VRX-OCL's meta-model. The VRX-OCL's meta-model is fitted into the overall meta-models of MASCARET, UML, and OCL. There exists important links between these meta-models. First, OCL's meta-model is defined as an integral part of UML's meta-model. In our context, MASCARET is positioned as an extension of UML. Meanwhile, VRX-OCL is defined as an extension of OCL. The main elements of VRX-OCL's meta-model can be stated as follows:

- Spatial entity: Spatial entities represent elements involved in spatial constraints. In the VRX-OCL's meta-model, we abstract the concept of spatial entity by the SpatialEntityClass class, a type of domain concept (the ModelElement class).
- Spatial constraints: Spatial constraints are specified by spatial expressions. We respectively abstract the concepts of spatial constraint and spatial expression by means of two classes VRXOCLConstraint and ExpressionInVRXOL.

- Semantic areas: Before, spatial constraints have been considered as *abstract* information. In our model, we informally define semantic areas (SemanticAreaClass) as geometrical representations related to a given spatial constraint. By means of semantic areas, our main goal is to conceptualise, and then visualise, abstract spatial constraints in VEs.

- Behaviours based on spatial constraints: Behaviours of users and agents in VEs intensively require spatial knowledge. To model spatial behaviours in VEs, we are based on the behaviour model of MASCARET and UML.

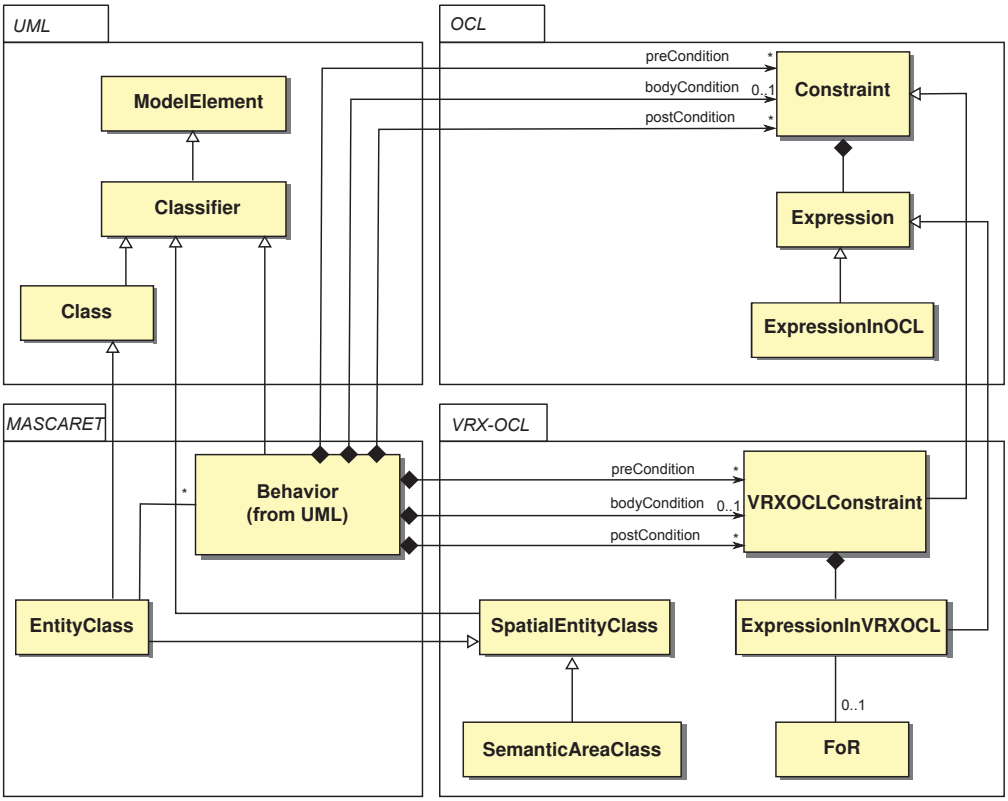


Figure 4.5: Position of VRX-OCL's meta-model within UML, OCL, and MASCARET's meta-model.

Meta-model of Spatial Entities

Figure 4.6 shows the meta-model representing the concept of spatial entity in the VRX-OCL approach. Our concept of spatial entity is defined as a generalisation of the concept of entity previously defined in MASCARET (the abstract class SpatialEntityClass). Therefore, every entity is a spatial entity. A spatial entity has a referential point (the Point class). In VRX-OCL, Point is an extension of the already existing meta-class Point borrowed from MASCARET's meta-model. A referential point has topological properties (the TopologicalProperty class) that are considered as a kind of property in a domain model (the Property class).

However, our concept of spatial entity has additional semantics with regard to the original concept of entity in MASCARET. First, we enrich the concept of referential point with more spatial information needed for computing spatial constraints (e.g., intrinsic direction represented by the three vectors $\vec{front}$, $\vec{left}$, $\vec{above}$). Second, every spatial entity has additional methods. Each method is used to compute a corresponding spatial relationship, such as `disjoint()`, `between()`, `leftOf()`, or `distanceTo()`.

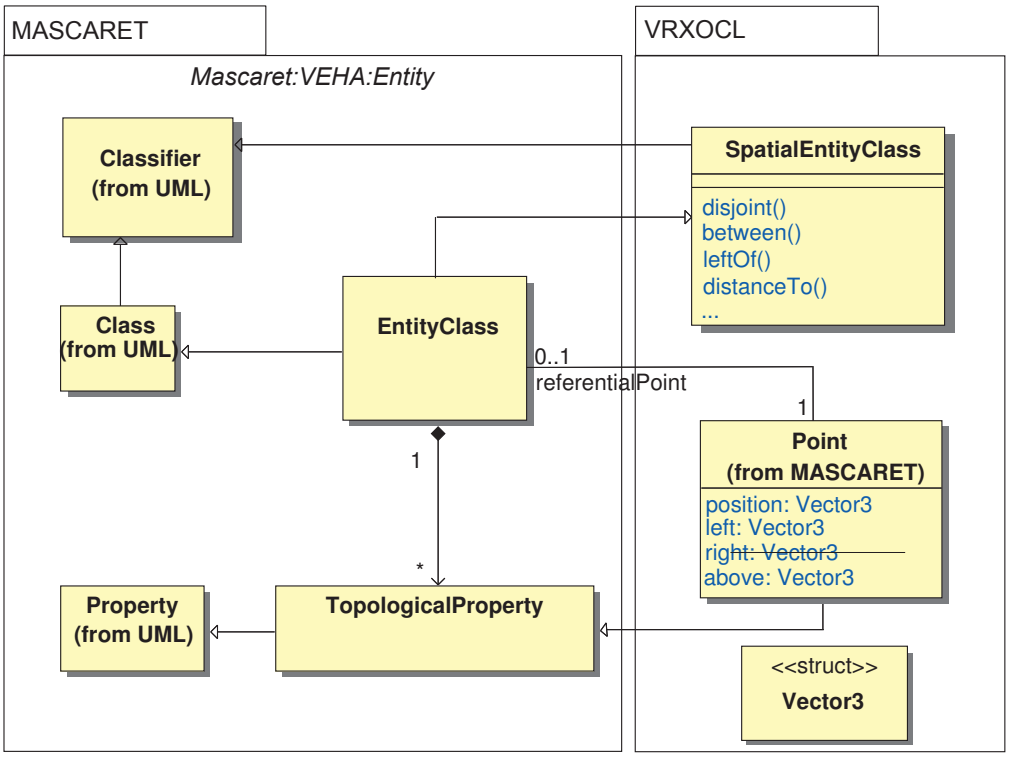


Figure 4.6: Class diagram representing the meta-model of SpatialEntity.

Meta-model of VRX-OCL Constraints

Figure 4.7 illustrates the meta-model of constraint. In the VRX-OCL approach, spatial constraints as well as other types of constraint are represented by the VRXOCLConstraint class (derived from the Constraint class existing both in UML and MASCARET). A constraint contains an expression (the ExpressionInVRXOCL class). Every expression returns a value (the ValueSpecification class). To evaluate a VRX-OCL expression, an expression parser (the VRXOCLExpressionParser class) is proposed. It is necessary to note that at the meta-level, the VRXOCLExpressionParser class simply represents a parser in an abstract manner.

Linking VRX-OCL Constraints and Behaviours

Based on UML, MASCARET provides two different ways to specify behaviours of entities within VEs. First, an entity may offer several operations, i.e., services

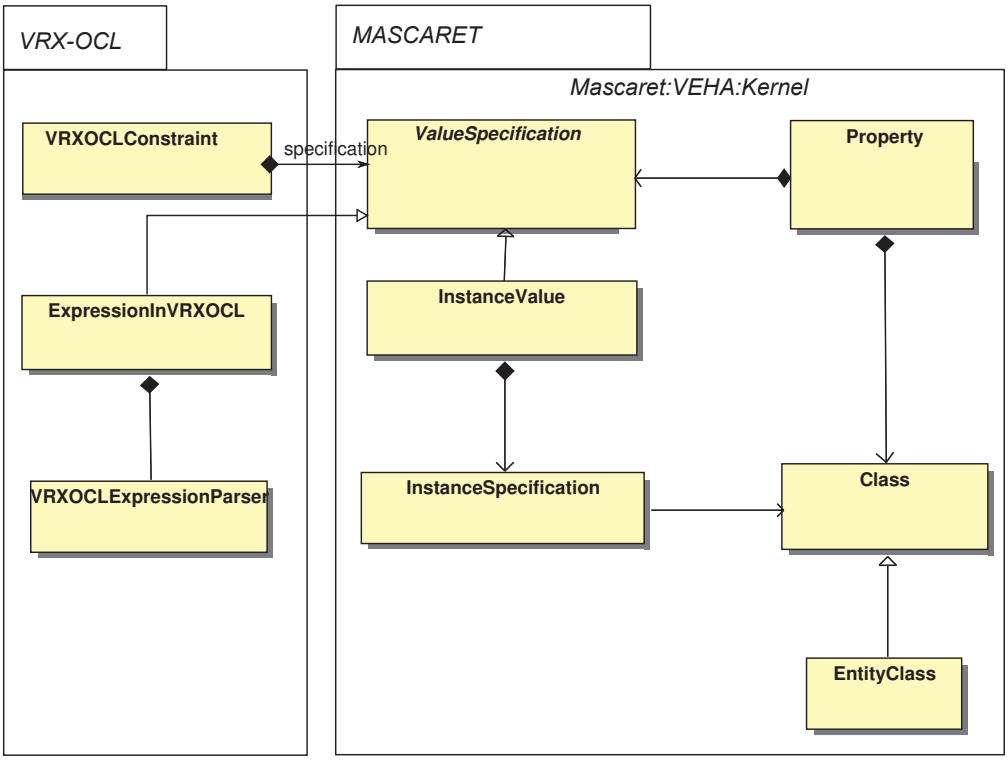


Figure 4.7: Class diagram representing the meta-model of VRXOCLConstraint.

that users and other entities can perform on itself. Second, an entity can be associated with a state machine. The state machine defines possible states of the entity. Spatial constraints allow to convey and precise the semantics of behaviours of entities. In the following, we show how the meta-model of spatial constraints can be integrated into MASCARET's meta-models of behaviours.

Spatial constraints (the VRXOCLConstraint class) can be used as pre- or post-conditions of an operation (the Operation class). The link between spatial constraints and operations are illustrated the meta-model in Figure 4.8. The Operation class allows to express what a spatial entity or a user can perform on another entity. An operation uses constraints as a pre- or post-condition. Even the execution of an operation (bodyCondition) can also be defined by an expression.

Figure 4.9 shows the link between spatial constraints (the VRXOCLConstraint class) and states machines associated to spatial entities (the StateMachine class). A constraint allows to express a state invariant. In addition, a constraint can be used as a guard condition for a state transition that is triggered by an event. To deal with spatial events, we enrich the meta-model by a new class, named SpatialEvent. This class handles events that lead to the re-evaluation of spatial constraints, such as motion, or rotation of spatial entities.

Meta-model of Semantic Areas

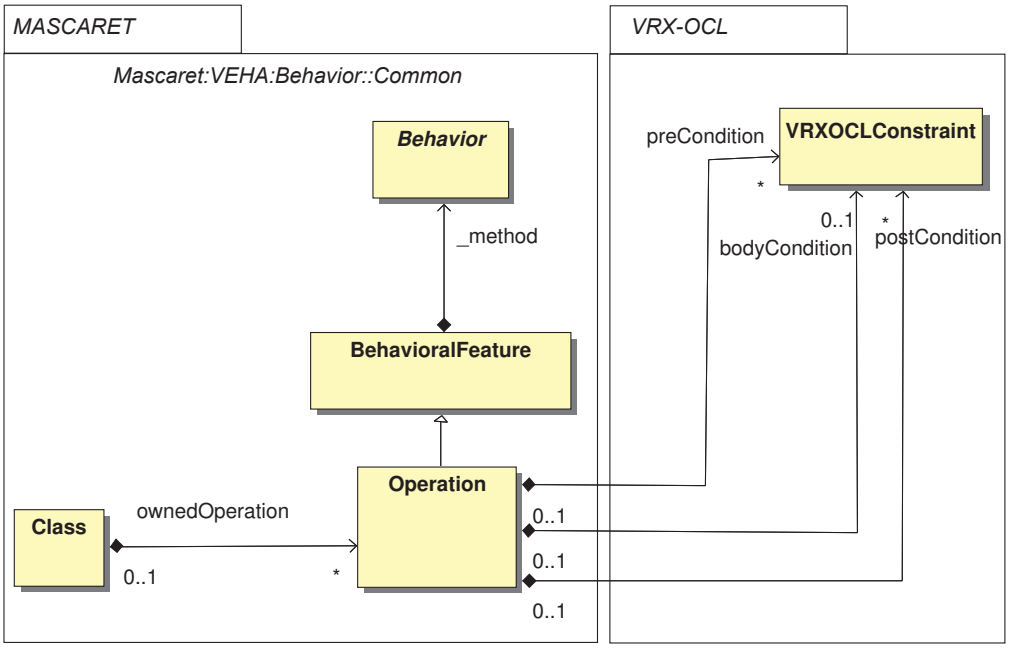


Figure 4.8: Class diagram representing the link between VRXOCLConstraint and Behavior.

Despite spatial relationships convey many semantics about space, they are however often considered as a kind of abstract information in several spatial semantic models of VEs (Bowman et al., 2003). Based on our model of spatial constraints, we formalise the concept of *semantic area*. Informally, we define a semantic area as a geometrical and visual representation related to a spatial relationship. For example, given a constraint such as "leftOf" between a primary entity and a reference entity, the semantic area is the graphical representation corresponding to the area "on the left" of the reference entity.

Our meta-model of semantic areas is illustrated in Figure 4.10. The main idea is that we consider a semantic area as a spatial entity. Nevertheless, the main difference is that a semantic area does not exhibit behaviours like a spatial entity. Semantic areas are generalised by the SemanticAreaClass class, derived from the SpatialEntityClass class. Semantic areas are instantiated and represented by the SemanticArea class, derived from the InstanceSpecification class. Similar to the concept of spatial entity, a semantic area has a shape and other geometrical properties. However, these geometrical properties are not user-defined but automatically computed and generated. The geometry of a semantic area can be built upon several geometrical primitives. For example, the semantic area "between two spatial entities" is built from the tangent planes of the two spatial entities. As we will see later, like all 3D objects, semantic areas are mainly based on simple geometrical primitives, such as points, lines, triangles, quadrangles, and surfaces. These geometrical primitives are generated based on our methods for computing spatial constraints.

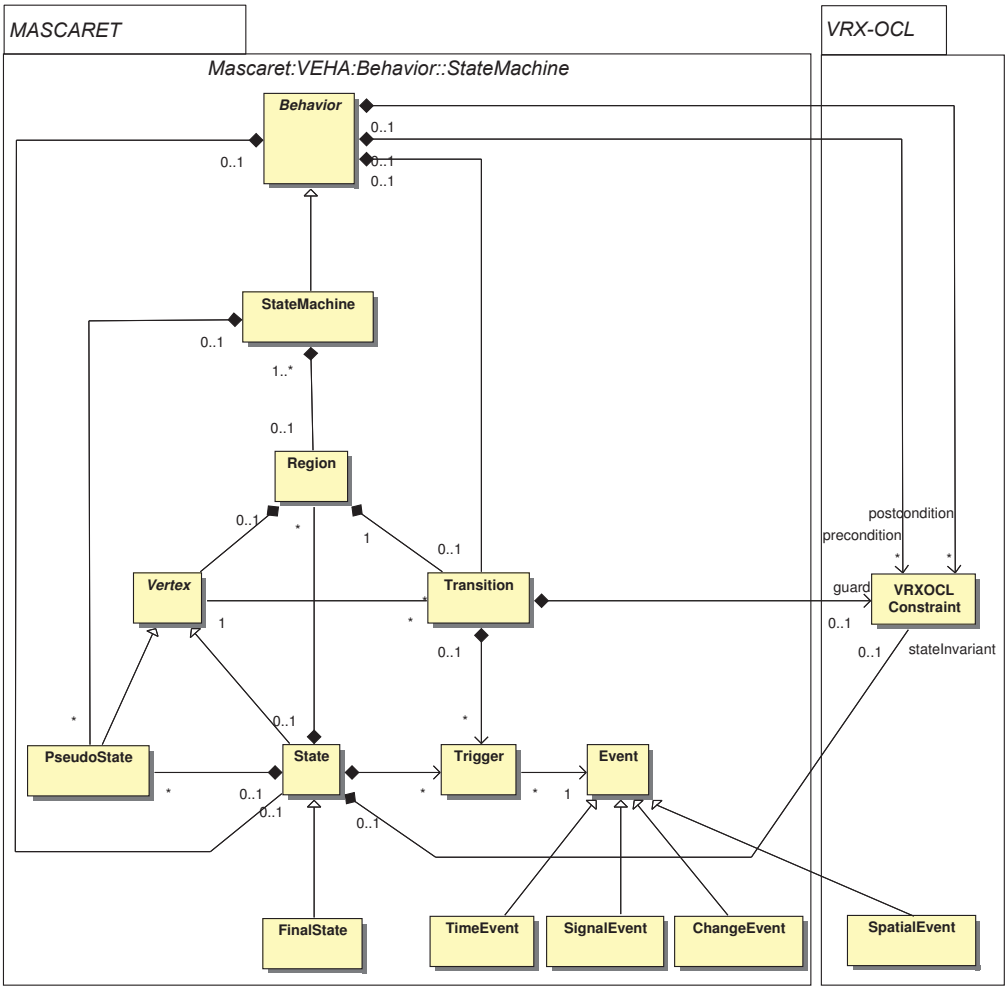


Figure 4.9: Class diagram representing the link between VRXOCLConstraint and StateMachine.

Furthermore, based on our meta-model of semantic area, it is interesting to note that a semantic area can be instantiated (i.e., at the M0 level). The instantiation of a semantic area takes several parameters as input, such as the name of spatial constraint, the identifiers of spatial entities involved, or the name given to this semantic area. The following example illustrates the instantiation of the semantic area "left of the desk named Desk1".

```
<SemanticArea
        name = "areaLeftOfTheDesk1"
        class = "LeftAreaClass"
        constraint = "leftOf"
        entity= "Desk1"
/>
```

Similarly, the semantic area "between the red table and the blue table" is

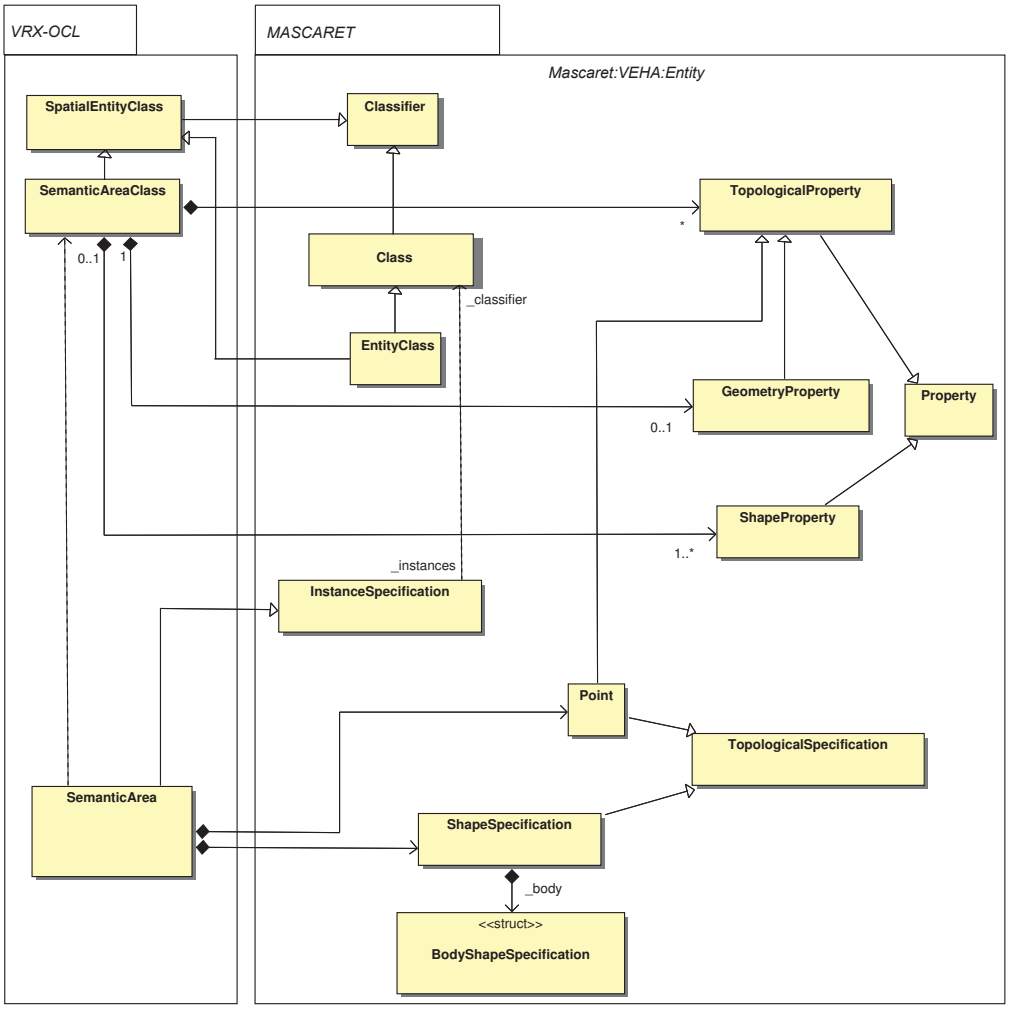


Figure 4.10: Class diagram representing the meta-model of SemanticArea.

instantiated as follows.

```
<SemanticArea
        name = "areaBetweenTheRedAndBlueTables"
        class = "BetweenAreaClass"
        constraint = "between"
        entity= "redTable"
        entity= "blueTable"
/>
```

Thanks to the meta-model of semantic areas, an abstract concept such as spatial relationships can be instantiated and further visualised. In the following sections, we detail the syntax and related semantic areas for each type of spatial constraint. Moreover, each type of spatial constrain is illustrated through examples from the simple and realistic VE (i.e., the virtual room) that has been presented in the previous chapter.

4.3 Topological Operators

Description

Topological operators of the VRX-OCL language allow to specify topological constraints in VEs. Our theoretical model of topological constraints has been presented in Section 3.4.

Syntax

Binary topological operators in VRX-OCL take the form of:
`po.rel_name(ro)`
where:

- `po` and `ro` are respectively the primary and reference objects.
- `rel_name` is one of the eight basic topological relations: `disjoint`, `meet`, `overlap`, `inside`, `coveredBy`, `contains`, `covers`, `equal`.

Example 5. Using VRX-OCL, Constraint 3 "every table must be disjoint with other tables in the room " is specified as follows.

```
context Table inv:
  Table.allInstances()->forAll(t1, t2 |
        t1<>t2 implies t1.disjoint(t2) )
```

Example 6. Figure 4.11a illustrates another example of topological constraint. That is "in the room, the black box must be on the red table". This constraint is expressed in VRX-OCL as follows.

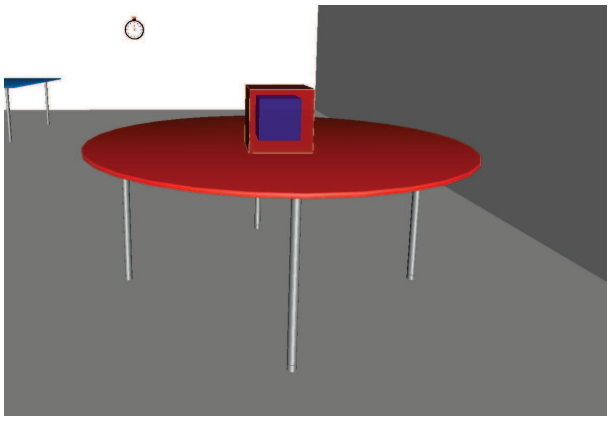
```
context Room inv:
let _blackBox:Box=Box.allInstances()
        ->select(name="blackBox"),
    _redTable:Table=Table.allInstances()
        ->select(name="redTable")
in  _blackBox.meet(_redTable)
```

Semantic Areas

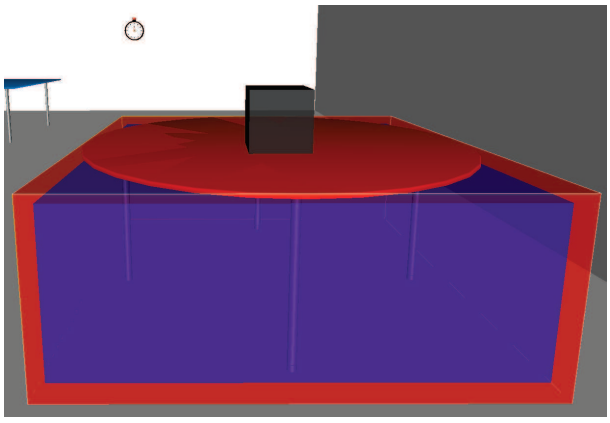
With regard to topological constraints, the semantic areas do not aim to visualise the areas "inside" or "overlap" formed by two entities. Instead, they aim to visualise the contact surfaces between two entities involved in a topological constraint. The topological semantic areas are:

- The thick boundary of the primary object: Given the topological constraint "the black box must be on the red table", Figure 4.11b shows the thick boundary of the primary object (i.e., the black box).
- The thick boundary of the reference object: Similarly, Figure 4.11c illustrates the thick boundary of the reference object (i.e., the red table).

(a) The topological constraint "in the room, the black box must be on the red table".



(b) Visualising the "thick boundary" of the primary object (i.e., the box).



(c) Visualising the "thick boundary" of the reference object (i.e., the table).

Figure 4.11: Semantic areas related to topological constraints.

The representation and visualisation of topological semantic areas (i.e., the thick boundaries) help users in interacting with spatial entities in a more precise manner.

4.4 Projective Operators

4.4.1 Ternary Projective Relations in Orthographic View

Description

Orthographic ternary projective operators allow to describe projective constraints among three spatial entities viewed from an orthographic view (i.e., front view, top view, and side view). Our theoretical model of ternary orthographic projective constraints has been presented in Section 3.5.1.

Syntax

Ternary projective constraints under orthographic views are formalised in the context of a frame of reference as follows.

`po.rel_name(ro1,ro2,plane,collinearity_level)@viewpoint(obs)`

where:

- `po`, `ro1`, `ro2` are respectively three spatial objects.
- `rel_name` is the name of the relationship. The five basic relations are `beforeOnPlane`, `betweenOnPlane`, `afterOnPlane`, `rightsideOnPlane`, and `leftsideOnPlane`.
- `plane` is the projection plane that is one of the three planes: `XY`, `YZ`, or `XZ`. Each plane corresponds to an orthographic view.
- `collinearity_level` represents different levels of detail related to the representation of spatial entities, involving: 1 - spatial entities are represented by their referential points; 2 - spatial entities are represented by their bounding boxes.
- `obs` stands for the viewer from which the relationship is seen. Among the five basic relations described above, only `leftsideOnPlane` and `rightsideOnPlane` relations need a viewer as an explicit frame of reference to disambiguate the relation between the primary object and the two reference objects under different orthographic views.

Example 7. As illustrated in Figure 4.12a, Constraint 14 "from the top-view of the room, the triangular green table must be on the left side of the rounded red table and the rectangular blue table" is expressed in VRX-OCL as below.

```
context Room inv:
let greenT:Table = Table.allInstances()
        ->select(name='greenTable'),
```

```
    redT:Table = Table.allInstances()
        ->select(name='redTable'),
    blueT:Table = Table.allInstances()
        ->select(name='blueTable')
in  greenT.rightsideOnPlane(redT, blueT, 'XY')
  @viewpoint(Camera.allInstances()
          ->select(name='TOP_CAMERA'))
```

In this example, we assume that the top-view results from the projection of the scene onto the `XY` plane. The evaluation of the constraint depends on `TOP_CAMERA`, an instance of the `Camera` class that contains information about the top-view.

Semantic Areas

There are five semantic areas. Each area corresponds to a projective constraint.

- The area "between" two spatial entities on a projection plane: Figure 4.12b illustrates the area "between" the rounded red table and the rectangular blue table under a top-view.
- The area "before" two spatial entities on a projection plane, see Figure 4.12c.
- The area "after" two spatial entities on a projection plane, see Figure 4.12d.
- The area "leftside" two spatial entities on a projection plane, see Figure 4.12e.
- The area "rightside" two spatial entities on a projection plane, see Figure 4.12f.

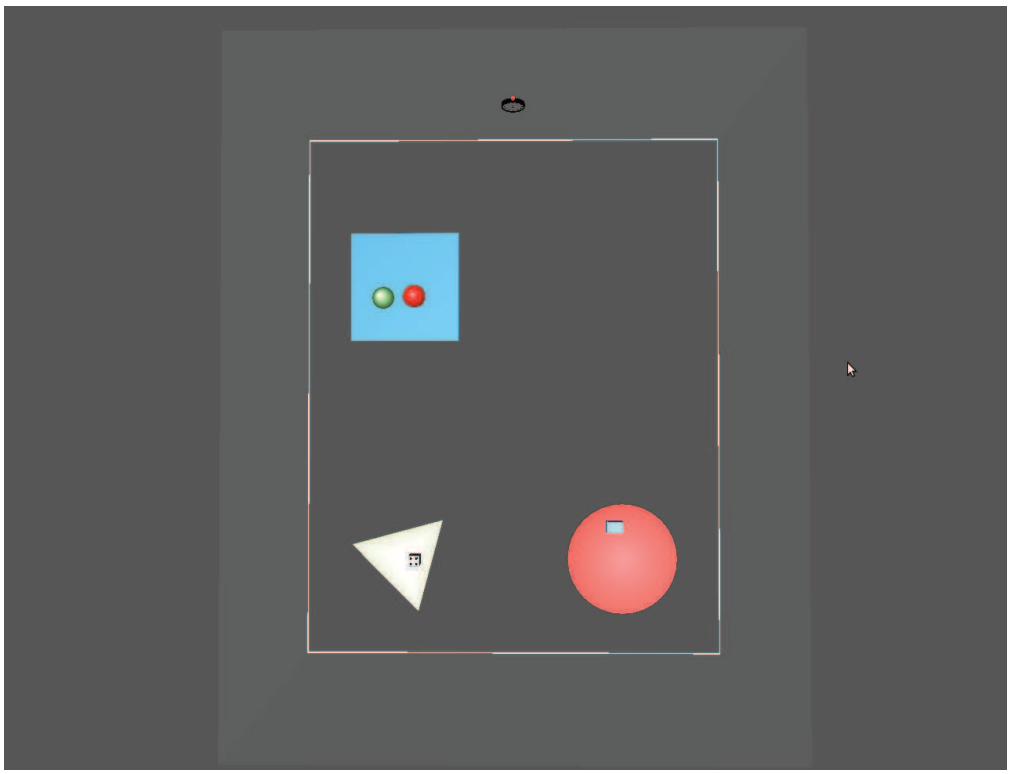
The five semantic areas are built upon other geometrical primitives that are internal tangent lines (see Figure 4.12g) and external tangent lines (see Figure 4.12h).

4.4.2 Ternary Projective Relations in Immersive View

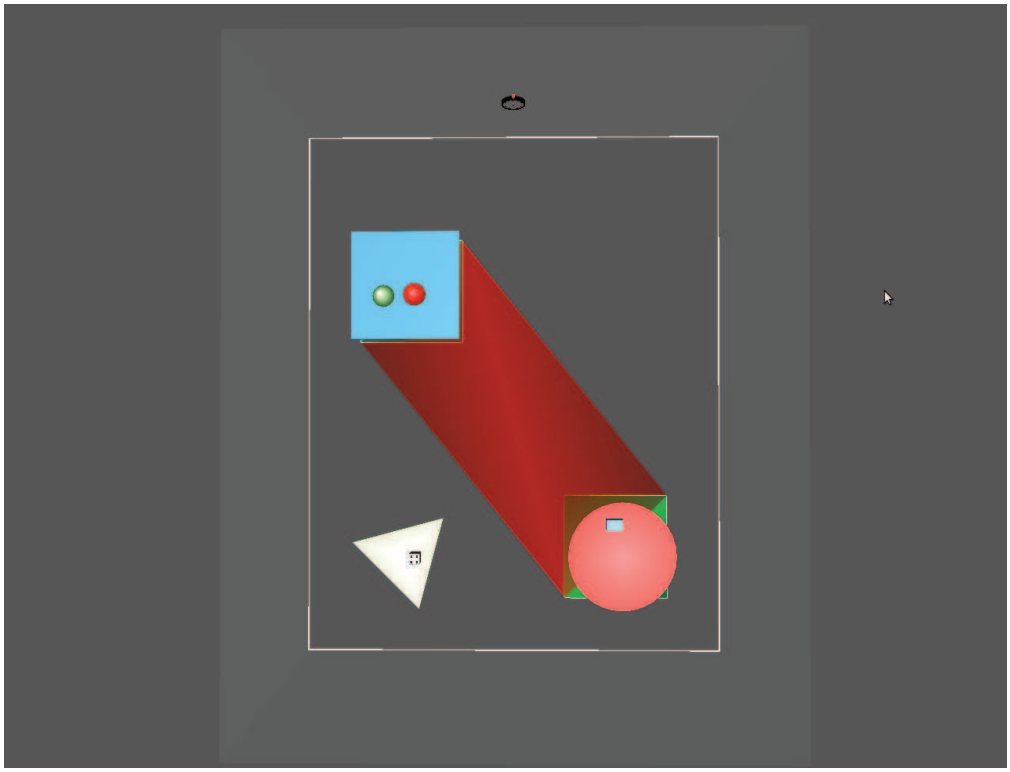
Description

Ternary projective operators allow to describe projective constraints among three spatial entities under an immersive view. Our theoretical model of ternary projective constraints has been presented in Section 3.5.1.

Syntax

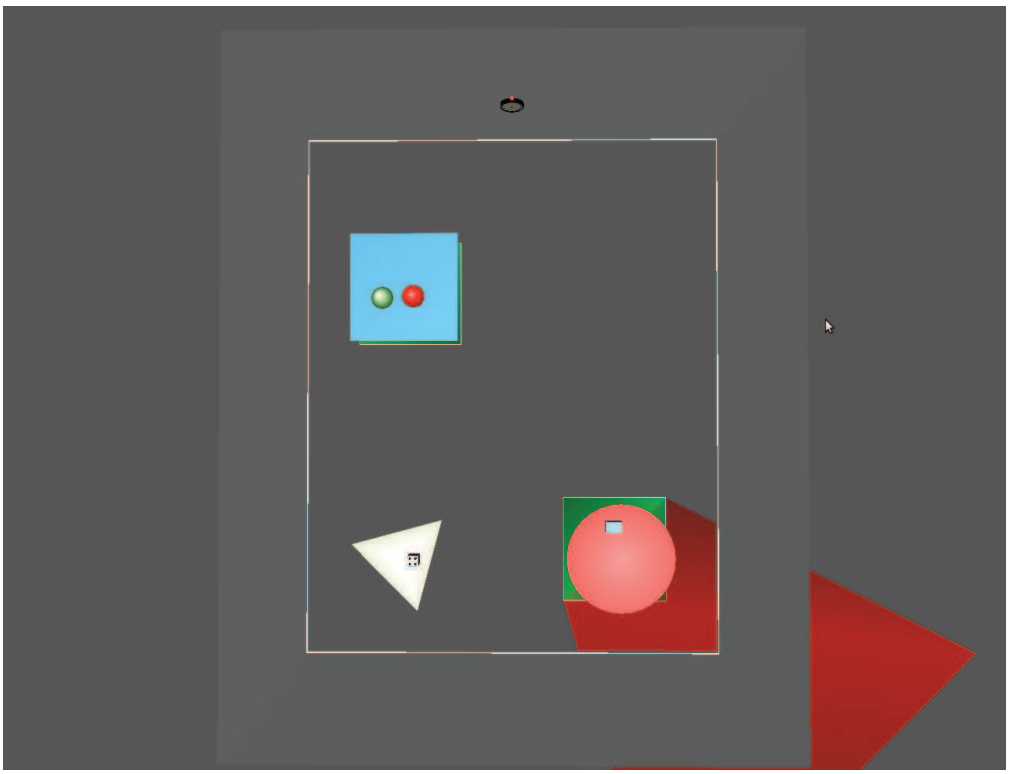


(a) The ternary orthographic projective constraint "from the top-view of the room, the triangular green table must be on the left side of the rounded red table and the rectangular blue table".

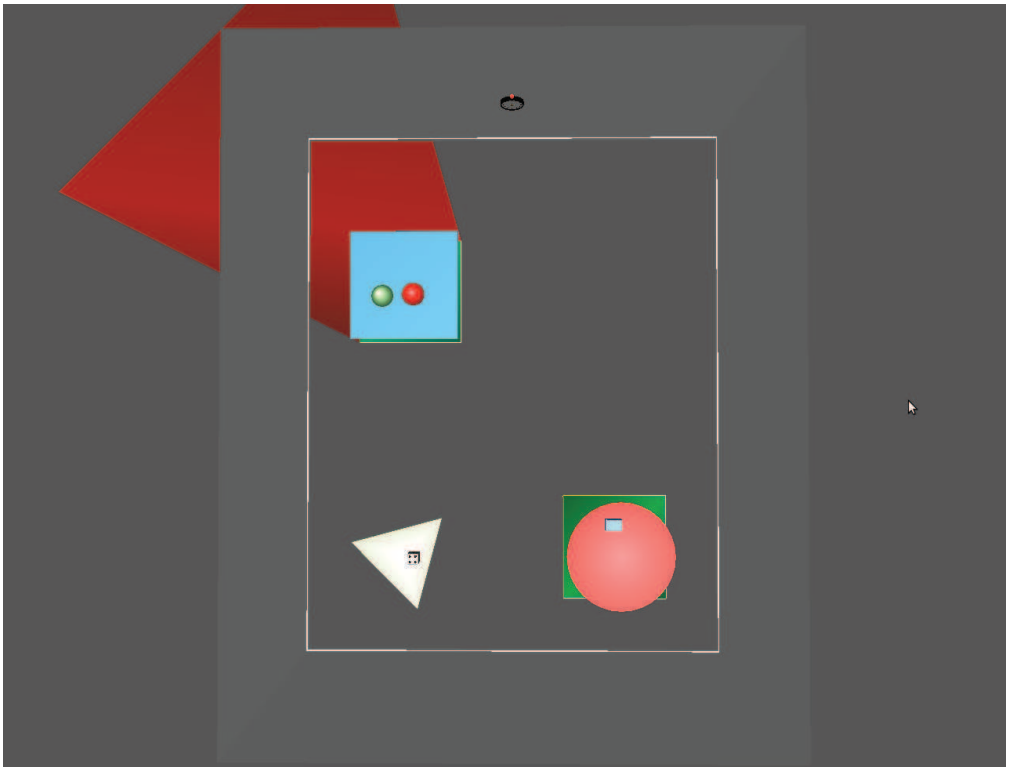


(b) The area "between" the rounded table and the rectangular table on a projection plane.

Figure 4.12: Semantic areas related to ternary projective constraints under orthographic views.

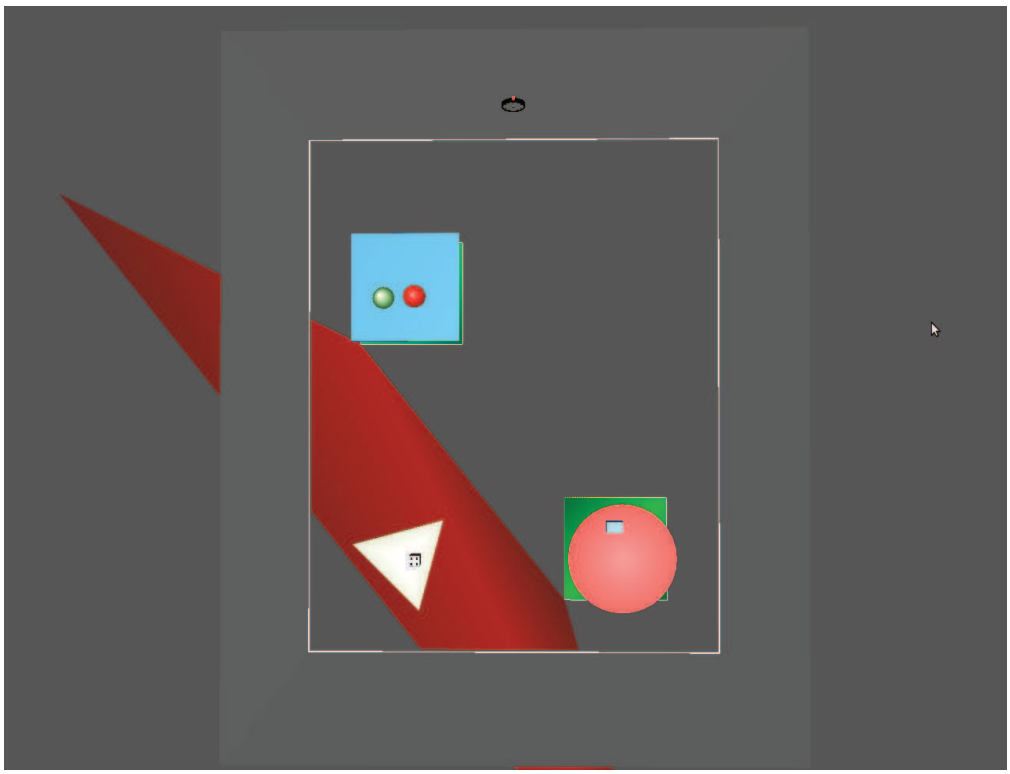


(c) The area "before" the rounded table and the rectangular table on a projection plane.

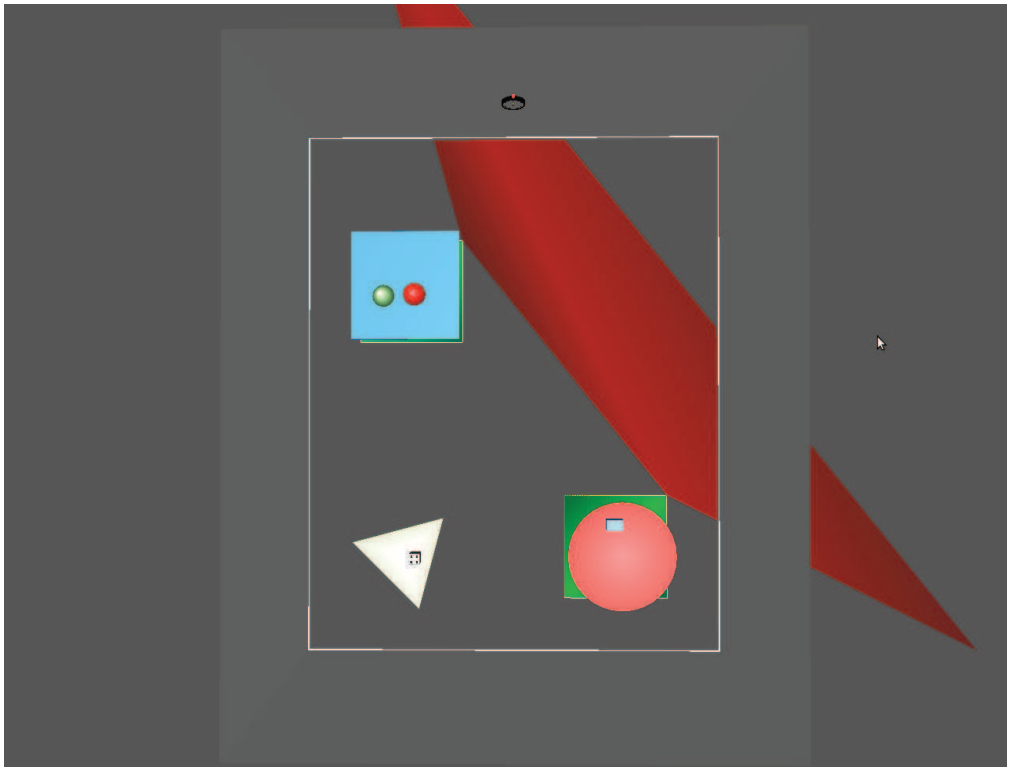


(d) The area "after" the rounded table and the rectangular table on a projection plane.

Figure 4.12: Semantic areas related to ternary projective constraints under orthographic views.

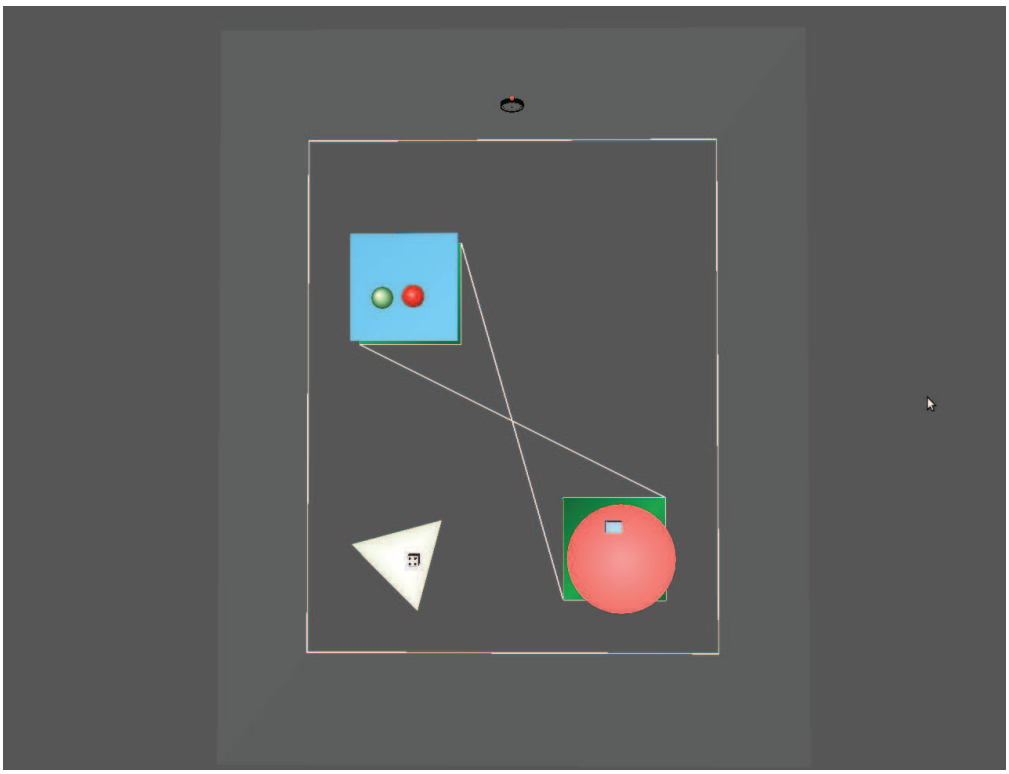


(e) The area “leftside” the rounded table and the rectangular table on a projection plane.

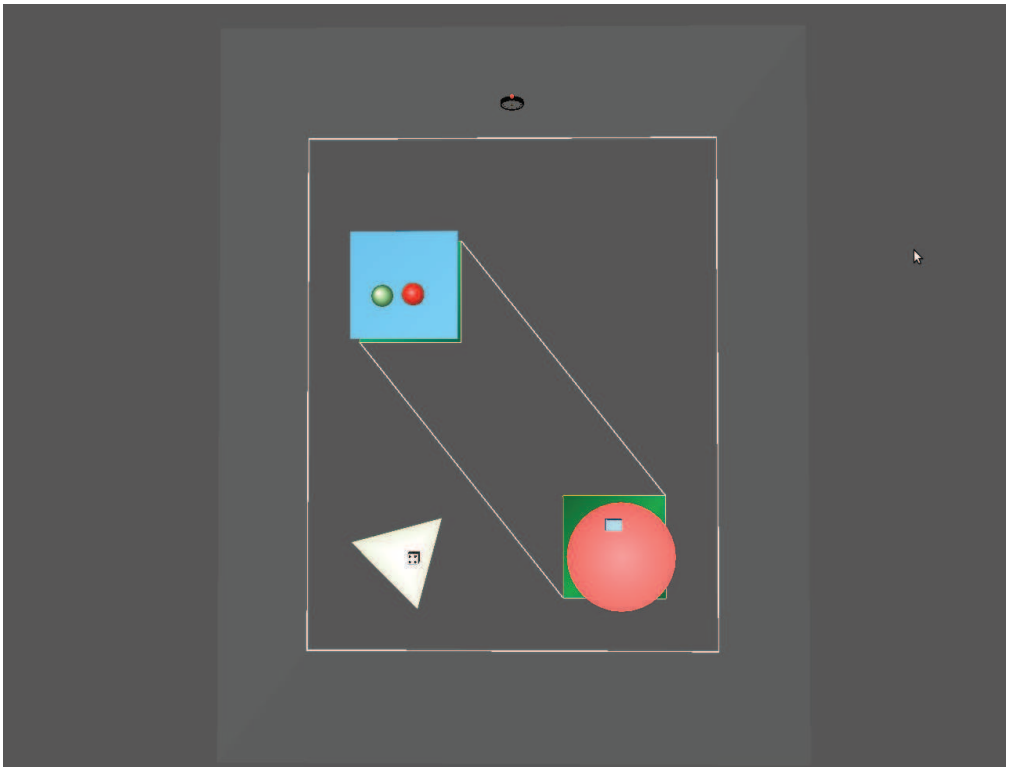


(f) The area “rightside” the rounded table and the rectangular table on a projection plane.

Figure 4.12: Semantic areas related to ternary projective constraints under orthographic views.



(g) The internal tangent lines.



(h) The external tangent lines.

Figure 4.12: Semantic areas related to ternary projective constraints under orthographic views.

As ternary projective constraints in immersive perceiving mode are inherently independent of the reference frame, we subsequently integrate these relations into VRX-OCL as new spatial operators that take the form as follows:

`po.rel_name(ro1, ro2, collinearity_level)`

where:

- `po`, `ro1`, `ro2` are respectively three spatial objects.
- `rel_name` is one of the followings: `before`, `between`, `after`, `aside`.
- `collinearity_level` represents different levels of detail related to the representation of spatial entities, involving: 1 - spatial entities are represented by their referential points; 2 - spatial entities are represented by their bounding boxes.

Example 8. Figure 4.13a provides an alternative view of the room. This view is used to illustrate Constraint 4. That is "in the room, the black chair must be between the rounded red table and the rectangular blue table". The VRX-OCL expression representing this constraint is as follows.

```
context Room inv:
let _blackChair:Chair = self.chair()
                        ->select(name='blackChair')
    ,_redTable:Table = self.table()
                        ->select(name='redTable')
    ,_blueTable:Table = self.table()
                        ->select(name='blueTable')
in _blackChair.between(_redTable, _blueTable)
```

Semantic Areas

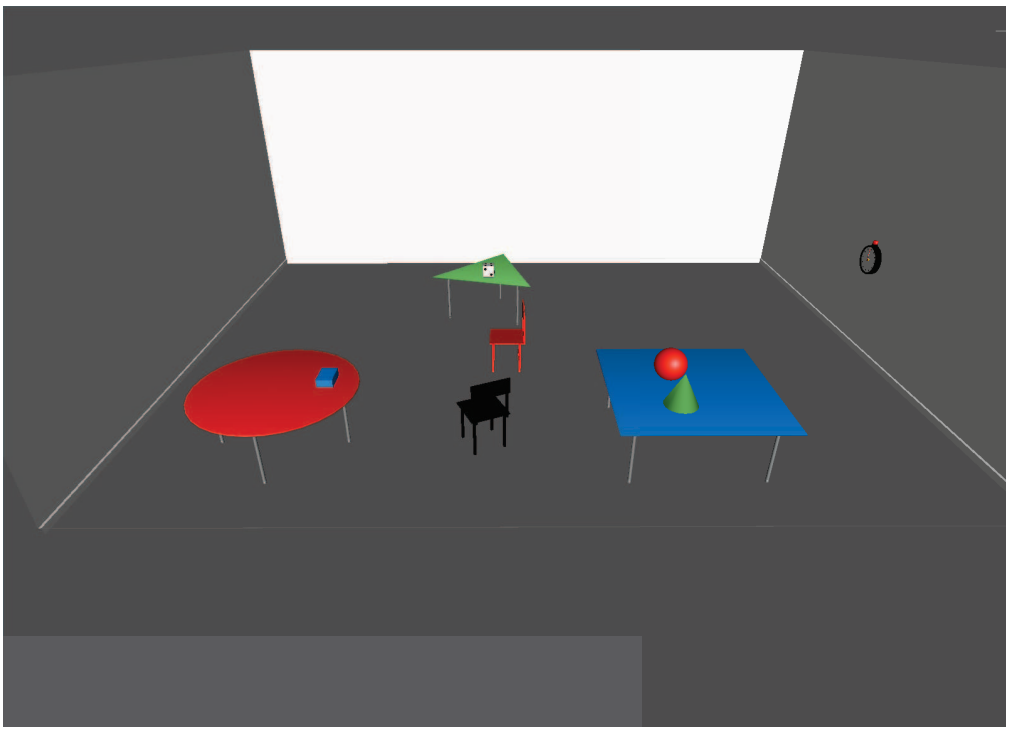
Based on our model of ternary projective constraints, the following semantic areas can be defined and visualised:

- The area "between" two spatial entities: Figure 4.13b shows the area "between" the rounded red table and the rectangular blue table.
- The area "before" two spatial entities, see Figure 4.13c.
- The area "after" two spatial entities, see Figure 4.13d.

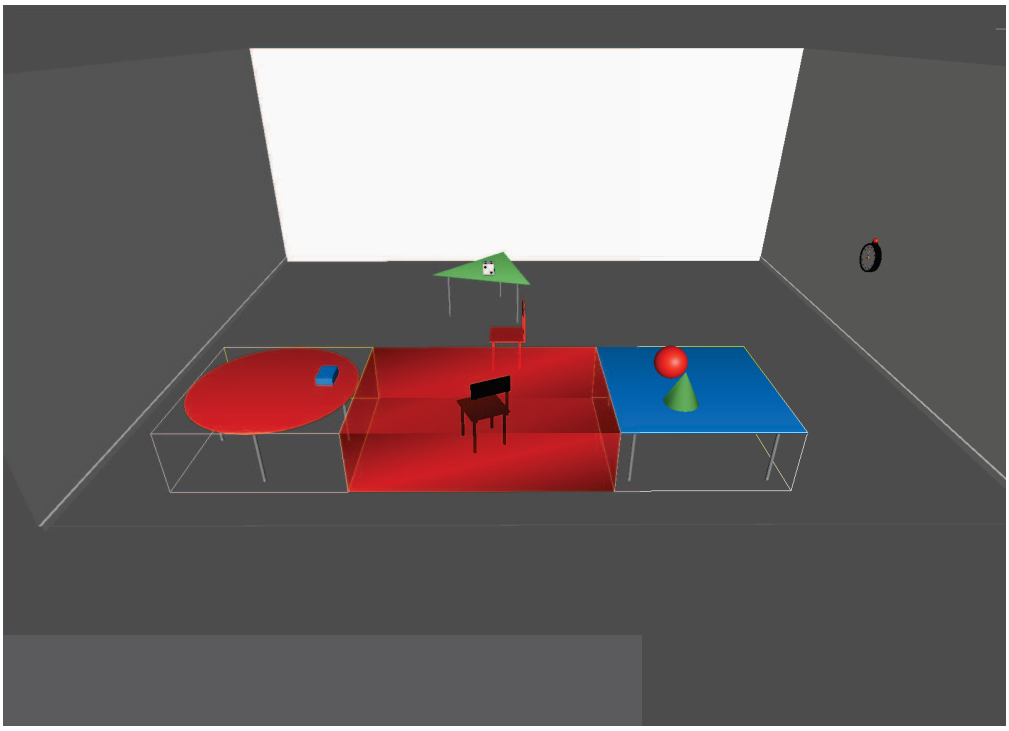
Note that, the semantic area corresponding to the constraint "aside" can not be visualised. Instead, it is geometrically defined as the complementary part of the union of the semantic areas "between", "before", and "after".

4.4.3 Quaternary Projective Relations

Description

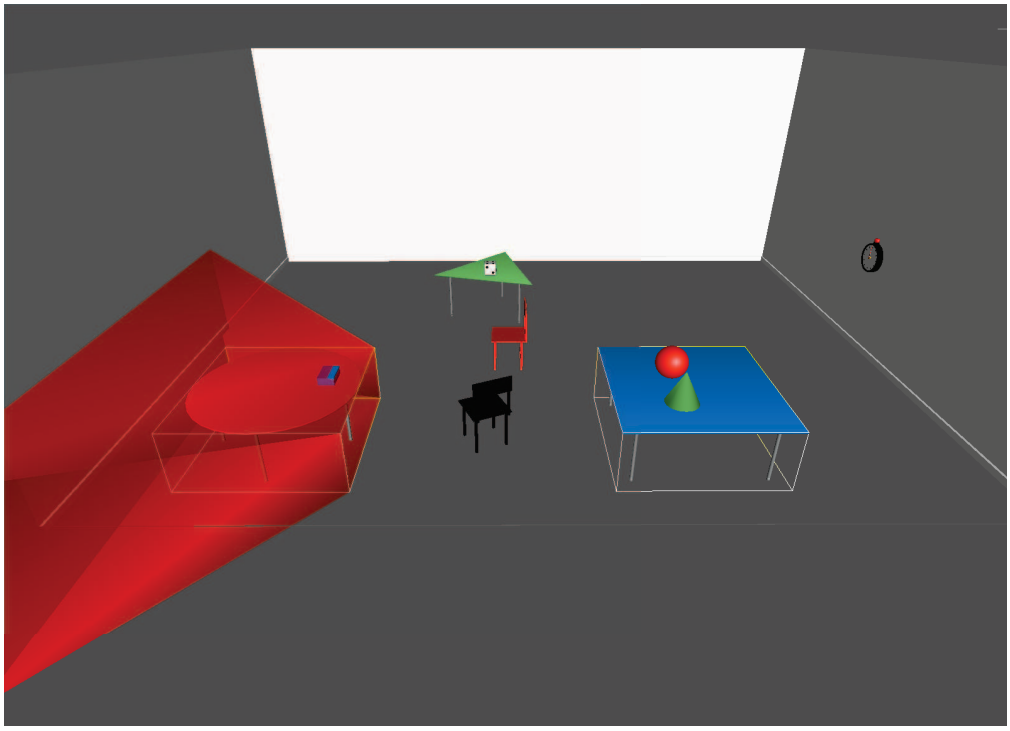


(a) The ternary projective constraint "in the room, the black chair must be between the rounded red table and the rectangular blue table".

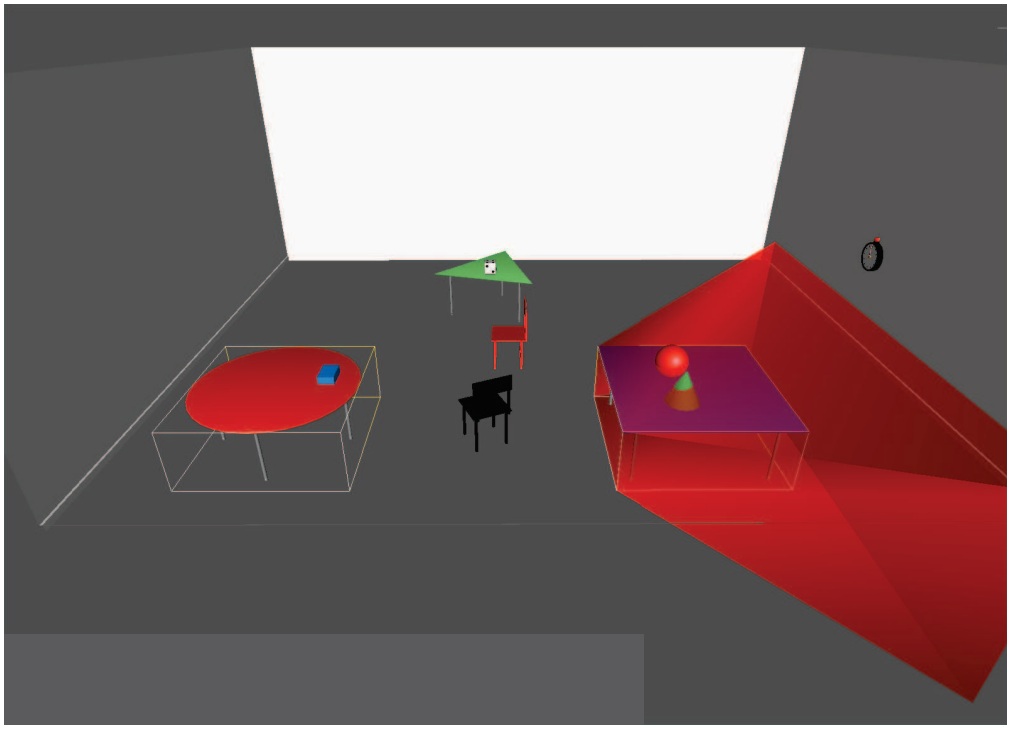


(b) The area "between" the rounded table and the rectangular table.

Figure 4.13: Semantic areas related to ternary projective constraints under immersive views.



(c) The area “before” the rounded table and the rectangular table.



(d) The area “after” the rounded table and the rectangular table.

Figure 4.13: Semantic areas related to ternary projective constraints under immersive views.

Quaternary projective operators allow to describe projective constraints among four spatial entities. Our theoretical model of quaternary projective constraints has been presented in Section 3.5.2.

Syntax

Quaternary projective constraints are formalised in VRX-OCL as follows.
`po.rel_name(ro1, ro2, ro3,coplanarity_level)`
where:

- `po` and `ro1`, `ro2`, `ro3` are respectively the primary and three reference objects.
- `rel_name` is one of the following: `inside`, `outside`, `above`, `below`.
- `collinearity_level` represents different levels of detail related to the representation of spatial entities, involving: 1 - spatial entities are represented by their referential points; 2 - spatial entities are represented by their bounding boxes.

Example 9. Constraint 5 "in the room, the red chair must be surrounded by the three tables: the triangular green table, the rounded red table, and the rectangular blue table" is expressed in VRX-OCL as follows.

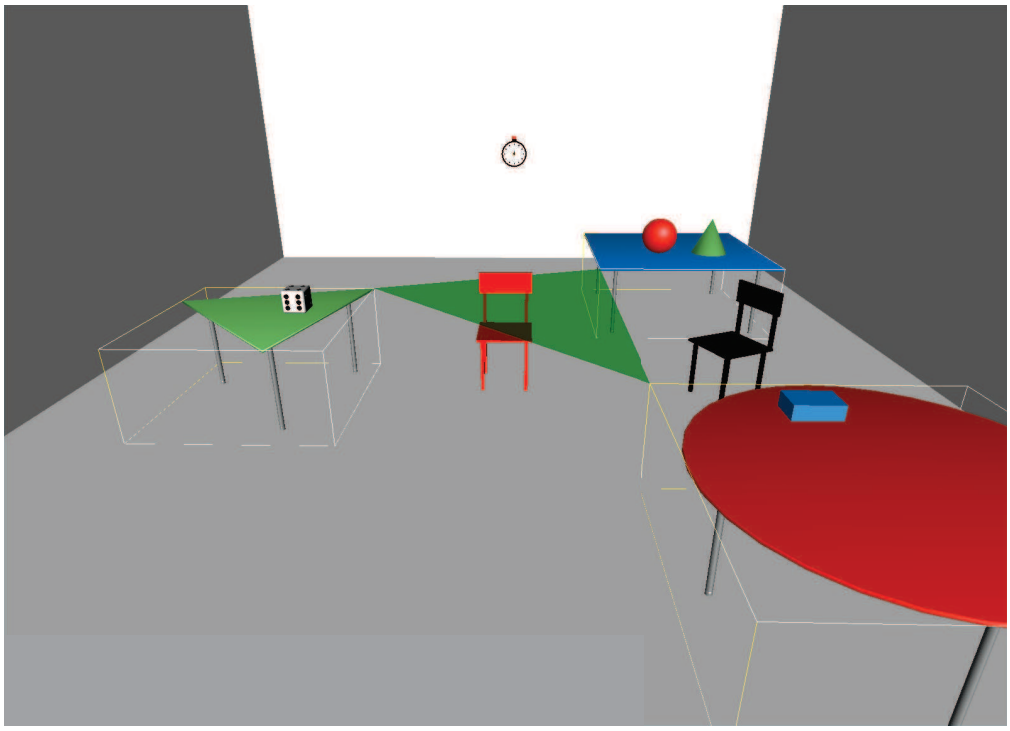
```
context Room inv:
let _blackChair:Chair = self.chair()
                        ->select(name='blackChair')
    ,_redTable:Table = self.table()
                        ->select(name='redTable')
    ,_blueTable:Table = self.table()
                        ->select(name='blueTable')
    ,_greenTable:Table = self.table()
                        ->select(name='greenTable')
in _blackChair.inside(_redTable, _blueTable, _greenTable)
```

Semantic Areas

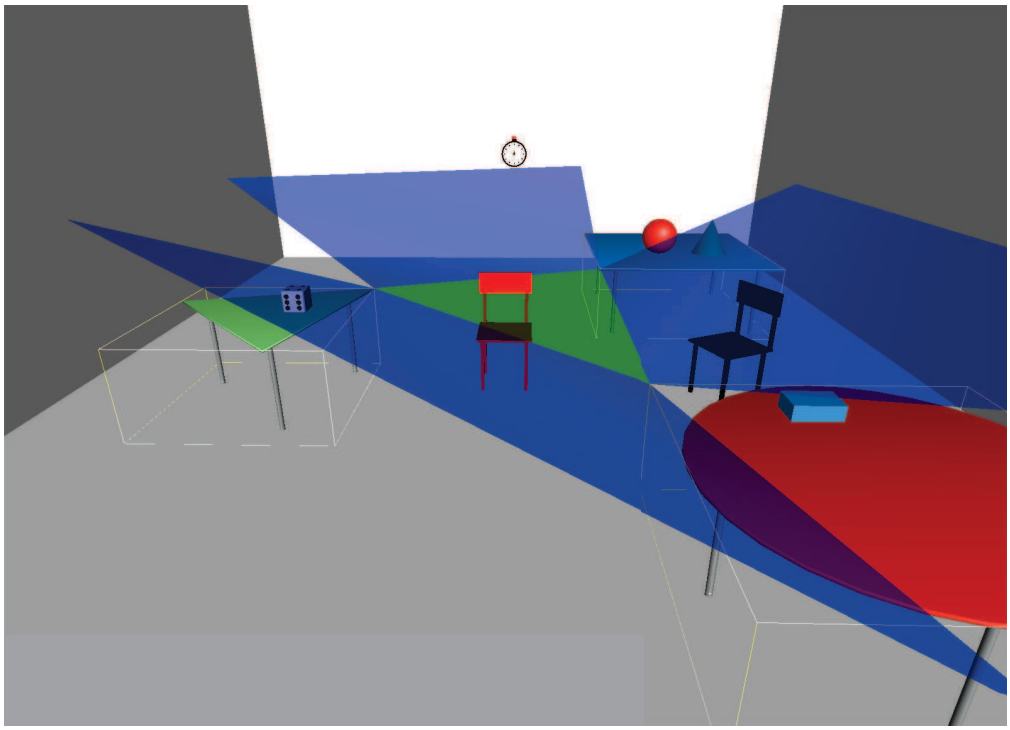
With regard to quaternary projective constraints, the semantic areas are:

- The area "above" three spatial entities: Figure 4.14b shows the area "above" the three tables (i.e., the triangular table, the rectangular table, and the rounded table) in the room.
- The area "below" three spatial entities
- The area "inside (surrounded)" by three spatial entities

The three semantic areas are built upon other geometrical primitives, they are internal tangent planes and external tangent lines (see Figure 4.14a).



(a) The external tangent planes of three spatial entities.



(b) The area "above three spatial entities".

Figure 4.14: Semantic areas related to quaternary projective constraints.

4.5 Directional Operators

4.5.1 Direction From a First-person Perspective

Description

To express direction form a first-person perspective, we use 27 directional operators. Our model of direction from a first-person perspective has been presented in Section 3.6.1.

Syntax

In VRX-OCL, directional operators from a first-person perspective is binary and takes the form as follows:

```
po.rel_name(ro)
```

where:

- `po`, `ro` are respectively the primary and reference objects.
- `rel_name` is one of the 27 directional relations, such as `isFrontOf`, `isLeftOf`, `isAboveOf`, etc.

Example 10. Considering a situation in Constraint 6, a directional constraint from a first-person perspective such as "in the room, the rounded red table must be in front of the desk" is expressed as follows.

```
context Room inv:
let _redTable:Table = Table.allInstances()
        ->select(name = 'redTable'),
    _desk:Desk = Desk.allInstances()
        ->select(name = 'deskTable')
in _redTable.frontOf(_desk)
```

Semantic Areas

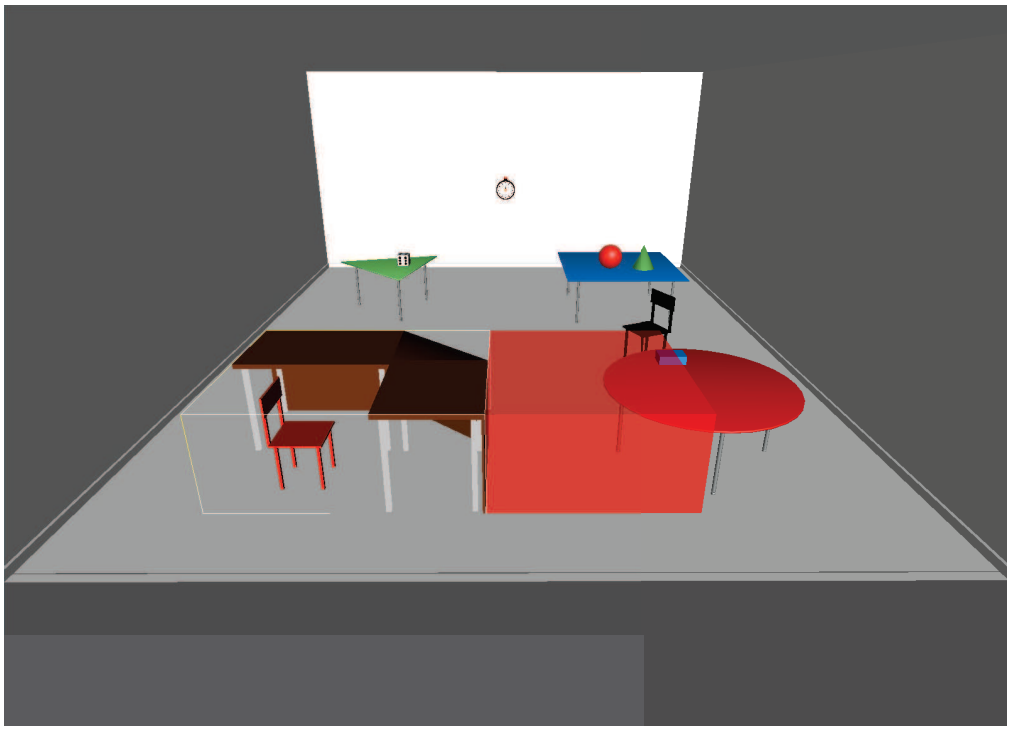
There are 27 semantic areas that respectively visualise 27 directional constraints from a first person perspective. Some of them are illustrated from Figure 4.15a to Figure 4.15j.

4.5.2 Direction From a Third-person Perspective

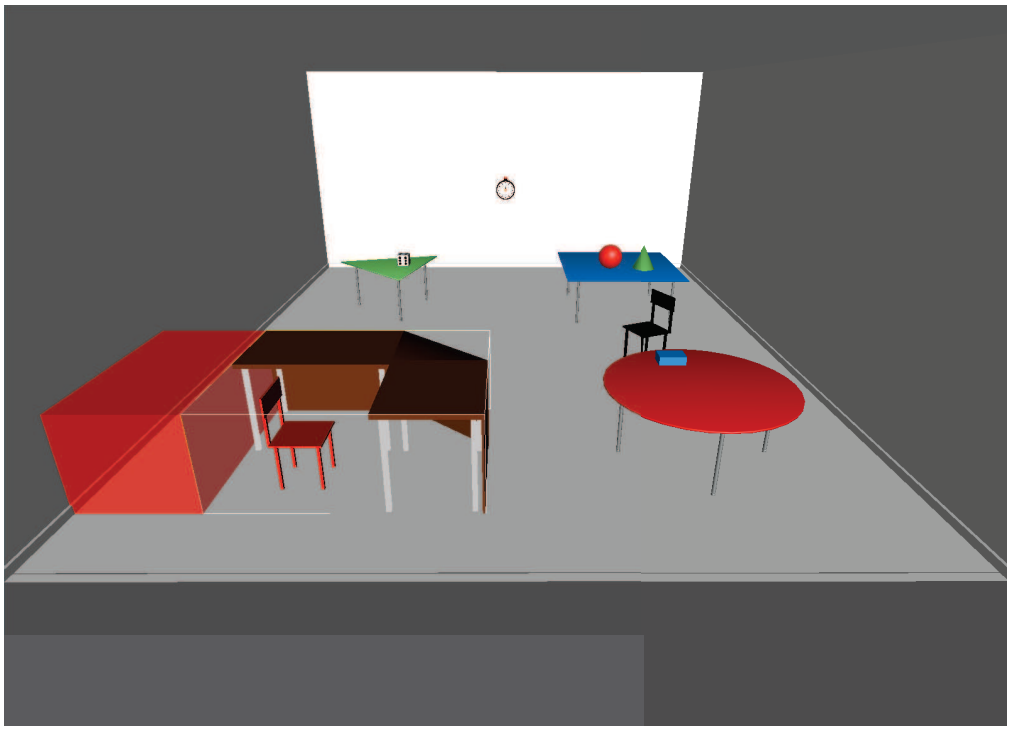
Description

To express direction form a third-person perspective, we use 27 directional operators. Our model of direction from a third-person perspective has been presented in Section 3.6.2.

Syntax



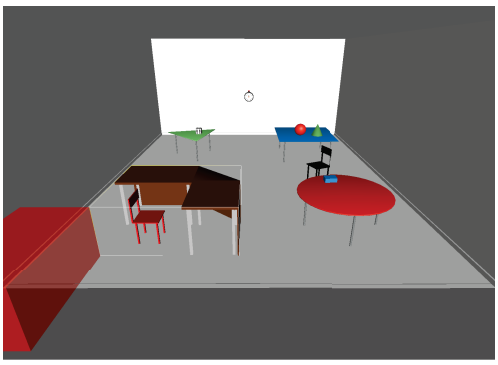
(a) The area “front” of the desk.



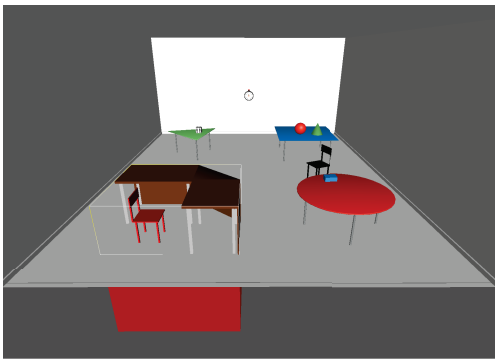
(b) The area “behind” the desk.

Figure 4.15: Semantic areas related to directional constraints.

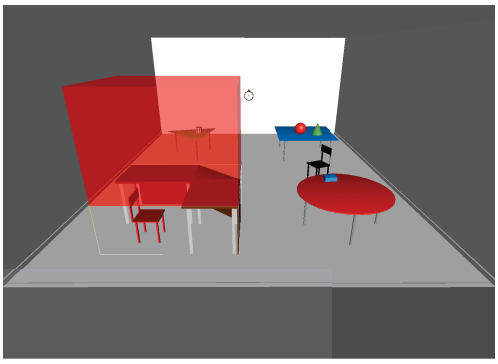
(c) The area "behind and left of" the desk.



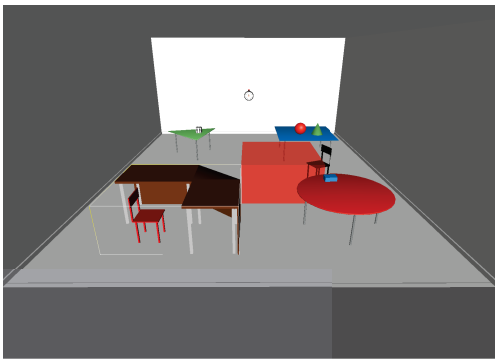
(d) The area "behind and right of" the desk.



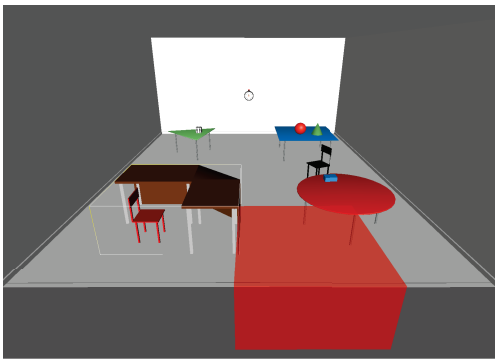
(e) The area "below" the desk.



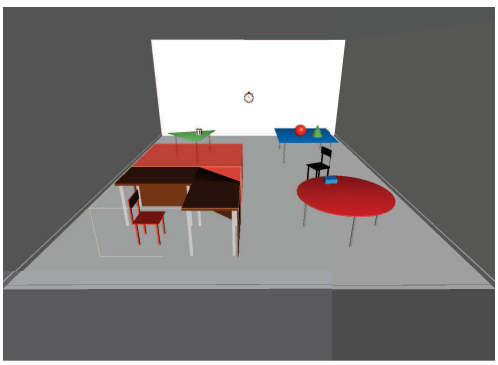
(f) The area "above" the desk.



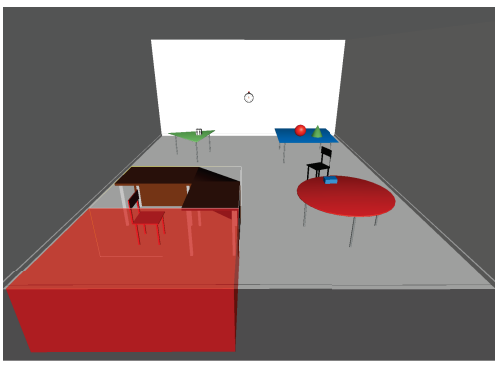
(g) The area "front and left of" the desk.



(h) The area "front and right of" the desk.



(i) The area "left of" the desk.



(j) The area "right of" the desk.

Figure 4.15: Semantic areas related to directional constraints.

Spatial relationships must be formalized in the context of a frame of reference as follows.

`po.rel_name(ro)@viewpoint(obs)`

where:

- `po`, `ro` are respectively the primary and reference spatial objects.
- `rel_name` is the name of the relationship. Note that our model of direction is consistent. Direction from third-person perspective shares the same granularity with direction from first-person perspective. The 27 possible directional relations are `isFrontOf`, `isLeftOf`, `isFrontLeftOf`, `isAboveFrontOf`, etc.
- `obs` stands for the viewer from which the relationship is seen.

Example 11. A directional constraint under a third-person perspective like Constraint 9 "from the viewpoint of the desk, the rectangular blue table must be on the left of the rounded red table" is expressed as follows.

```
context Room inv:
let _desk:Desk = Desk.allInstances()
      ->select(name = 'deskTable'),
    _blueTable:Table = Table.allInstances()
      ->select(name = 'blueTable'),
    _redTable:Table = Table.allInstances()
      ->select(name = 'redTable')
in _blueTable.leftOf(_redTable)@viewpoint(_desk)
```

In this example, we can see that the syntax of the original OCL was extended by the `@viewpoint` operator that allows the definition of a deictic FoR .

Semantic Areas

Similarly to direction from a first-person perspective, there are 27 semantic areas corresponding to 27 directional constraints from a third-person perspective.

4.6 Distance Operators

Description

Distance operators allow to express distance constraints in VEs. Our theoretical model of ternary orthographic projective constraints has been presented in Section 3.7.

Syntax

To express absolute distance that is directly computed between two spatial entities, the general form of VRX-OCL expressions is as follows:

`po.distanceTo(ro)` ;

where:

- `po` and `ro` are respectively the primary and reference object.
- `distanceTo` is the keyword used to express distance constraints.

Based on the absolute distance between the two spatial entities, it is possible to compute relative distance, i.e., to compare distances between two spatial entities to a third spatial entity. The comparison is purely qualitative. There are three possible qualitative constraints: "closer than", "equal to", and "further than". We do not formalise relative distance by specific operators. Instead, relative distance are expressed using comparative operators.

Example 12. Constraint 11 "in the room, the rounded red table must be closer than 1 meter to the desk" is expressed as follows.

```
context Room inv:
let _desk:Desk = Desk.allInstances()
        ->select(name = 'deskTable'),
    _redTable:Table = Table.allInstances()
        ->select(name = 'redTable')
in _redTable.distanceTo(_desk) < 1
```

Example 13. A relative distance constraint such as "in the room, the rounded red table is closer to the desk than to the rectangular blue table" is expressed as follows.

```
context Room inv:
let _desk:Desk = Desk.allInstances()
        ->select(name = 'deskTable'),
    _redTable:Table = Table.allInstances()
        ->select(name = 'redTable'),
    _blueTable:Table = Table.allInstances()
        ->select(name = 'blueTable')
in _redTable.distanceTo(_desk)
      < _blueTable.distanceTo(_desk)
```

Semantic Areas

With regard to distance constraints, our model allows to conceptualise and visualise the semantic area "with a distance *d*" around the reference object. Figure 4.15a provides an example of such semantic areas. The area "*1.5* meters" around the desk is visualised.

4.7 Summary

This chapter has presented the VRX-OCL language. VRX-OCL was defined as a spatial constraint and query language that enables the conceptualisation of

spatial constraints in VEs.

From the viewpoint of architecture and development process, the VRX-OCL language is positioned into the MASCARET framework. First, it uses a conceptual model of VE, produced by the MASCARET framework, as an input. Then, the language allows a high level, precise, and unambiguous specification of spatial constraints in the conceptual model. Semantics of the language is explained by its meta-model. VRX-OCL's meta-model covers new concepts that are necessary to represent spatial constraints. We described how the concepts of spatial entity and spatial constraint are formalised in the meta-model. Furthermore, we showed how spatial constraints can be used to specify behaviours in VEs.

Thereafter, we introduced the spatial syntax of VRX-OCL. VRX-OCL was defined as a spatial extension of the UML/OCL constraint language. VRX-OCL inherits the expressiveness of the OCL language, but it is extended with a novel syntax for modelling spatial constraints. Through a variety of examples, we showed that VRX-OCL is expressive enough to convey spatial knowledge in a formal and comprehensive way to users.

Part III

Applications

Chapter 5

Applications

The previous chapter introduced the VRX-OCL language. In this chapter, we describe real applications of this language for specifying spatial constraints in two different VEs. Section 5.1 shows how VRX-OCL is applied to define user's constraints in a VE for learning, named Virtual Physics Laboratory (VPLab). Section 5.2 describes the use of VRX-OCL in BRESTCOZ, a cultural heritage application.

Both VPLab and BRESTCOZ had been developed using MASCARET before VRX-OCL was integrated in. For each application, we first describe its current design solution and drawbacks, with regard to the representation of spatial constraints. Then, we discuss the benefits of VRX-OCL to conceptualise spatial constraints in each application. Section 5.3 summaries the chapter.

5.1 Virtual Physics Laboratory

5.1.1 Presentation

The Virtual Physics Laboratory (VPLab) is a VE for human learning dedicated to lab work in physics, particularly in optics, for students in the second year at university (Baudouin et al., 2008). Students have to learn how to measure the speed of light in different mediums, such as air, resin, and water. The goal of VPLab is twofold. First, its pedagogical objective is the learning of a measurement method, called *differential measurement* (Beney and Guinard, 2004). The method is detailed in the following paragraphs. Second, VPLab has been used as an experimental environment to analyse the usage of different virtual helps in VEs for learning (Baudouin et al., 2007).

Figure 5.1 shows the schema for measuring the speed of light in air. A light transmitter sends a visible light beam that is reflected by the mirror to return to a light receiver. An oscilloscope allows to measure the temporal interval between the emitted signal and the received signal. The real configuration of the setup is illustrated in Figure 5.2. The VPLab application is an adapted version of the real environment. Figure 5.3 provides an overview of the objects of the work surface with that the learner is supposed to interact. The application runs in

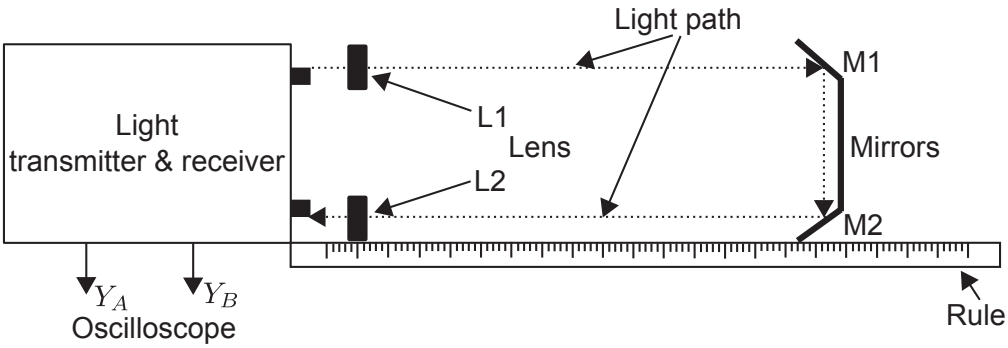


Figure 5.1: Schema for measuring the speed of light in air, according to (Beney and Guinard, 2004).

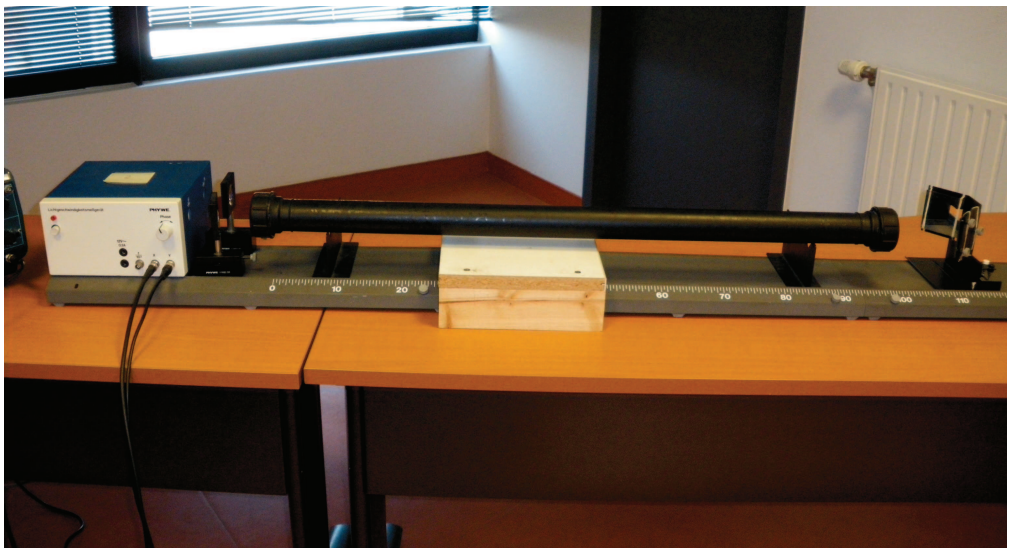


Figure 5.2: The real setup to perform the exercise of measuring the speed of light (here, through the tube of water).

different configurations. Figure 5.4 shows a configuration in which the learner can manipulate the objects using a haptic device.

Based on the schema, the difficulties in measuring the speed of light are as follows.

- The exact positions of the transmitter and the receiver are unknown
- The distance between the two mirrors $M1$ and $M2$ can not be measured accurately
- The zero time is unknown

Therefore, to measure the speed of light, the learners have no other choice but to apply a method, named *differential measurement*. One way to measure the speed of light in air is to put the mirror on an arbitrary position on the

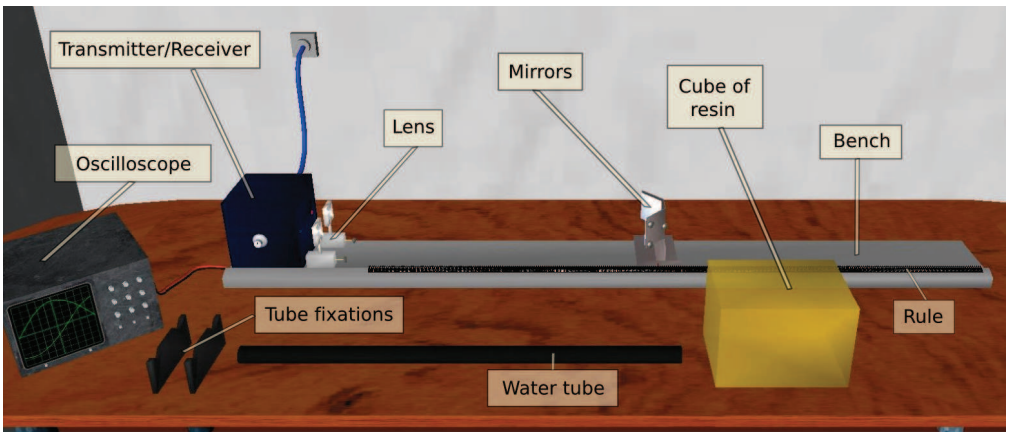


Figure 5.3: The work surface in the VPLab, a VE to perform the exercise of measuring the speed of light, according to (Baudouin et al., 2008) (labels were added to the figure for explanation).

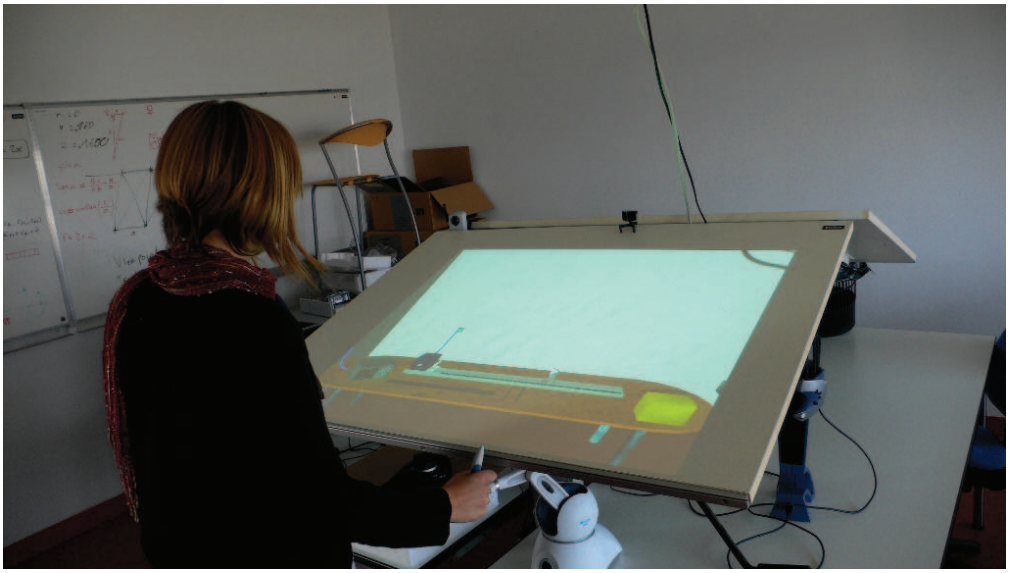


Figure 5.4: A user manipulates objects using an haptic device in a desktop mode.

bench, and then to note the phase difference t_1 displayed by the oscilloscope and the position P_1 of the mirror. Thereafter, the learners have to move the mirror, and then to note the new phase difference t_2 and the new position P_2 of the mirror. The speed of light is calculated by the formula $C = \frac{P_2 - P_1}{t_2 - t_1}$.

The method becomes more complicated for the other materials, such as resin or water. One solution is to place the mirror on an arbitrary position on the bench, and then to note the phase difference t_1. Thereafter, the learners have to place the material on the path of the light, and then to move the mirror with a length equal to the length of air replaced by the material, and then

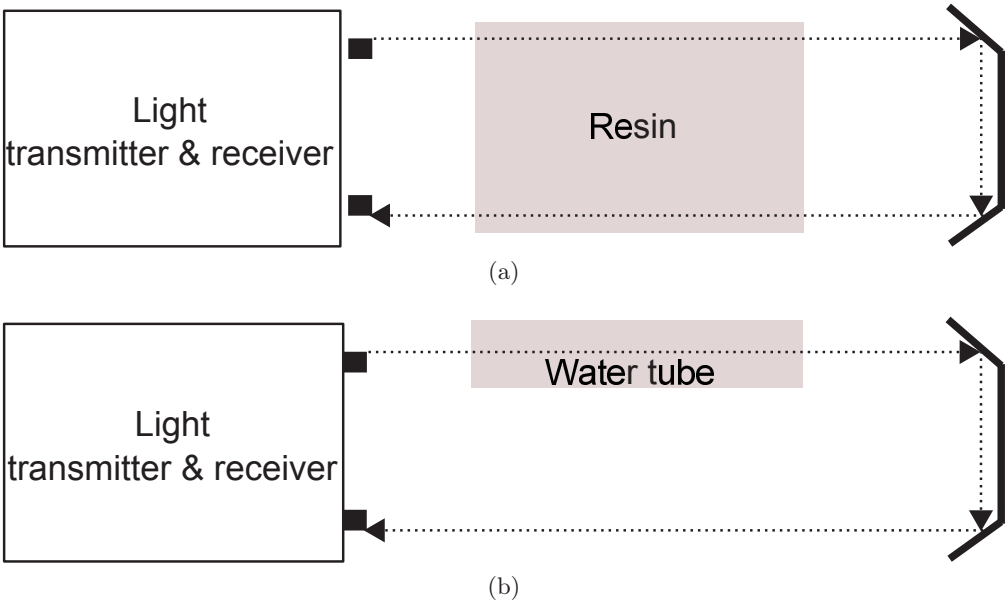


Figure 5.5: Schema for measuring the speed of light in several materials, such as (a) resin, (b) water, according to (Beney and Guinard, 2004).

to note the new phase difference t_2. The speed of light is calculated by the formula $C = \frac{l}{t_2 - t_1}$ where l is the length of the light path that travels through the material. Figure 5.5 illustrates the schema for measuring the speed of light in resin and water.

5.1.2 The Conceptual Model

VPLab has been designed using MASCARET. As for all VEs for learning, the two main models are as follows.

- Model of the learning environment: it includes the learning resources and the structure of the learning environment
- Model of exercises: it includes the procedures to realise the exercises, for example how to measure the speed of light in air

Modelling the Learning Environment

The learning environment is defined by all the objects and concepts that the learners have to manipulate to realise the exercises. Using MASCARET, the learning resources, as well as the structure of the learning environment, are modelled using class diagrams. Figure 5.6 illustrates the simplified class diagram that provides an overview of domain concepts of the VPLab. The structure of the learning environment is organised in the form of a Worksurface. The worksurface has a Bench, a Mirror, several Lens, and Mediums. Interestingly enough, in the

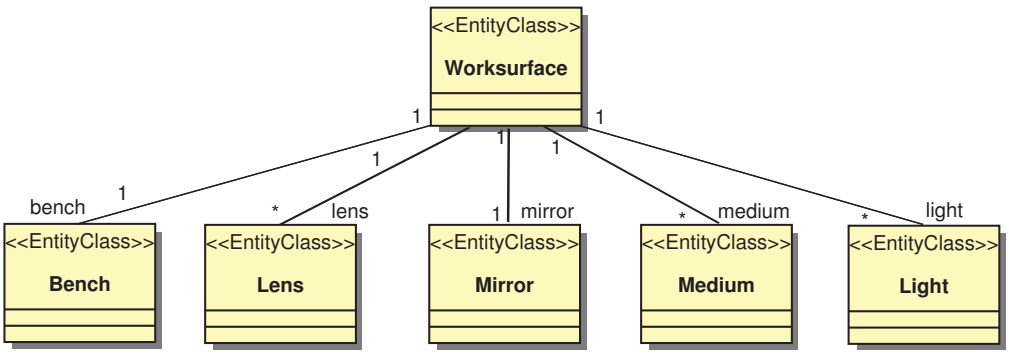


Figure 5.6: The simplified class diagram representing the work surface of the VPLab.

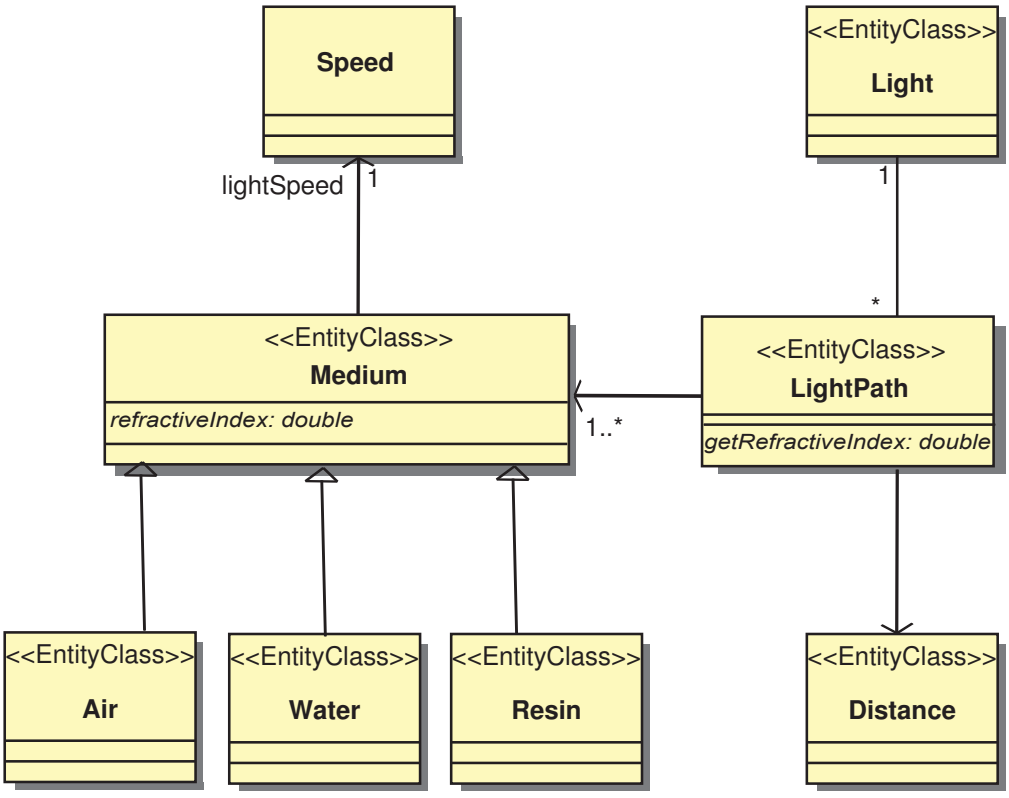


Figure 5.7: The class diagram representing the concept of light, speed of light, light path, and material in VPLab, adapted from (Marion, 2010).

conceptual model, even an abstract concept such as the light may be modelled by a class, named Light, whose instances are considered as virtual photons.

Figure 5.7 gives a more detailed description of the domain concepts in VPLab. Light is represented in the VE by its light path (the LightPath class). A light path travels through some materials (the Medium class). There are different types of material, such as Air, Resin or Water derived from the Medium class. Every mate-

rial has a refractive index (the *refractiveIndex* attribute) that impacts the speed of light of the material.

Modelling the Exercises

In VPLab, the learner follows a pre-defined procedure to realise an exercise. The procedure is described by an exercise statement in a textual format. Different exercise statements can be given by the teacher (Marion, 2010). For instance, the statement of the exercise for measuring the speed of light in a material is as follows.

1. Move the mirror to a position
2. Note the position P_1
3. Measure the phase difference t_1
4. Reset the phase difference of the oscilloscope
5. Put the material on the light path
6. Move the mirror to the position P_2
7. Measure the phase difference t_2
8. Compute the speed of light in the material

In VPLab, the learning procedures are specified using activity diagrams of Mascaret. Figure 5.8 shows the activity diagram representing the steps to measure the speed of light in a material. The last activity is not performed within the VE and thus is not mentioned in the diagram.

5.1.3 Specifying User Constraints Using VRX-OCL

The added values of Vrx-Ocl to the VPLab application are as follows:

- To specify spatial constraints that the learners must respect when realising exercises
- To offer additional virtual helps to the learners by visualising semantic areas in the learning environment

Specifying Spatial Constraints

There exists many spatial constraints that are implied when the learners interact with the learning environment. For example, in the procedure described in Figure 5.8, Constraint 15 is imposed when the learner performs Step 5 "*PutMaterialOnLightPath*".

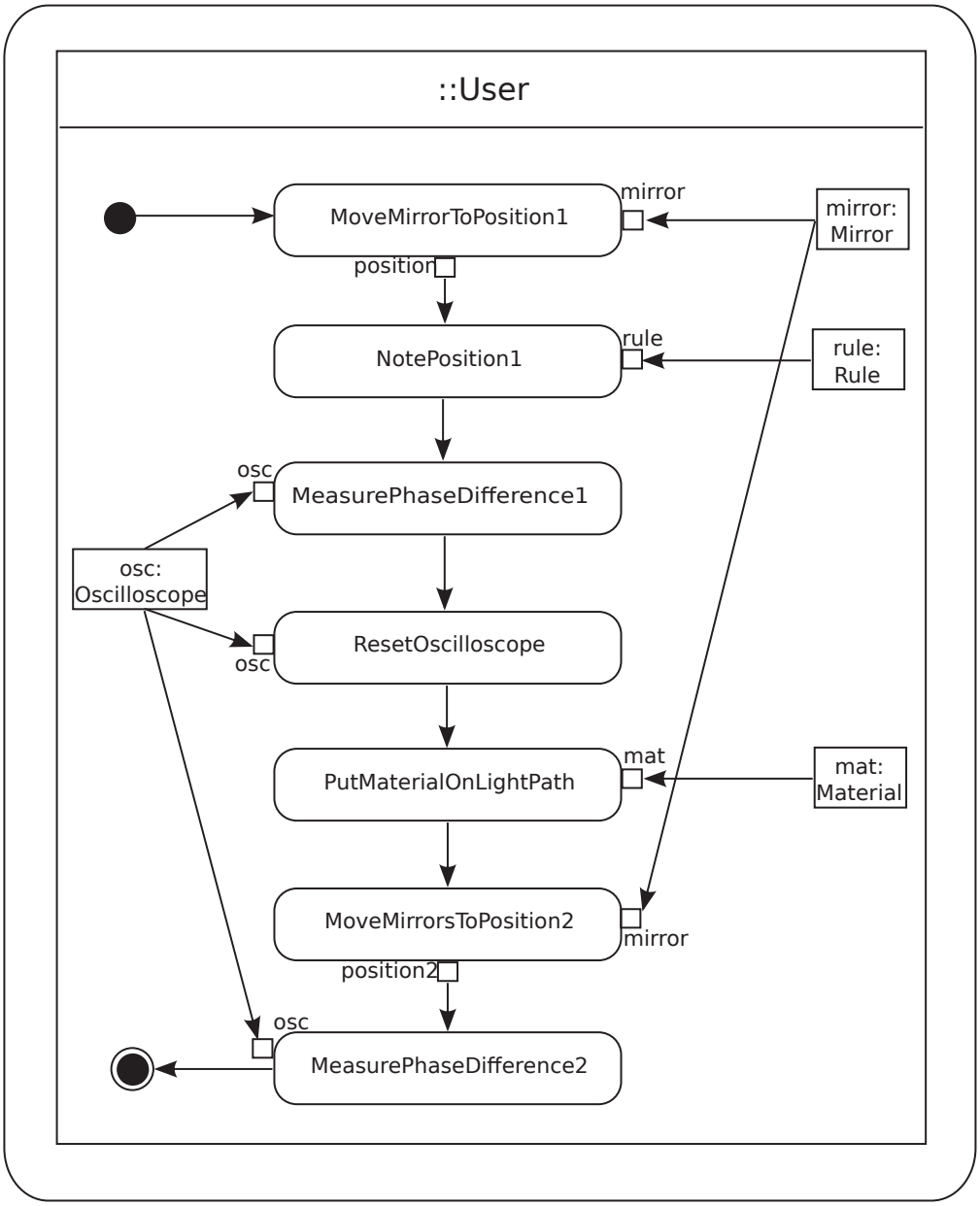


Figure 5.8: Procedure of measuring the speed of light in a material, represented in the form of an activity diagram, adapted from (Marion, 2010).

Constraint 15. *Put the cube of resin on the light path such that the cube is on the bench, between the lens and the mirror.*

Step 5 merely indicates an instruction that the learner has to follow. However, the expected result of this step (in other words, the *post-condition* of the learner's activity) is expressed in the form of several spatial constraints like "on" or "between". In the early version of VPLab, these spatial constraints were hard-coded in the controller of the user's interactions, using a specific al-

gorithm. Thanks to VRX-OCL, these spatial constraints can now be formally specified as the post-condition of the activity "*PutMaterialOnLightPath*" in the activity model. For readability, Constraint 15 is expressed in the two following examples.

Example 14. The first part of Constraint 15 "the resin cube should be placed on the bench" is expressed in VRX-OCL as follows.

```
context User::PutMaterialOnLightPath:Boolean post:
let resinCube:Resin = Resin.allInstances()
                ->select(name='resinCube'),
    bench:Bench = Bench.allInstances()
                ->select(name='bench')
in resinCube->meet(bench)
```

Note that, according to VRX-OCL's syntax, the `let..in` expression allows to define a variable that can be used in the constraint. Meanwhile, the `select()` function is used to specify a subset of a collection(e.g., selecting the medium named as "resinCube").

Example 15. The latter part of Constraint 15 "put the cube between the lens and the mirror" is expressed as follows.

```
context User::PutMaterialOnLightPath:Boolean post:
let resinCube:Resin = Resin.allInstances()
            ->select(name='resinCube'),
    tLens:Lens = Lens.lens()
            ->select(name='transmitterLens'),
    mirror:Mirror = Mirror.mirror()
            ->select(name='mirror')
in resinCube.between(tLens, mirror)
```

Both of the two examples are non-metric spatial constraints. The first constraint (i.e., "on" or "meet") stands for a topological constraint. The latter constraint is an example of ternary projective constraints that is based on the alignment between three objects: the lens, the cube, and the mirror. It can be alternatively formulated as "put the cube on the bench such that the mirror is after the lens and the cube". As we have seen previously, spatial constraints such as "between", "after" are called projective constraints because they are preserved under projective operations that maintain the collinearity of a set of points.

Within the VPLab, it is possible to provide some assistances to the learners. For example, it is possible to represent that the speed of light depends on the material it passes through. This abstract knowledge can be reified by graphically representing the light as virtual photons moving at a speed which depends on the object wherein they are located. Figure 5.9 illustrates such an assistance, called *Slow motion*. This assistance can be stated in form of constraints as follows.

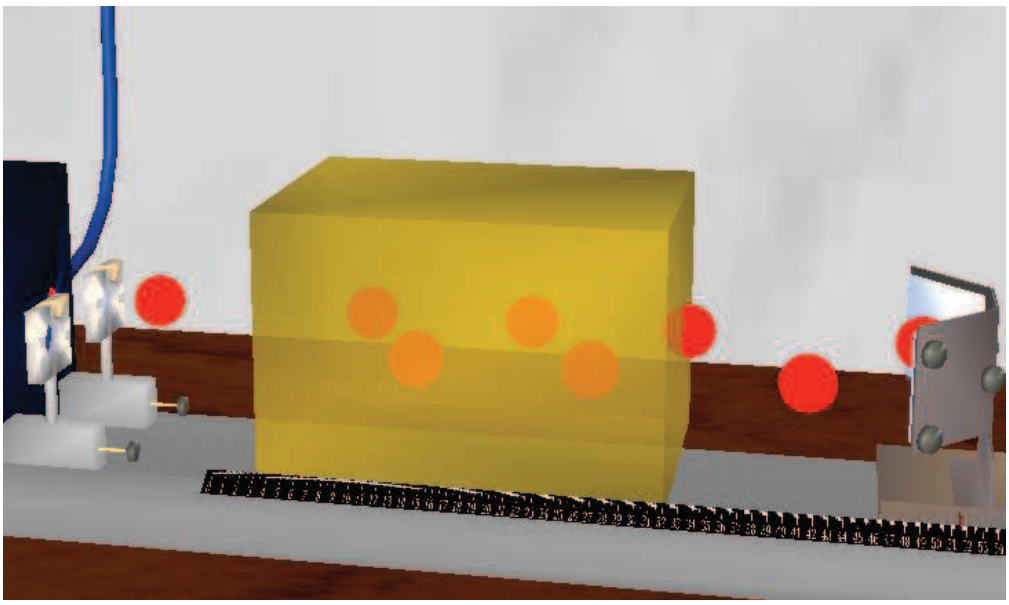


Figure 5.9: The effect *Slow motion* is simulated based on spatial constraints. The space between virtual photons is smaller in resin than in air.

Constraint 16. *When the Slow motion assistance is activated, if a virtual photon is inside an object, its speed is proportional to the refractive index of the material of this object, else its speed is proportional to the refractive index of air.*

In Constraint 16, it is necessary to handle within the same expression both object properties (e.g., the *refractiveIndex* attribute) and spatial relations (e.g., "inside"). The former can be expressed using standard OCL, but not the latter. As a spatial extension of OCL, VRX-OCL offers the ability to specify constraints related to properties as well as spatial relations between objects.

Example 16. Constraint 16 is expressed using VRX-OCL as follows.

```
context LightPath::getRefractiveIndex():Real body:
let r:Resin = Resin.allInstances()
              ->select(name='resinCube'),
    w:Water = Water.allInstances()
              ->select(name='waterTube'),
    a:Air = Air.allInstances()
              ->select(name='airEnv')
in if self.inside(r) then result = r.refractiveIndex
   else if self.inside(w) then result = w.refractiveIndex
   else result = a.refractiveIndex endif endif
```

In this example, the context of the constraint is the `LightPath` class. Each instance of the light path (represented by `self`) corresponds to a virtual photon.

The velocity of each photon is simulated based on the `getRefractiveIndex()` operation that is a query operation to get the refractive index of the material in which a photon travels. The `body` expression defines the content of the operation. The result of the operation (indicated by `result` keyword) is based on topological relations between the photons (`self`) and the materials.

Providing Additional Virtual Helps to the Learners

To facilitate the learning process, VPLab provides some virtual helps to the learners. According to (Baudouin et al., 2007), three types of virtual helps can be distinguished.

1. Show an invisible thing
2. Materialise an abstract concept
3. Make explicit how the exercise works

For example, the "*Slow Motion*" assistance illustrated previously in Figure 5.9 is an example of a virtual help of type " materialise an abstract concept". Meanwhile, to make explicit how the exercise works, it is possible to display the textual formula for computing the speed of light in the learning environment.

Based on spatial constraints, Vrx-Ocl allows to define additional virtual helps to the users. The main idea is to visualise semantic areas related to spatial constraints that the learners must satisfy. The visualisation of semantics areas is relevant to the definitions of two types of virtual helps: "show an invisible thing" and " materialise an abstract concept".

Figure 5.10 illustrates the semantic area related to the topological constraint as described in Constraint 15. The semantic area is the thick boundary of the cube of resin. The visualisation of the thick boundary is very useful, because it allows the learner to more precisely place the cube of resin on the bench. Similarly, Figure 5.11 shows the three virtual helps that visualise the semantic areas between the lens and the mirror.

5.2 BrestCoz

The previous section described VPLab as a relevant "in-door" environment for evaluating spatial constraints. In VPLab, we found mainly topological and projective constraints. In this section, we describe how we evaluate our model of spatial constraints and the Vrx-Ocl language in a large-scale environment, named BrestCoz.

The use of Vrx-Ocl in BrestCoz follows the same principle as described in VPLab. That is, we use the semantic model of spatial constraints to specify human-like activities of artificial agents and to visualise spatial constraints. Furthermore, BrestCoz is an interesting application for different types of spatial constraints, notably directional constraints.

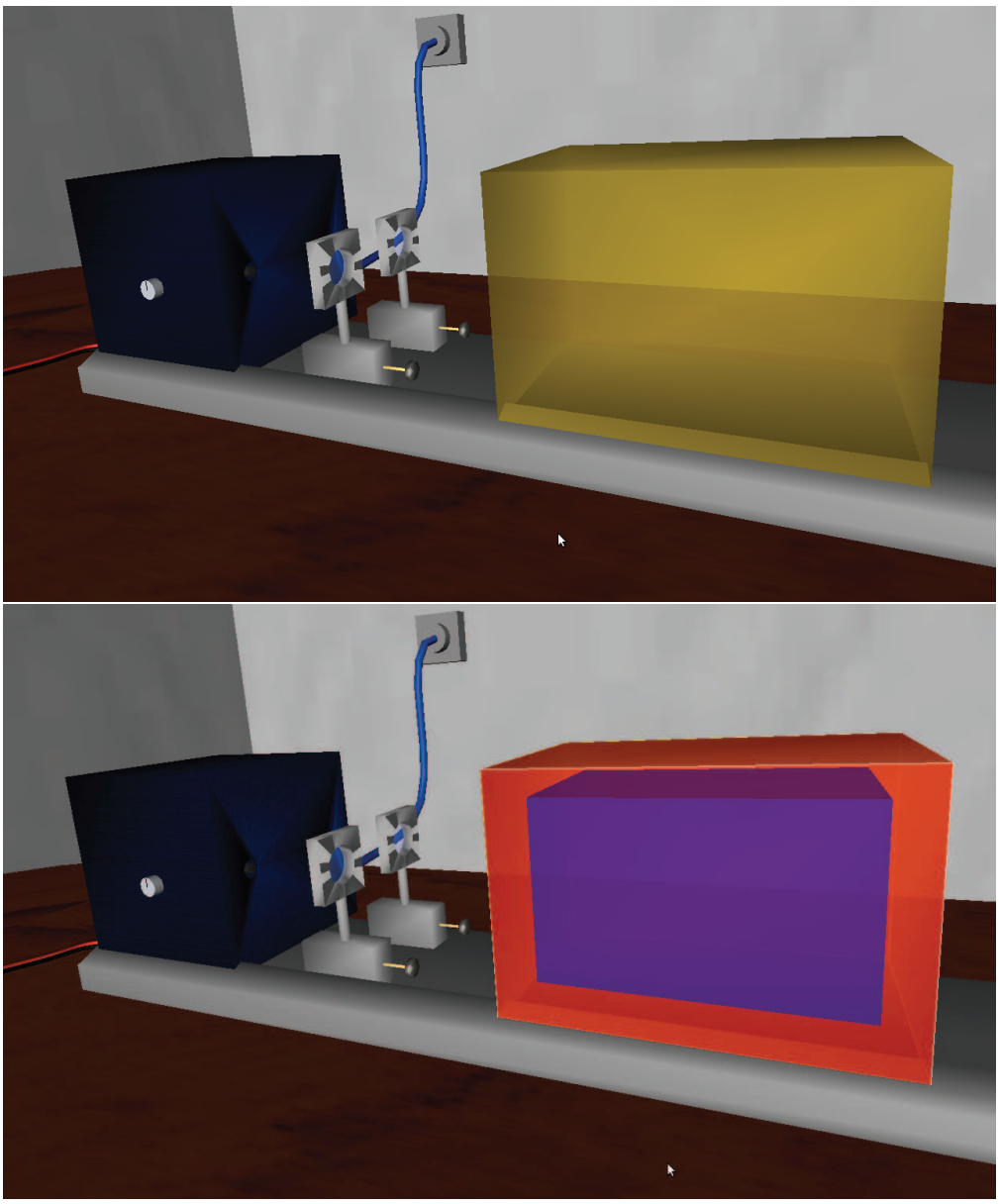
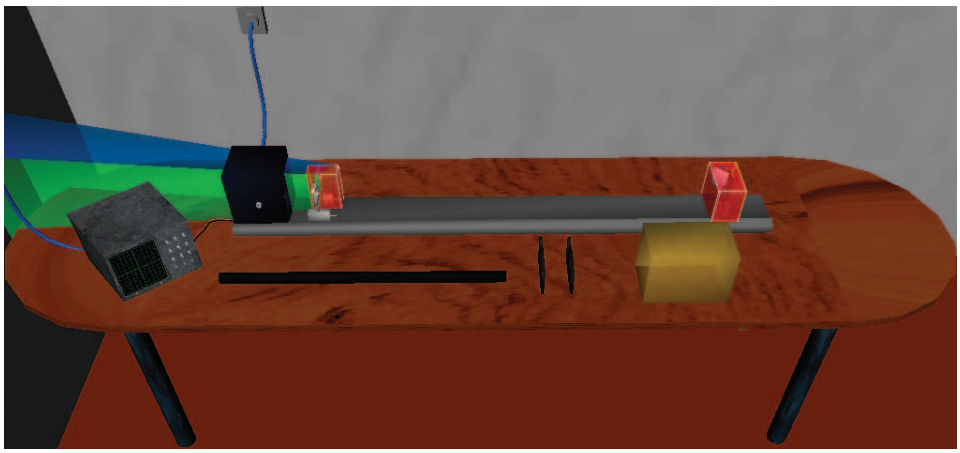


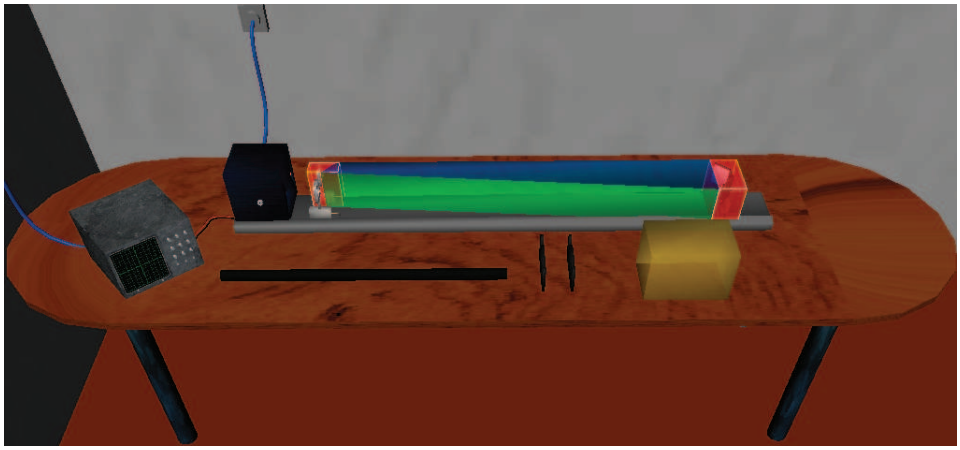
Figure 5.10: Visualisation of the cube of resin without (top figure) and with (bottom figure) its thick boundary.

5.2.1 Presentation

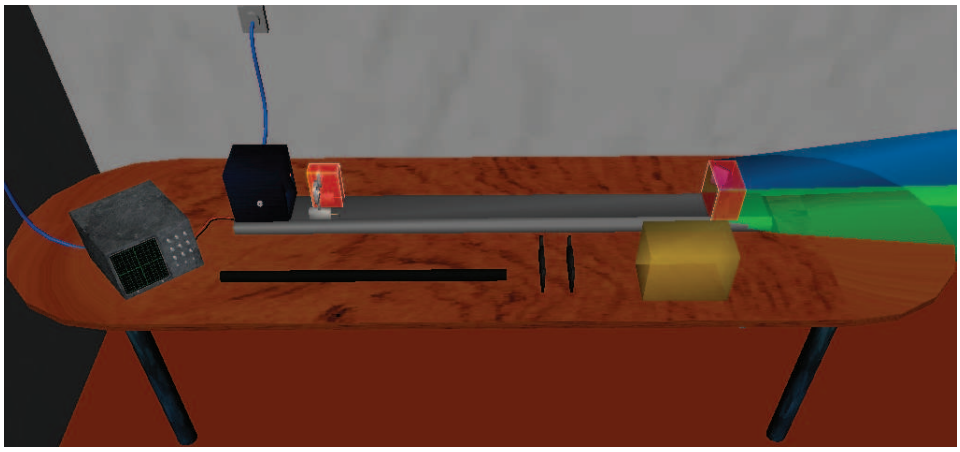
BrestCoz is a VE for visiting Brest harbour in the 18^{th} century. The Brest harbour was mostly destroyed due to heavy bombing during World War II. Using virtual reality techniques, BrestCoz aims to reconstruct the historical site of Brest harbour and thus allows one to visit and discover activities such as shipbuilding in the 18^{th} century. Figure 5.12 illustrates a view of the BrestCoz application.



(a) The zone *before* the lens and the mirror.



(b) The zone *between* the lens and the mirror.



(c) The zone *after* the lens and the mirror.

Figure 5.11: Visualisation of the acceptance sub-spaces between the transmitter lens and the mirror in the VPLab.

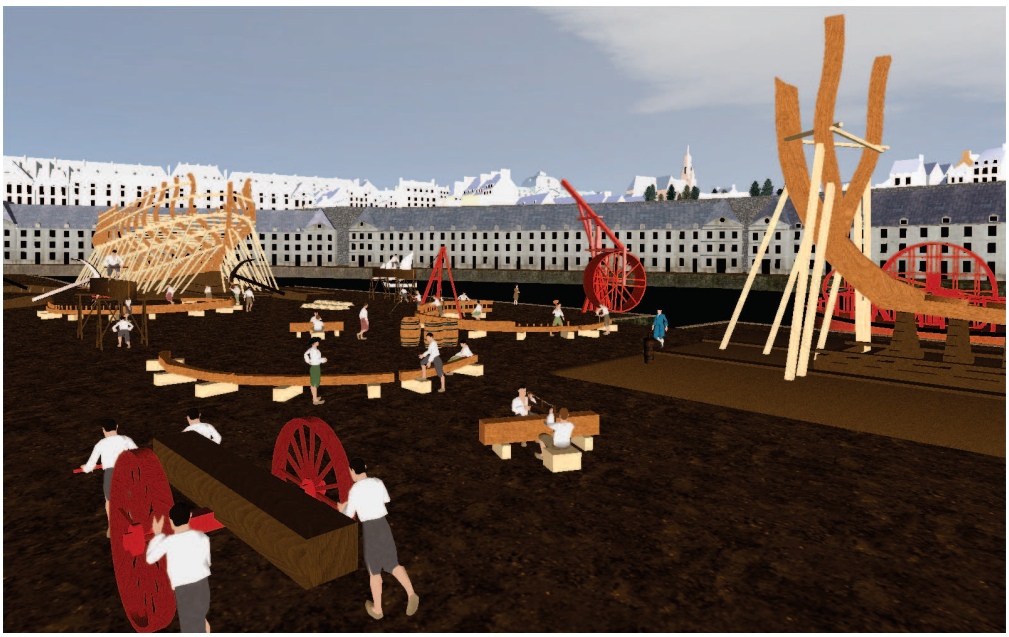


Figure 5.12: A view in the BrestCoz application.

The main purposes of BrestCoz are twofold:

- As a cultural heritage application, it is necessary to simulate domain activities (e.g., shipbuilding activities in the 18^{th} century)
- As a virtual visiting application, it is necessary to provide interactive guide to the visitors that allow to discover the domain concepts and human activities (e.g., how do workers build a ship? where is located a specific point of interest?)

5.2.2 The Conceptual Model

To achieve the above mentioned goals, BrestCoz has been designed using Mascaret framework. Thanks to its meta-modelling approach, involving a conceptual modelling phase, Mascaret offers the following advantages for the design of BrestCoz.

- First, Mascaret includes an explicit phase of domain modelling that allows domain experts (e.g., experts in naval activities), graphical designers, and software engineers to work together sharing their knowledge on a single model.
- Second, the meta-model of Mascaret allows to introspect the domain model at runtime. As a consequence, a virtual guide can be integrated into BrestCoz with the abilities to access to the content of the domain model and provide explanations to the visitors.

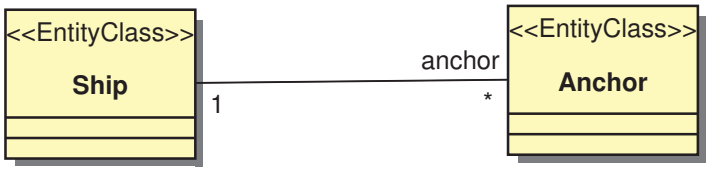
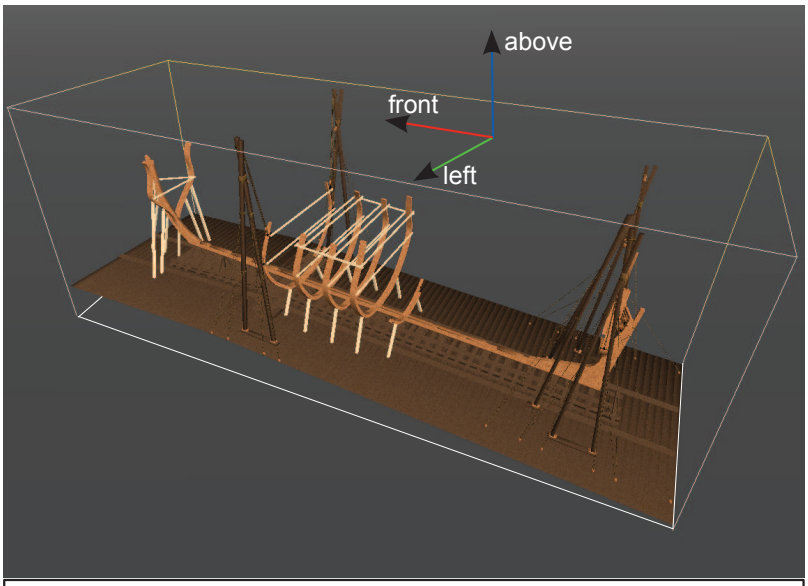


Figure 5.13: A partial simplified class diagram representing domain concepts such as ships, anchors, and their relations in BrestCoz, see (Barange et al., 2011) for a more detailed description of BrestCoz's domain model.




```
(1)<Entity class="Ship" name="ship1">
(2)  <Shape url="VRMLS/Ships/ship1.wrl"/>
(3)  <Position x="91.5075655701" y="30.1352004651" z="0"/>
(4)  <Rotation roll="0" pitch="0" yaw="-1.11529828327"/>
(5)  <IntrinsicDirection>
          < frontVector x="1" y="0" z="0"/>
          < leftVector  x="0" y="1" z="0"/>
          < aboveVector x="0" y="0" z="1"/>
     </IntrinsicDirection>
(6)</Entity>
```

Figure 5.14: Instantiation of a ship in BrestCoz.

The conceptual model of BrestCoz aims to specify domain concepts from the naval domain. For example, Figure 5.13 shows the simplified partial class diagram representing the concepts of ship and anchor. Conceptual relations between domain concepts are represented by associations between classes.

Once the domain model is defined, it is possible to instantiate a specific configuration of the VE. In the context of BrestCoz, different instance models can be defined. Each instance model corresponds to a specific configuration of the domain model. Figure 5.14 illustrates the instantiation of a ship - a spatial

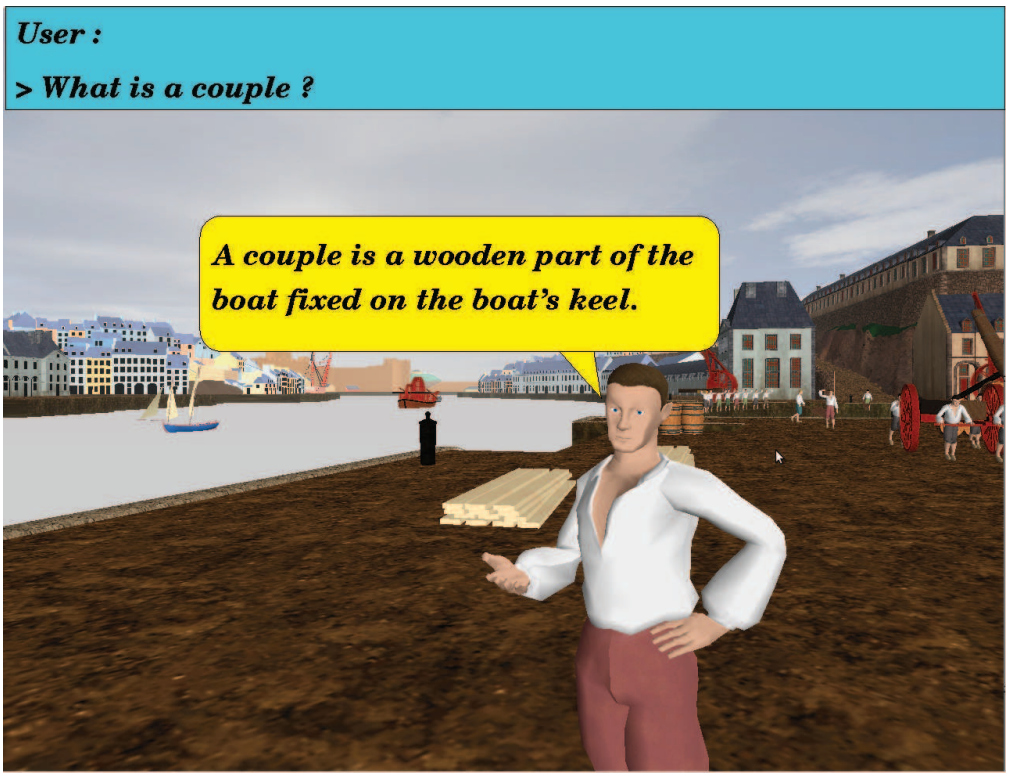


Figure 5.15: A virtual guide is integrated into BrestCoz providing explanations to the visitors, from (Barange et al., 2011).

entity instantiated from the `Ship` class. Here, a ship is defined not only with graphical properties such as shape, rotation, and position, but also with semantic spatial properties such as intrinsic direction.

To assist the visitors during their visit of the virtual harbour, we integrate into BrestCoz a virtual guide. The virtual guide is able to explain to users a specific domain concept such as "What is a keel in a ship?" or a domain activity such as "What a carpenter can do?". Such semantic explanations are possible thanks to the meta-model of Mascaret that allows a real-time reification and introspection of the domain model. Figure 5.15 illustrates the virtual guide in BrestCoz with the ability to communicate in natural language. A more detailed description of the domain model and the virtual guide of BrestCoz can be found in (Barange et al., 2011).

5.2.3 Modelling Human Activities Using Spatial Constraints

With regard to the purposes of BrestCoz application, the Vrx-Ocl language is applied in two different contexts:

- First, Vrx-Ocl is used as a constraint language that enables to specify constraints related to domain and human activities.

- Second, VRX-OCL is used to provide navigational aids to the visitors during their visit within the BRESTCOZ environment. In this situation, VRX-OCL enables the visualisation of semantic areas based on spatial constraints between spatial entities in the harbour.

Specifying Spatial Activities

An important issue in BRESTCOZ is related to the modelling of human activities. In general, human activities mainly include different tasks, such as to carry objects from one place to another, to meet other people, to search for a specific point of interest, or to operate tools within specific areas. Many activities are based on spatial constraints. Figure 5.16 exemplifies a human activity that is "A worker must move a trolley carrying timbers to the front of a ship".

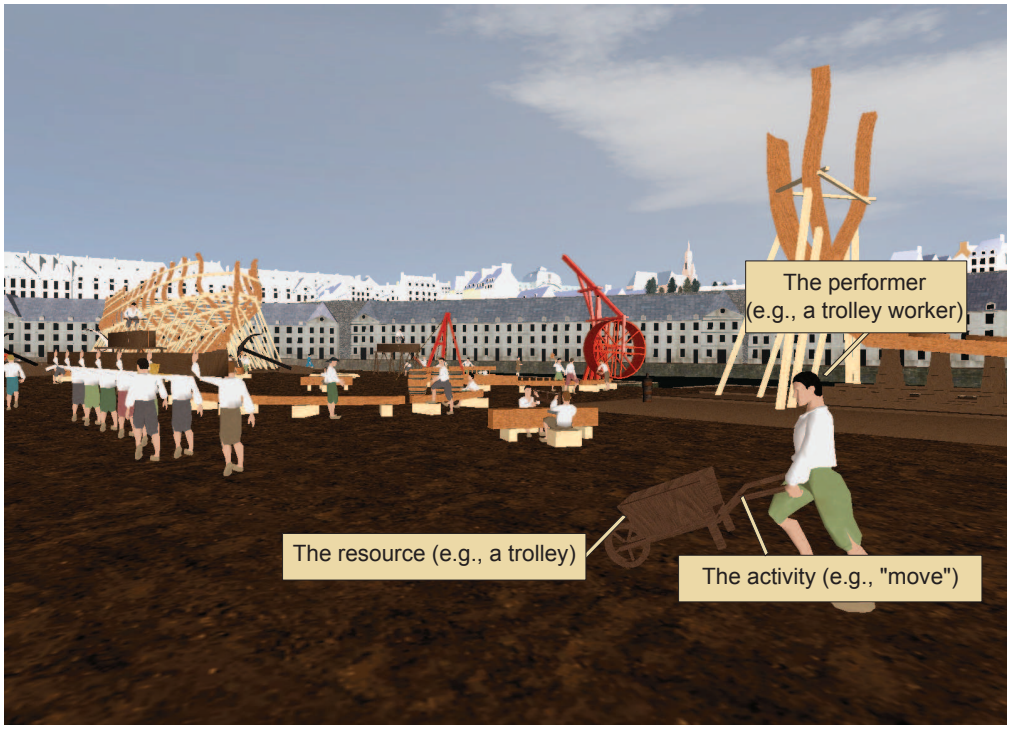


Figure 5.16: Example of an activity in the BRESTCOZ application.

In BRESTCOZ, human activities are conceptualised by means of activity diagrams. An activity diagram depicts the procedure to realise a task, with reference to resources required and roles (in a hierarchical organisation). Thereafter, human activities are simulated by artificial agents corresponding to the non-player characters. Many types of artificial agents are involved in BRESTCOZ, such as trolley workers, carpenters, sawers, or bearers. For example, Figure 5.17 shows the activity diagram corresponding to the human activity in Figure 5.16. This diagram allows to model the activity "carry", with the resources required as "trolley" and "timbers". It also defines "a trolley worker" as the performer of the

activity. However, the diagram was limited to specify spatial constraints such as "in front of a ship". In the initial implementation of BrestCoz, to simulate this human activity, the trajectory of agents was fixed in a configuration file that defines the target of the activity (i.e., the position in front of the ship).

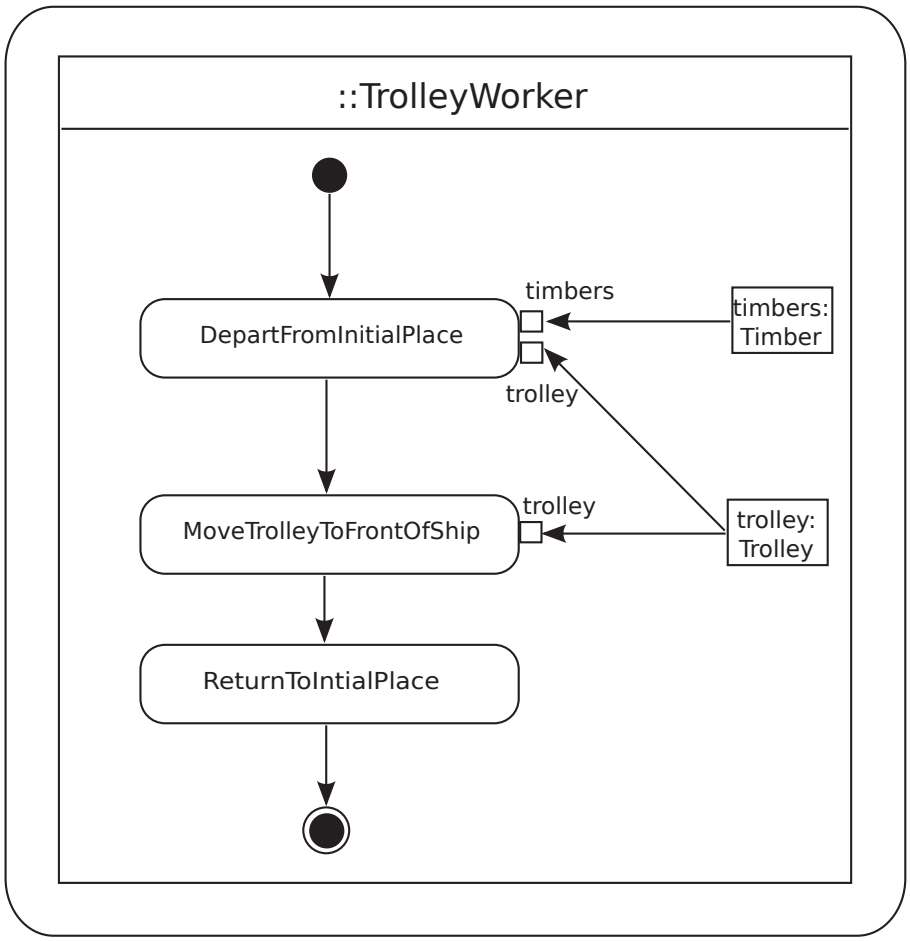


Figure 5.17: Activity diagram representing a human activity in BrestCoz.

To overcome this drawback, Vrx-Ocl allows to express spatial expressions like "in front of a ship" in the conceptual model of BrestCoz. Spatial constraints can be used in the activity diagram in different ways. One possible solution is composed of two steps.

1. to define the semantic area corresponding to the target (i.e., the area front of the ship)

2. to decide whether the worker is within the semantic area

For example, the semantic area "front of the ship" is defined in the conceptual model as follows.

<SemanticArea

```
        name = "areaFrontOfShip1"
        class = "FrontAreaClass"
        constraint = "frontOf"
        entity= "ship1"
        distance= 20
/>
```

Here, the target is defined as the area "front of the ship, named *ship1*". Furthermore, the area is limited by a distance of 20 meters from the ship. Once the semantic area corresponding to the target was defined, it can be used in the following VRX-OCL expression that specifies the post-condition of the activity *MoveTrolleyToFrontOfShip*.

```
context TrolleyWorker::MoveTrolleyToFrontOfShip:Boolean
post:
let target:FrontAreaClass = FrontAreaClass.allInstances()
          ->select(name = 'areaFrontOfTheShip1')
in self.inside(target)
```

The satisfaction of the post-condition allows to know whether the activity is accomplished or not. This example also shows a combination between topological and directional constraints. The activity *MoveTrolleyToFrontOfShip* of *TrolleyWorker* is considered as satisfied if the trolley worker is "inside" the area "front of" the ship.

Providing Navigational Helps to the Visitors

In BRESTCOZ, users are free to visit the environment and discover domain activities. During their visit, it is quite often that users need helps to better localise an item in the space that is in a spatial relation with other items. Figure 5.18 provides an example of a directional relation between an anchor and a ship, that is "the anchor is about 20m from the left of the ship". As we have seen previously, in BRESTCOZ, spatial constraints are expressed using VRX-OCL, and the semantic areas can be described in the conceptual model. Furthermore, when users can not localise the anchor, it is possible to visualise the semantic area that contains the anchor. For example, semantic areas such as "left", "right", or "behind" the ship can be visualised, as illustrated in Figure 5.19. Moreover, it is also possible to highlight the reference object (e.g., the ship), the primary object (i.e., the anchor), or the viewer.

5.3 Summary

In this chapter, we have presented the applications of VRX-OCL in two different VEs. The main goal of the language is to specify spatial constraints at the conceptual model of VEs.

First, the VRX-OCL language was applied to express spatial constraints in VPLab, a VE for learning of physics. We showed two situations in VPLab where spatial constraints are needed. On the one hand, VRX-OCL allows to express

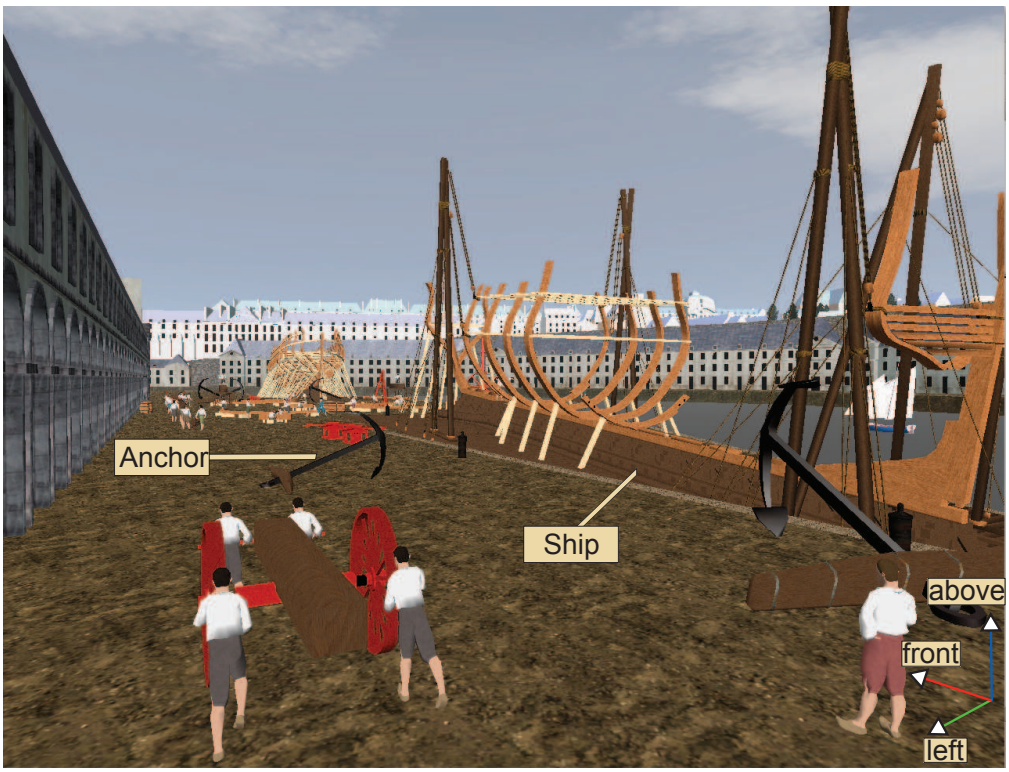


Figure 5.18: A view in BrestCoz application (labels were added to the figure for illustrating the directional relation between the anchor and the ship).

constraints on user's actions. These constraints must be respected when the learners manipulate objects to realise an exercise in the VE. On the other hand, the Vrx-Ocl language facilitates the specification of virtual assistances to the learners in the context of VEs for learning. Many educational assistances can be specified and visualised based on spatial constraints.

Second, Vrx-Ocl was applied to model activities of agents in BrestCoz, a VE for culture preservation. The motivation was that many activities of agents are based on spatial constraints. Furthermore, our model of spatial constraints was integrated into a virtual guide agent that provides navigational aids to the users.

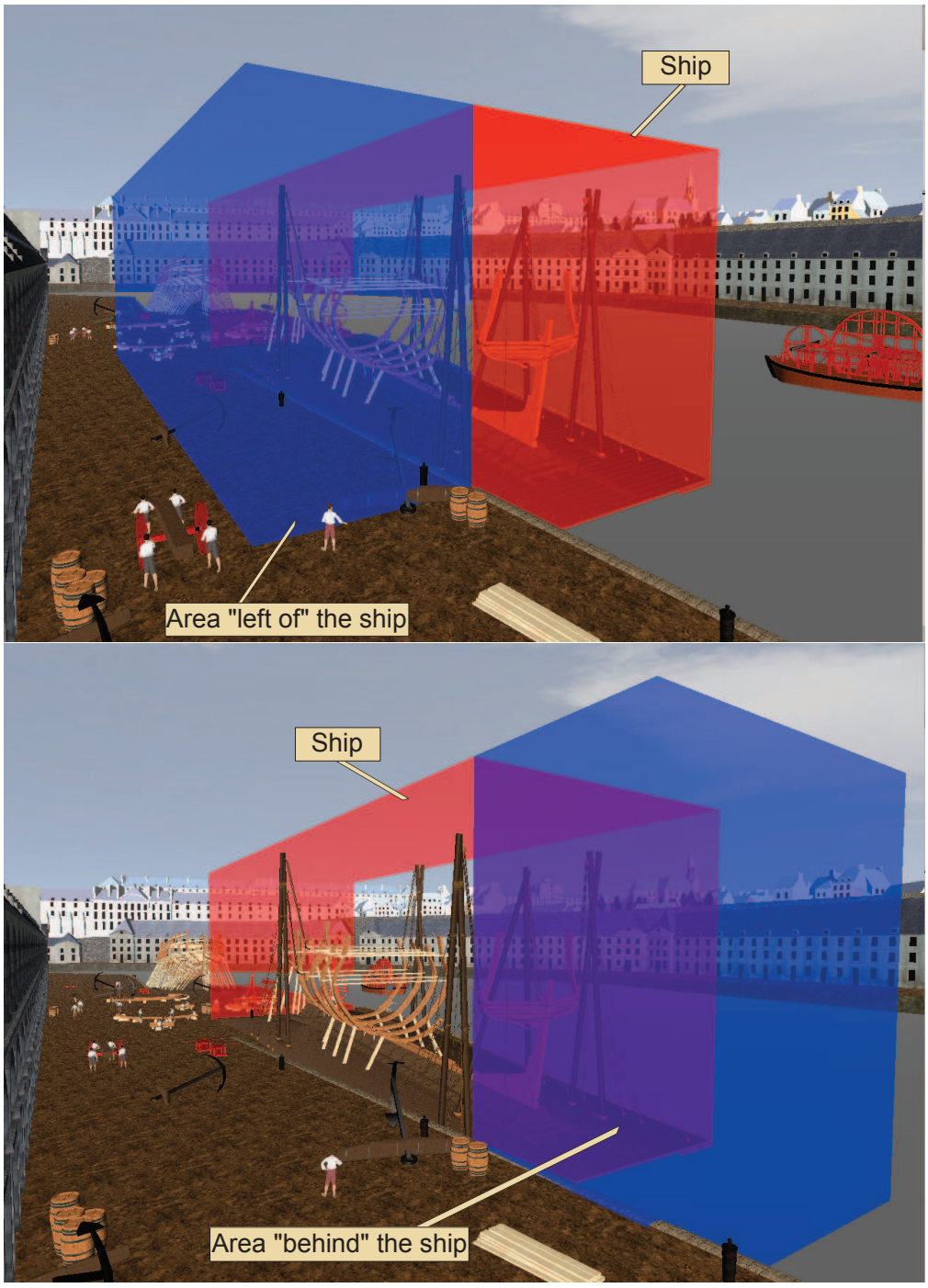


Figure 5.19: Visualising spatial constraints in BrestCoz. Top: the area "on the left and 20 meters" from the ship. Bottom: the area "behind and 20 meters" from the ship.

Conclusion

This thesis has proposed a model of spatial semantics of VEs. We studied spatial semantics from the point of view of spatial expressions that define spatial relationships among objects within VEs. In our approach, we represented spatial relations by means of spatial constraints. We formalised the VRX-OCL constraint language that allows to conceptualise spatial constraints in VEs using a model-based architecture. We argued that VRX-OCL is a relevant language for specifying constraints on artificial agents' and users' activities within VEs.

In this final chapter, Section 1 provides an overall summary of our approach. Section 2 discusses limitations of the proposed approach and gives suggestions for future work that could be carried out to extend this research.

1 Summary of Contributions

To model spatial semantics of VEs, we were inspired from semantic modelling, a novel paradigm for the design of semantic and intelligent VEs (Latoschik and Blach, 2008). In our approach, spatial semantics were introduced into the conceptual model of VEs during the modelling phase. As a result, our approach made spatial knowledge about VEs explicit and meaningful for users as well as artificial agents.

Modelling spatial knowledge of VEs has been recognised as a non-trivial task. The main difficulties are that, daily communication of spatial knowledge is very often given in an imprecise and incomplete manner. In addition, spatial knowledge of VEs is multi-dimensional and sometimes ambiguous. To fulfil the research objectives stated in the Introduction, our approach relied on the three main propositions.

- A generic and integrated model of spatial constraints for VEs
- A spatial language named VRX-OCL for specifying spatial constraints among objects at a conceptual level
- Practical validations and applications of the approach in real VEs

A Generic and Integrated Model of Spatial Constraints

Our first major contribution was a generic and integrated model of the main

families of spatial constraints. We tackled the issue of modelling spatial constraints from two aspects.

- Identifying the main families of spatial constraints. At the first stage, to model spatial constraints, it was crucial to understand what types of spatial relationships exist? How they are used in everyday communication? To answer to these questions, we reviewed the literature and then proposed a classification of spatial constraints. In our model, spatial constraints are metric or non-metric. Metric constraints include topological and projective constraints. Topological constraints are invariants under affine transformations, such as rotation, scaling, or translation. Meanwhile, projective constraints are based on the concept of collinearity and coplanarity that are invariants under projections. With regard to metric constraints, they are directional and distance constraints.

- Proposing a generic model of spatial constraints dedicated to VEs. At the second stage, based on our classification of spatial constraints, we proposed an integrated model of spatial constraints for VEs. Our model was built based on the notions of *spatial entity* and *frame of reference*. A spatial entity conceptually represents an object in a VE, but with additional spatial semantics, such as position, referential point, and intrinsic orientation. Based on the notion of spatial entity, we were able to model the main families of spatial constraint, namely topological, projective, directional, and distance constraints. Every spatial constraint is specified with a clear definition of frame of reference that can be implicit or explicit, first-person perspective or third-person perspective. Furthermore, the model was dedicated to VEs, taking into account the characteristics of VEs: interactive, real-time, and lack of precision in manipulating objects with interface devices.

The VRX-OCL Language For Conceptualising Spatial Constraints

To conceptualise spatial constraints in VEs, we proposed Vrx-Ocl, a Virtual Reality eXtension of Object Constraint Language. Our approach to conceptualising spatial constraints was composed of the two main steps.

- Defining the conceptual model of VEs. At this first step, to define the conceptual model of VEs, we were based on the Mascaret approach. Mascaret consists in using Uml as a sound basis for the semantic modelling of VEs. The main aspects of VEs, such as the structure of the environment or the behaviours of objects, are specified using Uml-Mascaret diagrams. Our contribution was that, we provided a meta-model such that the additional spatial concepts (e.g., spatial entity, frame of reference, and spatial constraint) could be covered in the conceptual model of VEs.

- Specifying spatial constraints using Vrx-Ocl: The language enabled the specification of spatial constraints in the conceptual model of VEs. We

defined VRX-OCL as an extension of OCL/UML. VRX-OCL is both a constraint and query language. As a constraint language, VRX-OCL allows to specify spatial constraints in a UML-MASCARET conceptual model using a textual and formal syntax. The syntax of the language was extended to cover specific concepts of spatial constraints, such as the concept of frame of reference. As a query language, VRX-OCL enables high-level queries for contents of the semantic conceptual model.

Applications of the Approach and the Language

To validate and show the feasibility of our approach, we applied the VRX-OCL language in the two real different applications: a learning environment and a cultural heritage application.

The first application named VPLab was a VE for learning of physics. In VPLab, to realise exercises, there are many spatial constraints that the learners have to follow. In the early version of the application, spatial constraints were hard-coded in the controller of the learner's interactions. Using the VRX-OCL language, it was possible to specify the user constraints at the conceptual model. In addition, we showed that VRX-OCL could be used to provide virtual helps to the learners.

The second application was BRESTCOZ, a cultural heritage application that provides virtual visits of Brest's harbour. The purpose of the visit was to allow users to discover domain activities, such as shipbuilding. In BRESTCOZ, the environment is populated by many types of virtual agent. Each type of agent simulates a specific domain activity. Agents mainly perform tasks such as to carry objects from one place to another, to meet other agents, or to operate tools within specific areas. Many activities of agents were based on spatial constraints. VRX-OCL was useful to specify spatial activities of agents. Additionally, the semantic areas introduced in VRX-OCL was used to enhance navigational aids to the visitors during their visit.

2 Limitations and Future Work

In this section, we discuss the limitations of our work and indicate several directions in which this work can be extended in the future. We group the discussion into categories as follows.

Limitations of the Proposed Model of Spatial Constraints For VEs

Everyday communication of spatial knowledge is mostly in a qualitative way. Therefore, an important point of our approach was to model spatial constraints in a qualitative manner. The qualitative approach offers a representation closer to how human communicates and reasons about spatial knowledge. In our model, spatial constraints were divided into two categories: metric and nonmetric. Our approach to nonmetric spatial constraints was purely qualitative, i.e., topological and projective constraints allow to represent relationships

among spatial entities without any metric information. With regard to metric constraints, (i.e., directional and distance constraints), we only proposed a qualitative representation of direction. Nevertheless, our model of distance still remains semi-qualitative. Only relative distance is qualitative, whereas absolute distance is represented in our model in a quantitative manner.

Distance has been an active subject for many studies in the field of VEs (Terziman et al., 2009; Grechkin et al., 2010). The common question is related to how human perceives and estimates distance within VEs (Plumert et al., 2005), whereas the question of how to model distance was neglected. Sometimes, current studies in distance perception in VEs lead to contradictory results, e.g., some studies stressed out the underestimation of distance within human in virtual and real environments, that did not hold in other studies (Interrante et al., 2006). Ideally, to represent and then reason about distance, distance should be qualitative, such as "near", "far" (Hernandez et al., 1995). However, this requires an additional formalisation of frame of reference (FoR) for distance, that is currently lacked in our model and thus needs further investigation. Currently, to model distance, we use canonical units based on the concepts of metre, kilometre, and so on. Whereas, the modelling of distance requires domain-oriented FoRs. For instance, distance is very often evaluated based on the number of bus stops or metro stations, or the travelling time and cost. FoRs of distance can also depend on internal properties of entities, such as size or shape. Recent work in the literature has been interested in modelling distance based on intrinsic properties of objects (e.g., size) but still far to fully support different types of FoR for distance (Bittner and Donnelly, 2007).

Limitations of the VRX-OCL Language

Based on OCL, VRX-OCL is very well suited for expressing local constraints on a specific element of a UML-based conceptual model, such as class invariants, pre- and post-conditions of operations, and guard conditions. The specific element is indicated by the context keyword. However, in our approach, we extended VRX-OCL to express global constraints related to the entire model. In this case, the VRX-OCL expressions mainly cover multi-classes and multi-instances constraints. As a consequent, the context of a constraint described by the reserved word self becomes restrictive. Furthermore, when a constraint involves different instances of many classes, the expression is overwhelmed with the let..in keywords that are used to declare variables in OCL. In the future, it would be valuable to extend the current spatial syntax of VRX-OCL such that allows to declare many contexts and many instances of a class in a simple expression. In the literature, a possible solution consists in using and referring instances in expressions by their identifiers that helps to avoid repeatedly writing long and complicated navigation paths (Casanova et al., 2000). Alternatively, an interface between natural language and spatial expressions can be introduced that allows to link natural language descriptions of spatial situations with spatial logical calculi (Hois and Kutz, 2008; Bateman et al., 2010). An approach for translating natural language constraints to the original OCL has been introduced in (Bajwa

et al., 2012). On the contrary, the specification of constraints using OCL can be translated to natural language (Burke and Johannisson, 2005). To make the link between the description of spatial constraints using natural language and VRX-OCL, a further investigation is needed.

Spatial Reasoning

Despite our model of spatial constraints is a sound basis for spatial reasoning, there currently lacks of a specific module dedicated to reasoning in our approach. As a consequence, it would also be of value to further investigate and incorporate into our model suitable spatial reasoning techniques that enable artificial agents to find out new relationships from existing ones. For example, given "the statue is in front of the picture" and "the table is in front of the statue", one can conclude that "the table is in front of the picture". Current reasoning models aim to reason about a specific aspect of space, e.g. direction (Wolter and Lee, 2010), or a combination of different aspects (Brageul and Guesgen, 2007). However, most of them have been shown to be limited in terms of complexity, even when objects are in 2D and simplified as points (Liu and Li, 2011). Spatial reasoning in real-time VEs needs specialised and adapted techniques, such that the reasoning result is achieved in an acceptable time. Furthermore, spatial reasoning should take into account specific concepts of VEs, such as the point of view of virtual agents, or the FoRs used as the contexts for reasoning (Brom et al., 2011).

Temporal Constraints and Reasoning

Beside spatial knowledge, the problem of representing temporal knowledge and temporal reasoning is critical in dynamic VEs. Temporal constraints have not yet been supported in VRX-OCL. A further extension of VRX-OCL should follow this direction. Interestingly enough, recent work has reconsigned OCL as one of the most promising approaches to represent temporal constraints (Soden and Eichler, 2009). Similar to spatial constraints, there exists two main approaches to temporal constraints: quantitative approaches and qualitative approaches. In quantitative approaches, temporal constraints are represented as precise points in time. Consequently, temporal constraints such as "a worker has to move a trolley periodically to the front of the ship at most 15 time units" can be expressed (Flake and Mueller, 2004). Whereas, qualitative approaches are mainly based on Allen's temporal intervals that describe relative order between timing events like "after", "before", "during", or "overlaps" (Allen, 1983). Based on basic temporal intervals, it is possible to construct more meaningful and complex temporal expressions, such as "always..until", "sometime..before", "always..since", or "sometime..since" (Ziemann and Gogolla, 2003).

Further Cognitive Validation

Finally, another research could be related to further applications and cognitive validations of the integration of semantic spatial relationships into VEs.

Taking a navigation aid system for example, such a future work might be related to empirical studies on how the visualisation of spatial constraints can help users in navigating in a large-scale VE. With regard to other application domains, a semantic model of spatial constraints could be used in other modules integrated into VEs that exhibit intelligent behaviours to users, such as explaining causal relationships (Aylett and Cavazza, 2001). Considering an intelligent tutoring system, we plan to use our proposed model of spatial constraints to detect complex errors realised by users in VEs and then explain to the users the reasons why the errors have occurred. An initial step to this direction was introduced in (Trinh et al., 2009).

Appendix A

Publications

The following papers concerning the VRX-OCL approach have been published:

1. T.-H. Trinh, P. Chevaillier, M. Barange, J. Soler, P. De Loor, and R. Querrec (2011). "Integrating semantic directional relationships into virtual environments: A meta-modelling approach". In *JVRC 2011: Proceedings of the Joint Virtual Reality Conference of EGVE – EuroVR*, pages 67–74, 20-21 September, Nottingham, UK, 2011.

2. M. Barange, P. De Loor, V. Louis, R. Querrec. J. Soler, T.-H. Trinh, E. Maisel, and P. Chevaillier (2011). "Get involved in an interactive virtual tour of Brest harbour: Follow the guide and participate". In *IVA 2011: Proceedings of the 11th International Conference on Intelligent Virtual Agents*, pages 93–99, vol. 6895 LNAI, 15-17 September, Reykjavík, Iceland, 2011.

3. P. Chevaillier, T.-H. Trinh, M. Barange, F. Devillers, J. Soler, P. De Loor, and R. Querrec (2011). "Semantic modelling of virtual environments using MASCARET". In *SEARIS 2011: Proceedings of the Fourth Workshop on Software Engineering and Architectures for Realtime Interactive Systems*, in conjunction with IEEE VR, 29 March, Singapore, 2011.

4. T.-H. Trinh, R. Querrec, P. De Loor, and P. Chevaillier (2010). "Specifying and dynamically visualizing semantic spatial constraints in a virtual environment for training using VRX-OCL". In *AFRV 2010: Actes des 5èmes Journées de l'Association Française de Réalité Virtuelle, Augmentée, Mixte et d'Interaction 3D*, pages 127–134, 6-8 December, Paris, France, 2010.

5. T.-H. Trinh, R. Querrec, P. De Loor, and P. Chevaillier (2010). "Ensuring semantic spatial constraints in virtual environments using UML/OCL". In *VRST 2010: Proceedings of the 17th ACM Symposium on Virtual Reality Software and Technology*, pages 219–226, 22-24 November, Hong Kong, China, 2010.

6. T.-H. Trinh, C. Buche, R. Querrec, and J. Tisseau (2009). "Modeling of errors realized by a human learner in virtual environment for training". *International Journal of Computers, Communications & Control*, IV(1):73–81, March 2009 (*special issue of selected and extended papers of ICVL 2008*).

7. T.-H. Trinh, C. Buche, and J. Tisseau (2008). "Modeling of errors realized by a human learner in virtual environment for training". In *ICVL 2008: Proceedings of the 3rd International Conference on Virtual Learning*, pages 71–80, 31 October - 02 November, Constanta, Romania, 2008.

Appendix B

VRX-OCL Grammar

This appendix presents the complete grammar of the VRX-OCL language implemented using the Boost Spirit[1] library.

```
oclFile = +(no_node_d[str_token("package")]
        >> root_node_d[packageName]
        >> oclExpressions
        >> no_node_d[str_token("endpackage")]);

packageName = pathName;

oclExpressions = *(constraint);

constraint = contextDeclaration
        >> (+(constraintDefinition
        |(stereotype >> !(NAME)>> COLON>> oclExpression)));

constraintDefinition = str_token("def") >> NAME
        >> !(formalParameterList) >> !(COLON >> typeSpecifier)
        >> EQUAL >> expression;

contextDeclaration = no_node_d[str_token("context")]
        >> (operationContext | classifierContext);

classifierContext = (NAME >> COLON >> NAME) | NAME;

operationContext = NAME >> DCOLON >> operationName
        >> LPAREN >> formalParameterList >> RPAREN
        >> !(COLON >> returnType);

stereotype = str_token("pre")| str_token("post")
        | str_token("inv");
```

[1] http://boost-spirit.com

```
operationName = NAME | EQUAL | NEQUAL| PLUS | MINUS
        | GE | LT | LE |  GT | DIVIDE | MULT
        | str_token("implies") | str_token("not")
        | str_token("or") | str_token("xor")
        | str_token("and");

formalParameterList = !(infix_node_d[formalParameter
        >> *(COMMA >> formalParameter)]);

formalParameter = NAME >> COLON >> typeSpecifier;

typeSpecifier = simpleTypeSpecifier | collectionType;

collectionType = collectionKind
        >> LPAREN >> simpleTypeSpecifier >> RPAREN;

oclExpression = !(*(letExpression )
        >> no_node_d[str_token("in")] ) >> expression;

returnType = typeSpecifier;

expression = logicalExpression;

ifExpression = discard_node_d[str_token("if")] >> expression
        >> discard_node_d[str_token("then")] >> expression
        >> discard_node_d[str_token("else")] >> expression
        >> no_node_d[str_token("endif")];

logicalExpression = relationExpression
        >> *(root_node_d[logicalOperator]
        >> relationExpression);

relationExpression = additiveExpression
        >> !(root_node_d[relationalOperator]
        >> additiveExpression);

additiveExpression = multiplicativeExpression
        >> *(root_node_d[addOperator]
        >> multiplicativeExpression);

multiplicativeExpression = unaryExpression
        >> *(root_node_d[multiplyOperator]
        >> unaryExpression);

unaryExpression = (unaryOperator >> postfixExpression)
```

```
        | postfixExpression;

postfixExpression = primaryExpression
        >> *((root_node_d[DOT]>> propertyCall)
        | (root_node_d[RARROW] >> collectionCall));

collectionCall = (iteratorCallName >> iteratorCallParameters)
        | (collectionMethodName >> propertyCallParameters);

iteratorCallName = str_token("exists") | str_token("forAll")
        | str_token("isUnique") | str_token("any")
        | str_token("one") | str_token("collect")
        | str_token("select") | str_token("reject")
        | str_token("sortBy") | str_token("iterate");

iteratorCallParameters = (LPAREN >> NAME >> *(COMMA >> NAME)
        >> !(COLON >> simpleTypeSpecifier)
        >> !(SEMICOL >> NAME >> COLON
                >> typeSpecifier >> EQUAL >> expression)
        >> BAR >> oclExpression >> discard_node_d[RPAREN])
        |(LPAREN >> oclExpression >> discard_node_d[RPAREN]);

collectionMethodName = str_token("isEmpty") | str_token("size")
        | str_token("notEmpty") | str_token("sum")
        | str_token("includes") | str_token("excludes")
        | str_token("count") | str_token("includesAll")
        | str_token("excludesAll") | str_token("product")
        | str_token("union") | str_token("intersection")
        | str_token("including") | str_token("excluding")
        | str_token("n") | str_token("symmetricDifference")
        | str_token("flatten") | str_token("asOrderedSet")
        | str_token("asSequence") | str_token("asBag")
        | str_token("append") | str_token("preprend")
        | str_token("insertAt") | str_token("subOrderedSet")
        | str_token("at") | str_token("indexOf")
        | str_token("first") | str_token("last")
        | str_token("subSequence");

primaryExpression = literalCollection | ifExpression
        | (STRING | NUMBER | leaf_node_d[
                (NAME >> DCOLON >> NAME >> *(DCOLON >> NAME))])
        | propertyCall |
                (inner_node_d[ LPAREN >> expression >> RPAREN]);

propertyCallParameters = LPAREN >>
        !( expression >>*(discard_node_d[COMMA] >> expression))
```

```
        >> RPAREN;

simpleTypeSpecifier = pathName;

literalCollection = collectionKind >> discard_node_d[LCURLY]
        >> !(collectionItem >> *(discard_node_d[COMMA]
        >> collectionItem))      >> discard_node_d[RCURLY];

collectionItem = expression >> !(DOTDOT >> expression);

propertyCall = specialOperationCall
        | spatialOperationCallWithFoR
        | spatialOperationCallWithoutFoR
        | operationCall | attributeCall;

specialOperationCall = (str_token("oclName")
        | str_token("oclIsTypeOf")
        | str_token("oclIsKindOf") | str_token("oclIsInState")
        | str_token("oclAsType") | str_token("allInstances"))
        >> propertyCallParameters;

spatialOperationCallWithoutFoR =
        ( str_token("distanceTo") | str_token("ray")
        | str_token("isInRay") | str_token("distance")
        | str_token("radar") | str_token("isInRadar")
        | str_token("dimensions")
        | str_token("isInDimensions")
        | str_token("disjoint") | str_token("meet")
        | str_token("equal") | str_token("inside")
        | str_token("coveredBy") | str_token("contains")
        | str_token("covers") | str_token("overlap")
        | str_token("asideOnPlane")
        | str_token("betweenOnPlane")
        | str_token("nonbetweenOnPlane")
        | str_token("beforeOnPlane")
        | str_token("afterOnPlane")
        | str_token("collinear") | str_token("noncollinear")
        | str_token("between")| str_token("nonbetween")
        | str_token("before")| str_token("after")
        |str_token("aside")      | str_token("outside")
        | str_token("coplanar") | str_token("noncoplanar")
        | str_token("internal") | str_token("external")
        | str_token("above")| str_token("below")
        ) >> propertyCallParameters;

spatialOperationCallWithFoR =
```

```
        ( str_token("leftsideOnPlane")
        | str_token("rightsideOnPlane")
        /* we enumerate here only 9 (of total 27)
        directional operators of the medium layer */
        | str_token("frontOf") | str_token("frontLeftOf")
        | str_token("frontRightOf") | str_token("leftOf")
        | str_token("rightOf") | str_token("neutralOf")
        | str_token("behindOf") | str_token("behindLeftOf")
        | str_token("behindRightOf")
        ) >> propertyCallParameters
        >> !(str_token("@viewpoint"))
        >> !(propertyCallParameters);

operationCall = NAME >> propertyCallParameters;

attributeCall = NAME >> !(qualifiers);

qualifiers = LBRACK >>  expression
        >> *(discard_node_d[COMMA] >> expression) >> RBRACK;

pathName = NAME >> *( DCOLON >> NAME );

timeExpression = ATSIGN >> str_token("pre");

actualParameterList = expression
        >> *(discard_node_d[COMMA] >> expression);

logicalOperator = str_token("and") | str_token("or")
        | str_token("xor") | str_token("implies");

collectionKind = str_token("Set") | str_token("Bag")
        | str_token("Sequence") | str_token("Collection")
        | str_token("OrderedSet");

relationalOperator = EQUAL | NEQUAL | GE | LE | GT | LT ;

ddOperator = PLUS | MINUS;

multiplyOperator = MULT | DIVIDE;

unaryOperator = MINUS | str_token("not");

LPAREN = ch_p('(');
RPAREN = ch_p(')');
LBRACK = ch_p('[');
RBRACK = ch_p(']');
```

```
LCURLY = ch_p('{');
RCURLY = ch_p('}');
COLON = ch_p(':');
DCOLON = str_token("::");
COMMA = ch_p(',');
EQUAL = ch_p('=');
NEQUAL = str_token("<>");
LT = ch_p('<');
GT = ch_p('>');
LE = str_token("<=");
GE = str_token(">=");
RARROW = str_token("->");
DOTDOT = str_token("..");
DOT = ch_p('.');
POUND = ch_p('#');
SEMICOL = ch_p(';');
BAR = ch_p('|');
ATSIGN = ch_p('@');
PLUS = ch_p('+');
MINUS = ch_p('-');
MULT = ch_p('*');
DIVIDE = ch_p('/');

NAME = no_node_d[*(str_token(" "))]
        >> leaf_node_d[lexeme_d[chset<>("a-zA-Z_")
        >> *(chset<>("a-zA-Z0-9_"))]]
        >> no_node_d[*(str_token(" "))];

NUMBER = access_node_d[real_p][assign_real()]
       | access_node_d[int_p][assign_int()];

STRING = access_node_d[
        leaf_node_d[lexeme_d[ch_p('\'')>>*(~(chset<>("'\n\r")))
        >>ch_p('\'')]]][assign_string()];
```

References

Akehurst, D. and Bordbar, B. (2001). On Querying UML data models with OCL. *«UML» 2001—The Unified Modeling Language. Modeling Languages, Concepts, and Tools*, pages 91–103. 109

Al-Khatib, W., Day, Y., Ghafoor, A., and Berra, P. (1999). Semantic modeling and knowledge representation in multimedia databases. *Knowledge and Data Engineering, IEEE Transactions on*, 11(1):64–80. 1

Allen, J. F. (1983). Maintaining knowledge about temporal intervals. *Commun. ACM*, 26(11):832–843. 54, 165

Aylett, R. and Cavazza, M. (2001). Intelligent virtual environments - A state-of-the-art report. In Duke, D. and Scopigno, R., editors, *STAR Proceedings of Eurographics 2001*, Manchester, UK. Eurographics Association. 2, 166

Aylett, R. and Luck, M. (2000). Applying artificial intelligence to virtual reality: Intelligent virtual environments. *Applied Artificial Intelligence*, 14(1):3–32. 2

Bajwa, I. S., Lee, M., and Bordbar, B. (2012). Translating natural language constraints to OCL. *Journal of King Saud University - Computer and Information Sciences*, (0):–. 164

Barange, M., De loor, P., Louis, V., Querrec, R., Soler, J., Trinh, T.-H., Maisel, E., and Chevaillier, P. (2011). Get involved in an interactive virtual tour of Brest harbour: Follow the guide and participate. In *Proceedings of the 11th International Conference on Intelligent Virtual Agents, IVA'11*, volume 6895 of *Lecture Notes in Artificial Intelligence*, pages 93–99, Reykjavick, Iceland. Springer-Verlag. 9, 31, 154, 155

Bateman, J. A., Hois, J., Ross, R., and Tenbrink, T. (2010). A linguistic ontology of space for natural language processing. *Artif. Intell.*, 174:1027–1071. 164

Baudouin, C., Beney, M., Chevaillier, P., and Le Pallec, A. (2007). Virtual helps usage in a virtual environment for learning. In *4th INTUITION International Conference and Workshop*, pages 52–60. ISBN 978-960-254-665-9. 141, 150

Baudouin, C., Chevaillier, P., Le Pallec, A., and Beney, M. (2008). Feedback on design and use of a virtual environment for practical lab work. In *Proceedings of the Virtual Reality International Conference, VRIC 2008*, pages 117–125. 141, 143

Bejaoui, L., Pinet, F., Bedard, Y., and Schneider, M. (2009). Qualified topological relations between spatial objects with possible vague shape. *International Journal of Geographical Information Science*, 23(7):877–921. 77

Beney, M. and Guinard, J. (2004). L'évaluation de l'efficacité du guidage dans les travaux pratiques de deug: un problème méthodologique complexe. *Didaskalia*, (24):29–64. 141, 142, 144

Ber, F. L. and Napoli, A. (2002). The design of an object-based system for representing and classifying spatial structures and relations. *Journal of Universal Computer Science*, 8(8):751–773. 54

Billen, R. and Clementini, E. (2006). Projective relations in a 3D environment. In Raubal, M., Miller, H. J., Frank, A. U., and Goodchild, M. F., editors, *GIScience*, volume 4197 of *Lecture Notes in Computer Science*, pages 18–32. Springer. 49, 82, 87, 89

Bittner, T. and Donnelly, M. (2007). A formal theory of qualitative size and distance relations between regions. In *Proceedings of the 21st International Workshop on Qualitative Reasoning*. 164

Borrmann, A. and Rank, E. (2009). Specification and implementation of directional operators in a 3d spatial query language for building information models. *Adv. Eng. Inform.*, 23(1):32–44. 51, 52, 55

Bouzeghoub, M. and Métais, E. (1991). Semantic modeling of object oriented databases. In Lohman, G. M., Sernadas, A., and Camps, R., editors, *17th International Conference on Very Large Data Bases*, pages 3–14, Barcelona, Catalonia, Spain. Morgan Kaufmann. 1

Bowman, D. A., North, C., Chen, J., Polys, N. F., Pyla, P. S., and Yilmaz, U. (2003). Information-rich virtual environments: theory, tools, and research agenda. In *VRST '03: Proceedings of the ACM symposium on Virtual reality software and technology*, pages 81–90, New York, NY, USA. ACM. 4, 17, 24, 25, 117

Brageul, D. and Guesgen, H. W. (2007). A model for qualitative spatial reasoning combining topology, orientation and distance. In *FLAIRS Conference, Proceedings of the Twentieth International Florida Artificial Intelligence Research Society Conference*, pages 653–658, Key West, Florida, USA. 165

Brom, C., Vyhnánek, J., Lukavský, J., Waller, D., and Kadlec, R. (2011). A computational model of the allocentric and egocentric spatial memory by means of virtual agents, or how simple virtual agents can help to build complex computational models. *Cognitive Systems Research*, (0):–. 165

Buche, C., Querrec, R., De Loor, P., and Chevaillier, P. (2004). MASCARET : A pedagogical multi-agent system for virtual environment for training. *International Journal of Distance Education Technologies (JDET)*, 2(4):41–61. ISSN 1539-3100. 36

Burke, D. and Johannisson, K. (2005). Translating Formal Software Specifications to Natural Language/A Grammar-Based Approach. *Logical Aspects of Computational Linguistics (LACL 2005)*, 3492:51–66. 165

Casanova, M., Wallet, T., and D'Hondt, M. (2000). Ensuring quality of geographic data with UML and OCL. In Evans, A., Kent, S., and Selic, B., editors, *UML2000 - The Unified Modeling Language, Advancing the Standard, Third International Conference, York, UK, October 2-6, 2000, Proceedings*, volume 1939 of *Lecture Notes in Computer Science*, pages 225–239. Springer. 55, 110, 164

Casati, R. and Varzi, A. C. (1997). Spatial entities. In Stock, O., editor, *Spatial and Temporal Reasoning*, pages 73–96. Kluwer Academic Publishers, Dordrecht. 6

Chen, J., Liu, D., Jia, H., and Zhang, C. (2007). Cardinal direction relations in 3D space. In Zhang, Z. and Siekmann, J. H., editors, *KSEM*, volume 4798 of *Lecture Notes in Computer Science*, pages 623–629. Springer. 51, 93

Chen, T. and Schneider, M. (2010). Modeling cardinal directions in the 3D space with the objects interaction cube matrix. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, SAC '10, pages 906–910, New York, NY, USA. ACM. 52

Chevaillier, P. (2006). *Les systèmes multi-agents pour les environnements virtuels de formation*. Habilitation à diriger des recherches, Université de Bretagne Occidentale, Brest. 31

Chevaillier, P., Querrec, R., and Septseault, C. (2009). VEHA, un métamodèle d'environnement virtuel informé et structuré. *Technique et Science Informatiques*, 28(6-7):715–740. 32, 38, 39, 40

Chevaillier, P., Trinh, T.-H., Barange, M., Devillers, F., Soler, J., De Loor, P., and Querrec, R. (2011). Semantic modelling of virtual environments using MASCARET. In *Proceedings of the Fourth Workshop on Software Engineering and Architectures for Realtime Interactive Systems SEARIS, in conjunction with IEEE VR 2011*, Singapore. 5, 8, 32, 36, 37

Clementini, E. and Billen, R. (2006). Modeling and computing ternary projective relations between regions. *Knowledge and Data Engineering, IEEE Transactions on*, 18(6):799–814. 49, 82, 84, 87

Clementini, E. and Di Felice, P. (1997). Approximate topological relations. *International Journal of Approximate Reasoning*, 16(2):173–204. 77

Cockcroft, S. (1997). A taxonomy of spatial data integrity constraints. *Geoinformatica*, 1(4):327–343. 44, 45

Cockcroft, S. (2004). The design and implementation of a repository for the management of spatial data integrity constraints. *GeoInformatica*, 8(1):49–69. 44

Cohn, A. and Gotts, N. (1996). The 'egg-yolk'representation of regions with indeterminate boundaries. *Geographic objects with indeterminate boundaries*, 2:171–187. 77

Cohn, A. and Renz, J. (2008). Qualitative spatial representation and reasoning. *Handbook of Knowledge Representation*, pages 551–596. 6, 44

Darken, R. and Peterson, B. (2001). Spatial orientation, wayfinding, and representation. *Handbook of virtual environment technology*, pages 1–22. 3

De Troyer, O., Kleinermann, F., Pellens, B., and Bille, W. (2007). Conceptual modeling for virtual reality. In *ER '07: Tutorials, posters, panels and industrial contributions at the 26th international conference on Conceptual modeling*, pages 3–18, Darlinghurst, Australia, Australia. Australian Computer Society, Inc. 16, 17

Donikian, S. (1997). Vuems: A virtual urban environment modeling system. In *Proceedings of the 1997 Conference on Computer Graphics International*, CGI '97, pages 84–, Washington, DC, USA. IEEE Computer Society. 15, 17

Duboisset, M., Pinet, F., Kang, M.-A., and Schneider, M. (2005). Precise modeling and verification of topological integrity constraints in spatial databases: From an expressive power study to code generation principles. In Delcambre, L. M. L., Kop, C., Mayr, H. C., Mylopoulos, J., and Pastor, O., editors, *ER*, volume 3716 of *Lecture Notes in Computer Science*, pages 465–482. Springer. 110

Durlach, N., Allen, G., Darken, R., Garnett, R., Loomis, J., Templeman, J., and Wiegand, T. (2000). Virtual environments and the enhancement of spatial behavior: Towards a comprehensive research agenda. *Presence: Teleoperators & Virtual Environments*, 9(6):593–615. 3

Egenhofer, M. (1994). Spatial SQL: A query and presentation language. *Knowledge and Data Engineering, IEEE Transactions on*, 6(1):86–95. 55

Egenhofer, M. (2002). Toward the semantic geospatial web. In *Proceedings of the 10th ACM international symposium on Advances in geographic information systems*, pages 1–4. ACM. 1

Egenhofer, M. and Franzosa, R. (1991). Point-set topological spatial relations. *Int'l. J. Geogr. Inf. Syst.*, 5(2):161–174. 75

Egenhofer, M. J. (1991). Reasoning about binary topological relations. In *SSD '91: Proceedings of the Second International Symposium on Advances in Spatial Databases*, pages 143–160, London, UK. Springer-Verlag. 47, 48

Farenc, N., Boulic, R., and Thalmann, D. (1999). An informed environment dedicated to the simulation of virtual humans in urban context. *Computer Graphics Forum*, 18(3):309–318. 15, 17, 18

Fernando, T., Murray, N., Tan, K., and Wimalaratne, P. (1999). Software architecture for a constraint-based virtual environment. In *Proceedings of the ACM symposium on Virtual reality software and technology*, pages 147–154. ACM. 56

Flake, S. and Mueller, W. (2004). Past- and future-oriented time-bounded temporal properties with OCL. In *SEFM '04: Proceedings of the Software Engineering and Formal Methods, Second International Conference*, pages 154–163, Washington, DC, USA. IEEE Computer Society. 165

Frank, A. (1992). Qualitative spatial reasoning about distances and directions in geographic space. *Journal of Visual Languages and Computing*, 3:343–343. 50, 93

Frank, A. U. (1996). Qualitative spatial reasoning: cardinal directions as an example. *International Journal of Geographical Information Science*, 10(3):269–290. 51

Freksa, C. (1992). Using orientation information for qualitative spatial reasoning. In Frank, A. U., Campari, I., and Formentini, U., editors, *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space. Intl. Conf. GIS—From Space to Territory*, volume 639 of *Lecture Notes in Computer Science*, pages 162–178, Berlin. Springer. 49, 50

Goyal, R. K. and Egenhofer, M. J. (2001). Similarity of cardinal directions. In *Proceedings of the 7th International Symposium on Advances in Spatial and Temporal Databases*, SSTD '01, pages 36–58, London, UK, UK. Springer-Verlag. 51

Grechkin, T. Y., Nguyen, T. D., Plumert, J. M., Cremer, J. F., and Kearney, J. K. (2010). How does presentation method and measurement protocol affect distance estimation in real and virtual environments? *ACM Trans. Appl. Percept.*, 7:26:1–26:18. 164

Grimaldo, F., Barber, F., and Lozano, M. (2006a). An ontology-based approach for IVE+VA. In *International Conference on Intelligent Virtual Environments and Virtual Agents (IVEVA)*, Aguascalientes (Mexico). 22, 23

Grimaldo, F., Barber, F., Lozano, M., and Orduña, J. M. (2006b). Semantic virtual environments for interactive planning agents. In *International Digital Games Conference (iDiG)*, Portalegre (Portugal). 17, 30, 31

Gutierrez, M., Vexo, F., and Thalmann, D. (2005). Semantics-based representation of virtual environments. *International Journal of Computer Applications in Technology*, 23(2):229–238. 22, 23, 30

Gutierrez, M. A. (2005). *Semantic Virtual Environments*. PhD thesis, Lausanne, EPFL. 2, 17, 22

Harrouet, F., Cazeaux, E., and Jourdan, T. (2006). Arevi. In Fuchs, P., Moreau, G., and Tisseau, J., editors, *Le traité de la Réalité Virtuelle*, volume 3, pages 369–392. Les Presses de lÉcole des Mines, 3ieme edition. 34

Hazarika, S. (2005). *Qualitative Spatial Change: Space-Time Histories and Continuity*. PhD thesis, The University of Leeds. 52

Hernández, D. (1994). *Qualitative Representation of Spatial Knowledge*, volume 804 of *Lecture Notes in Artificial Intelligence*. Springer, Berlin. 3, 73

Hernandez, D., Clementini, E., and Di Felice, P. (1995). Qualitative distances. *Lecture Notes in Computer Science*, 988:45–58. 52, 164

Heumer, G., Schilling, M., and Latoschik, M. (2005). Automatic data exchange and synchronization for knowledge-based intelligent virtual environments. In *Virtual Reality, 2005. Proceedings. VR 2005. IEEE*, pages 43–50. IEEE. 26

Hois, J. and Kutz, O. (2008). Natural Language Meets Spatial Calculi. In *Proceedings of the international conference on Spatial Cognition VI: Learning, Reasoning, and Talking about Space*, page 282. Springer-Verlag. 164

Ibáñez, J. and Delgado-Mata, C. (2011). Lessons from research on interaction with virtual environments. *J. Netw. Comput. Appl.*, 34:268–281. 2, 14

Ibáñez, J. and Delgado-Mata, C. (2006). A basic semantic common level for virtual environments. *IJVR - International Journal of Virtual Reality*, 5(3):25–32. 17, 19, 20, 30

Interrante, V., Ries, B., and Anderson, L. (2006). Distance perception in immersive virtual environments, revisited. In *VR '06: Proceedings of the IEEE conference on Virtual Reality*, pages 3–10, Washington, DC, USA. IEEE Computer Society. 164

Kallmann, M. and Thalmann, D. (1999). Direct 3D interaction with smart objects. In *Proceedings of the ACM symposium on Virtual reality software and technology*, pages 124–130. ACM. 17, 30, 70

Kalogerakis, E., Christodoulakis, S., and Moumoutzis, N. (2006). Coupling ontologies with graphics content for knowledge driven visualization. *Virtual Reality Conference, IEEE*, 0:43–50. 15, 17, 20, 21, 30, 31

Kleinermann, F., De Troyer, O., Mansouri, H., Romero, R., Pellens, B., and Bille, W. (2005). Designing semantic virtual reality applications. In *Proceedings of the 2nd INTUITION International Workshop, Senlis, France*, pages 5–10. 26, 27, 30, 31

Kleinermann, F., Mansouri, H., De Troyer, O., Pellens, B., and Ibáñez-Martínez, J. (2008). Designing and Using Semantic Virtual Environment over the Web. *International Journal of Virtual Reality*, 7(3):53–58. 4

Kosters, G., Pagel, B., and Six, H. (1997). Gis-application development with geoooa. *International Journal of Geographical Information Science*, 11(4):307–335. 53

Latoschik, M. and Schilling, M. (2003). Incorporating VR databases into AI knowledge representations: A framework for intelligent graphics applications. In *Proceedings of the Sixth IASTED International Conference on Computer Graphics and Imaging*, pages 79–84. 26

Latoschik, M. E. and Blach, R. (2008). Semantic modelling for virtual worlds: A novel paradigm for realtime interactive systems? In *VRST '08: Proceedings of the 2008 ACM symposium on Virtual reality software and technology*, pages 17–20, New York, NY, USA. ACM. 1, 2, 161

Latoschik, M. E. and Fröhlich, C. (2007). Towards intelligent VR - multi-layered semantic reflection for intelligent virtual environments. In Braz, J., Vázquez, P.-P., and Pereira, J. M., editors, *GRAPP (AS/IE)*, pages 249–260. INSTICC - Institute for Systems and Technologies of Information, Control and Communication. 16, 17, 25, 30

Lee, T., Hendler, J., Lassila, O., et al. (2001). The semantic web. *Scientific American*, 284(5):34–43. 1

Liu, W. and Li, S. (2011). Reasoning about cardinal directions between extended objects: The NP-hardness result. *Artificial Intelligence*, 175(18):2155–2169. 165

Liu, Y., Wang, X., Jin, X., and Wu, L. (2005). On internal cardinal direction relations. In Cohn, A. G. and Mark, D. M., editors, *COSIT*, volume 3693 of *Lecture Notes in Computer Science*, pages 283–299. Springer. 52

Liu, Y., Xu, C., Pan, Z., and Pan, Y. (2006). Semantic modeling for ancient architecture of digital heritage. *Computers & Graphics*, 30(5):800 – 814. 15, 17, 19

Louwsma, J., Zlatanova, S., Lammeren, R., and Oosterom, P. (2006). Specifying and implementing constraints in GIS–with examples from a geo-virtual reality system. *Geoinformatica*, 10(4):531–550. 45, 47

Manjunath, B., Salembier, P., and Sikora, T. (2002). *Introduction to MPEG-7: multimedia content description interface*, volume 1. John Wiley & Sons Inc. 22

Marion, N. (2010). *Modélisation de scénarios pédagogiques pour les environnements de réalité virtuelle d'apprentissage humain*. PhD thesis, Université de Bretagne Occidentale. 145, 146, 147

Marion, N., Septseault, C., Boudinot, A., and Querrec, R. (2007). GASPAR : Aviation management on an aircraft carrier using virtual reality. In *Proc. Int'l Conf. on Cyberworlds 2007*, pages 15–22. 31

MOF, O. M. G. (2011). MetaObject Facility (MOF) Core Specification, formal/06-01-01. www.uml.org/mof. 35

OMG (2006). Unified Modeling Language 2.0 Object Constraint Language Specification, formal/06-05-01. http://www.omg.org/spec/OCL/2.0/. 5, 32, 109

OMG, O. M. G. (2011). Model-driven architecture. http://www.omg.org/mda/. 5, 13, 32

Otto, K. (2005a). Semantic virtual environments. In *Special interest tracks and posters of the 14th international conference on World Wide Web*, page 1037. ACM. 21

Otto, K. (2005b). The semantics of multi-user virtual environments. In *Workshop Towards Semantic Virtual Environments (SVE'05)*. 21, 31

Pacheco, J., Escrig, M. T., and Toledo, F. (2002). Qualitative spatial reasoning on three-dimensional orientation point objects. In Agell, N. and Ortega, J. A., editors, *16th International WorkShop on Qualitative Reasoning (QR'02)*, pages 113–124, Barcelona, Spain. 50

Papadias, D. and Theodoridis, Y. (1997). Spatial relations, minimum bounding rectangles, and spatial data structures. *International Journal of Geographical Information Science*, 11(2):111–138. 70

Parent, C., Spaccapietra, S., and Zimányi, E. (1999). Spatio-temporal conceptual models: data structures + space + time. In *GIS '99: Proceedings of the 7th ACM international symposium on Advances in geographic information systems*, pages 26–33, New York, NY, USA. ACM. 54

Pellens, B., De Troyer, O., Bille, W., and Kleinermann, F. (2005). Conceptual modeling of object behavior in a virtual environment. In *Proceedings of Virtual Concept*, pages 93–94. 27

Pinet, F., Duboisset, M., and Soulignac, V. (2007). Using UML and OCL to maintain the consistency of spatial data in environmental information systems. *Environmental Modelling and Software*, 22(8):1217–1220. 55

Pinet, F., Kang, M.-A., and Vigier, F. (2004). Spatial constraint modelling with a GIS extension of UML and OCL: Application to agricultural information systems. In *Metainformatics*, pages 160–178. 54

Plumert, J. M., Kearney, J. K., Cremer, J. F., and Recker, K. (2005). Distance perception in real and virtual environments. *ACM Trans. Appl. Percept.*, 2:216–233. 164

Polys, N. and Bowman, D. (2004). Design and display of enhancing information in desktop information-rich virtual environments: challenges and techniques. *Virtual Reality*, 8(1):41–54. 25, 30

Querrec, R. (2002). *Les Systèmes Multi-Agents pour les Environnements Virtuels de Formation : Application à la sécurité civile.* PhD thesis, Université de Bretagne Occidentale, Brest (France). 31

Querrec, R. (2010). *Apprentissage de procédures en environnements virtuels.* Habilitation à diriger des recherches, Université de Bretagne Occidentale, Brest, France. 31, 32

Querrec, R., Buche, C., Lecorre, F., and Harrouet, F. (2011). Agent metamodel for virtual reality applications. In Ryzko, D., Rybinski, H., Gawrysiak, P., and Kryszkiewicz, M., editors, *Emerging Intelligent Technologies in Industry, 19th International Symposium, ISMIS 2011, Warsaw, Poland, June 28-30, 2011. Proceedings of the Industrial Session*, volume 369 of *Studies in Computational Intelligence*, pages 81–90. Springer. 36, 38

Querrec, R., Buche, C., Maffre, E., and Chevaillier, P. (2004). Multiagents systems for virtual environment for training. application to fire-fighting. *International Journal of Computers and Applications (IJCA)*, 1(1):25–34. ISSN 1710-2251. 36

Randell, D. A., Cui, Z., and Cohn, A. G. (1992). A spatial logic based on regions and connection. In *Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning (KR'92)*, pages 165–176. 47, 48, 75

Renz, J. and Nebel, B. (2007). Qualitative spatial reasoning using constraint calculi. In *Handbook of Spatial Logics*, pages 161–215. Springer Netherlands. 5

Rishe, N. (1992). *Database design: the semantic modeling approach.* McGraw-Hill Companies. 1

Salamin, P., Thalmann, D., and Vexo, F. (2006). The benefits of third-person perspective in virtual and augmented reality? In *VRST '06: Proceedings of the ACM symposium on Virtual reality software and technology*, pages 27–30, New York, NY, USA. ACM. 72

Salamin, P., Thalmann, D., and Vexo, F. (2009). Intelligent switch: An algorithm to provide the best third-person perspective in augmented reality. In *Proceedings of the 22nd Annual Conference on Computer Animation and Social Agents (CASA 2009)*, Amsterdam, the Netherlands. 73

Salehi, M., Bédard, Y., Mostafavi, M. A., and Brodeur, J. (2007). On languages for the specification of integrity constraints in spatial conceptual models. In Hainaut, J.-L., Rundensteiner, E. A., Kirchberg, M., Bertolotto, M., Brochhausen, M., Chen, Y.-P. P., Cherfi, S. S.-S., Doerr, M., Han, H., Hartmann, S., Parsons, J., Poels, G., Rolland, C., Trujillo, J., Yu, E. S. K., and Zimányi, E., editors, *Advances in Conceptual Modeling - Foundations and Applications, ER 2007 Workshops CMLSA, FP-UML, ONISW, QoIS,*

RIGiM,SeCoGIS, Auckland, New Zealand, November 5-9, 2007, Proceedings, volume 4802 of *Lecture Notes in Computer Science*, pages 388–397. Springer. 54

Schafer, W. and Bowman, D. (2004). Evaluating the effects of frame of reference on spatial collaboration using desktop collaborative virtual environments. *Virtual Reality*, 7(3):164–174. 73

Septseault, C. (2007). *Représentation d'environnements virtuels informés et de leur dynamique par un personnage autonome en vue d'une crédibilité comportementale*. PhD thesis, Université de Bretagne Occidentale. 31

Shekhar, S., Liu, X., and Chawla, S. (1999). An object model of direction and its implications. *Geoinformatica*, 3(4):357–379. 95

Skiadopoulos, S., Giannoukos, C., Sarkas, N., Vassiliadis, P., Sellis, T., and Koubarakis, M. (2005). Computing and managing cardinal direction relations. *IEEE Trans. on Knowl. and Data Eng.*, 17(12):1610–1623. 51

Smith, G. and Stuerzlinger, W. (2001). Integration of constraints into a VR environment. In *VRIC '01: Proceedings of the Virtual Reality International Conference 2001*, pages 103–110. 57

Smith, S. P. and Willans, J. S. (2006). Virtual object specification for usable virtual environments. In *OZCHI '06: Proceedings of the 18th Australia conference on Computer-Human Interaction*, pages 183–190, New York, NY, USA. ACM. 57

Soden, M. and Eichler, H. (2009). Temporal extensions of OCL revisited. In *ECMDA-FA '09: Proceedings of the 5th European Conference on Model Driven Architecture - Foundations and Applications*, pages 190–205, Berlin, Heidelberg. Springer-Verlag. 55, 110, 165

Stuerzlinger, W. and Smith, G. (2002). Efficient manipulation of object groups in virtual environments. In *VR '02: Proceedings of the IEEE Virtual Reality Conference 2002*, page 251, Washington, DC, USA. IEEE Computer Society. 57

Terziman, L., Lecuyer, A., Hillaire, S., and Wiener, J. M. (2009). Can camera motions improve the perception of traveled distance in virtual environments? In *VR '09: Proceedings of the 2009 IEEE Virtual Reality Conference*, pages 131–134, Washington, DC, USA. IEEE Computer Society. 164

Trinh, T. H., Buche, C., Querrec, R., and Tisseau, J. (2009). Modeling of errors realized by a human learner in virtual environment for training. *International Journal of Computers, Communications & Control*, IV(1):73–81. 9, 166

Trinh, T. H., Buche, C., and Tisseau, J. (2008). Modeling of errors realized by a human learner in virtual environment for training. In *3th International Conference on Virtual Learning - ICVL'08*, pages 71–80, Constanta (Romania). 9

Trinh, T.-H., Chevaillier, P., Barange, M., Soler, J., Loor, P. D., and Querrec, R. (2011). Integrating semantic directional relationships into virtual environments: A meta-modelling approach. In Coquillart, S., Steed, A., and Welch, G., editors, *JVRC11: Joint Virtual Reality Conference of EGVE - EuroVR, Nottingham, UK, 2011. Proceedings*, pages 67–74. Eurographics Association. 8

Trinh, T.-H., Querrec, R., De Loor, P., and Chevaillier, P. (2010a). Ensuring semantic spatial constraints in virtual environments using UML/OCL. In *Proceedings of the 17th ACM Symposium on Virtual Reality Software and Technology*, VRST '10, pages 219–226, New York, NY, USA. ACM. 8

Trinh, T.-H., Querrec, R., De Loor, P., and Chevaillier, P. (2010b). Specifying and dynamically visualizing semantic spatial constraints in a virtual environment for training using VRX-OCL. In *Actes des 5èmes Journées de l'Association Française de Réalité Virtuelle, Augmentée, Mixte et d'Interaction 3D (AFRV)*, pages 127–134, Paris, France. 8

Tutenel, T., Bidarra, R., Smelik, R. M., and Kraker, K. J. D. (2008). The role of semantics in games and simulations. *Comput. Entertain.*, 6:57:1–57:35. 8

Tutenel, T., Smelik, R., Bidarra, R., and Jan De Kraker, K. (2009). Using semantics to improve the design of game worlds. In *Proceedings of the Fifth Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE'09)*, pages 14–16. 17, 25

Vanacken, L., De Boeck, J., Raymaekers, C., and Coninx, K. (2008). Using relations between concepts during interaction modelling for virtual environments. In Latoschik, M. E., Reiners, D., Blach, R., Fidueroa, P., and Dachselt, R., editors, *Software Engineering and Architectures for Realtime Interactive System (SEARIS)*, page 65–70, March. 27

Vanacken, L., Raymaekers, C., and Coninx, K. (2007). Introducing semantic information during conceptual modelling of interaction for virtual environments. In *Proceedings of the 2007 workshop on Multimodal interfaces in semantic interaction*, WMISI '07, pages 17–24, New York, NY, USA. ACM. 16, 17, 28

Waller, D., Hunt, E., and Knapp, D. (1998). The transfer of spatial knowledge in virtual environment training. *Presence: Teleoper. Virtual Environ.*, 7(2):129–143. 3

Waller, D., Loomis, J., Golledge, R., and Beall, A. (2000). Place learning in humans: The role of distance and direction information. *Spatial Cognition and Computation*, 2(4):333–354. 3, 43

Wang, Y., Jayaram, U., Jayaram, S., and Imtiyaz, S. (2003). Methods and algorithms for constraint-based virtual assembly. *Virtual Reality*, 6(4):229–243. 56

Warmer, J. and Kleppe, A. (2003). *The Object Constraint Language: Getting Your Models Ready for MDA (2nd Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA. 55, 108

Werder, S. (2009). Formalization of Spatial Constraints. In *Proceedings of the 12th AGILE International Conference on Geographic Information Science, Jun*, pages 2–5. 55, 110

Wolter, D. and Lee, J. (2010). Qualitative reasoning with directional relations. *Artificial Intelligence*, 174(18):1498–1507. 165

Ziemann, P. and Gogolla, M. (2003). OCL extended with temporal logic. In *Perspective of System Informatics, LNCS 2244*, pages 351–357. Springer. 165

Zlatanova, S. (2000). On 3D topological relationships. In *DEXA '00: Proceedings of the 11th International Workshop on Database and Expert Systems Applications*, page 913, Washington, DC, USA. IEEE Computer Society. 48

Zlatanova, S., Rahman, A., and Shi, W. (2004). Topological models and frameworks for 3D spatial objects. *Computers and Geosciences*, 30(4):419–428. 48

Zlatev, J. (2007). Spatial semantics. In Cuyckens, H. and Geeraerts, D., editors, *Oxford Handbook of Cognitive Linguistics*, pages 318–350. Oxford University Press, Oxford, U.K. 1, 3