Conceptual Modeling of Articulated Bodies in Virtual Environments

Wesley Bille*, Olga De Troyer, Bram Pellens, Frederic Kleinermann
Wesley.Bille@vub.ac.be, Olga.DeTroyer@vub.ac.be,
Bram.Pellens@vub.ac.be, Frederic.Kleinermann@vub.ac.be
WISE, Vrije Universiteit Brussel
Pleinlaan 2, B-1050 Brussel

Abstract. Today, the design of a Virtual Reality (VR) application is still a specialized and a time consuming task. Most techniques and software tools require considerable knowledge of VR technology, programming skills and/or knowledge about script languages. This limits the participation of non VR-experts in the design and development of a VR application considerably. For this reason, we have developed an approach called "VR-WISE" that allows specifying a VR application at a conceptual level (free from implementation details) and from the viewpoint of a domain expert, allowing nonVR-experts to participate in the design of a Virtual Environment. This may shorten the development process and facilitate the communication. Our approach uses ontologies, incorporating domain knowledge, and high-level modeling concepts for describing the Virtual Environment. In this paper, we explain how articulated bodies can be modeled at a conceptual level by connecting objects using conceptual modeling primitives. We will present the different modeling concepts for connections and the associated constraints and illustrate this with examples.

1. Introduction

Although the creation of Virtual Reality (VR) applications is supported by a number of software tools, its development is still a specialized and tedious task. The tools available today for developing VR applications can be classified in two categories. The first category consists of the so-called toolkits (like Vortex [1]). Toolkits are programming libraries that provide a set of functions with which a skilled programmer can create VR applications. The second category is the one of the authoring tools (like 3D Studio Max [2]), which are complete programs with graphical interfaces for building Virtual Environments without having to resort to detailed programming. Although these tools assist the developer in creating a VR application, they still require considerable background knowledge about VR technology.

When developing a VR application, a first difficulty encountered is the necessity to translate the domain objects needed in the Virtual Environment (e.g. a house), into a combination of VR primitives (such as cylinders, spheres, textures, …) and free deformations. Currently none of the available VR development tools allow the developer to specify the Virtual Environment in terms of domain concepts. We illustrate this with X3D [3]. Although X3D allows the developer to create 3D content without having to deal with the low-level details of the platform or rendering process, the developer still has to specify the concepts using low-level primitives.

Another observation is that the design phase in the development process of a VR application (from the perspective of a classical software engineering life cycle) is usually a very informal activity. Few formal techniques in the context of VR exist to support this phase effectively. A

* Research Assistant for the Fund of Scientific Research – Flanders (FWO)

systematic approach that uses the output of the design phase as input for the implementation phase does not exist. Adding an explicit design phase to the development process of a VR application could also meet the first snag mentioned. If the design phase is done at a conceptual level, the design of a Virtual Environment can be expressed in terms of domain concepts. This could make the design accessible for non VR-experts such as domain experts. In addition, tools can be developed that may assist in the translation of the domain objects into VR primitives. Based on this idea we developed the VR-WISE approach [4][5] that supports an explicit conceptual design phase for VR application development. It provides a set of high-level modeling concepts to allow modeling a VR application using knowledge from the application domain. When we developed the set of high-level modeling concepts to support this design process, we carefully watched over their intuitiveness for non VR-experts. However, we wanted to investigate as well how to derive an implementation from such a conceptual design, it was also important to take their expressive power into account. The expressiveness of the modeling concepts needs to be high enough to be able to serve as input for the implementation process (automatic code generation). Next, the modeling concepts need to be unambiguous. Unambiguousness is needed from the perspective of the designer but also from the perspective of code generation. A graphical notation for the modeling concepts is provided. This will enhance the communication between the designers, programmers and other stakeholders. In addition, it is more efficient in use.

Until now our work focused mainly on modeling concepts for the design of simple objects and behaviors [13]. However, most VR applications also use complex or articulated objects. In this paper we focus on conceptual modeling concepts for articulated bodies. An articulated body is created by connecting different rigid bodies using joints. These connections physically attach two objects to each other and also limit the degrees of freedom for the movement of the objects with respect to each other. We provide three different modeling concepts for connecting objects: the *Connection Point Relation*, the *Connection Axis Relation* and the *Connection Surface Relation*. These relations respectively represent a connection over ‘a center of motion’, over ‘an axis of motion’, and over ‘a plane of motion’.

The paper is structured as follows. In the next section we introduce the VR-WISE approach. Section 3 describes the modeling concepts for connecting two bodies within the VR-WISE approach. Also their graphical representation is introduced and all modeling concepts are illustrated by means of examples. In section 4, we shortly discuss the tool support for our approach and section 5 discusses related work concerning methodologies for developing VR and high-level compound modeling. The paper ends with a conclusion and future work.

2. The VR-WISE approach

This section provides a general overview of our research in which the work described in this paper is situated. The approach we developed is called VR-WISE and can be used to design and generate VR applications. More details about the approach can be found in [5]. The goal of VR-WISE is to facilitate and shorten the development process of Virtual Environments by means of conceptual specifications. A conceptual specification is a high-level representation of the objects in the Virtual Environment, how they are related to each other, how they will behave and interact with each other and with the user. Such a conceptual specification (also called a *conceptual model*) must be free from any implementation details and not influenced by the current technical limitations of the VR technology. This new approach in the development of VR may also improve the reusability, extensibility and modularity of the VR application.

In VR-WISE, ontologies [6][7] are used as the underlying representation formalism. Ontologies are used for two different purposes: (1) explicitly during the design process for representing knowledge about the domain under consideration; (2) internally as general information representation formalism. In this paper, we will only briefly describe the ontologies used for purpose (1).

The design process in the VR-WISE approach consists of three sequential steps, namely the *specification step*, the *mapping step* and the *generation step* (see figure 1). We will briefly discuss these three steps.

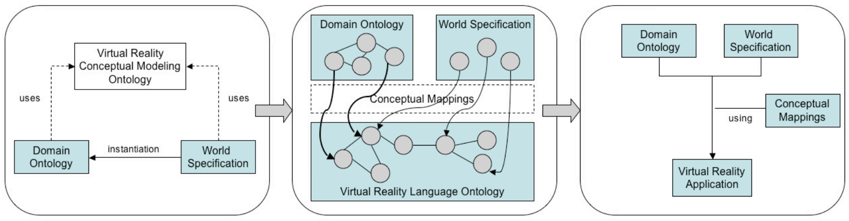


Figure 1: Overview of the VR-WISE approach

2.1. The Specification Step

The specification step allows specifying the Virtual Environment (VE) at a high level, using domain knowledge, and without taking any implementation details into account. The specification is done at two levels: a type level and an instance level. The type level is specified using a *Domain Ontology* and describes the concepts needed from the domain under consideration. Concepts can be compared to class definitions from the Object-Oriented (OO) modeling paradigm. The Domain Ontology describes the domain concepts by means of their properties as well as their relationships. For example, in the architectural domain, this ontology would contain concepts like Wall, Door, Window, Doorpost, and relationships such as “a Doorpost is always in a Wall”, “a Door needs to be attached to a Doorpost”. It is possible that such a domain ontology already exists (originally created for other purposes). In that situation, this ontology can be reused.

The instance level is specified by means of the *World Specification*, which will contain the actual conceptual description of the VE to be built. In fact, it describes the population of the VE. This World Specification is created by instantiating concepts of the Domain Ontology. These instances represent the objects that will populate the VE. Again this can be compared to the instantiation of a class in the OO paradigm. For the architectural example, there could be a number of Wall-instances, Door-instances connected to multiple Doorpost-instances. In addition, instance-specific information, (e.g. values for properties like size, color, location and orientation) and information specific for the world itself (e.g. gravity, lights…) is given in the World Specification.

To define concepts, instances, properties and relationships, our approach provides a number of high-level modeling concepts. Most of these modeling concepts are available at the type level as well as at the instance level. In addition, these modeling concepts are independent of the application domain. We will briefly explain the basic modeling concepts, as they are needed to understand the remainder of this paper.

a. Concepts – Instances
As our approach follows the object-oriented paradigm we have the modeling concepts ‘Concept’ and ‘Instance’. Concepts and Instances can have multiple attributes (properties). The value of an attribute of a concept can be either a concrete value or a parameter. Concrete values will be used as a default for every instance of that concept. Parameters can be replaced

by a concrete value when creating an instance of the concept. A property 'length' could for example have a parameter *x* as its value. This parameter could then be set to some concrete value. This promotes reusability and flexibility. Graphically, a concept is represented by a rectangle with a label specifying the name of the concept (see figure 2(a)). Figure 2(b) shows the graphical representation of an instance: an ellipse with the instance name and the concept name of the instance as label.

ConceptName
(a)
ConceptName: InstanceName
(b)

Figure 2: Graphical representation for (a) concept and (b) instance

b. Spatial Relations

One possible way to position instances in the VE is to give actual coordinates. However, sometimes (and especially for non VR-experts) it is more intuitive to use Spatial Relations to position objects in the VE. Therefore, we have defined a set of Spatial Relations: *Left*, *Right*, *Front*, *Back*, *Above*, *Under*, *Middle* and *OnTop*. These Spatial Relations can also be combined to form new spatial directions. Figure 3(a) shows the graphical notation to specify a Spatial Relation between an instance A from a class X and an instance B from a class Y. Figure 3(b) shows an example: a Chair instance c5 is 0.5 units in Front of the Table instance t1.

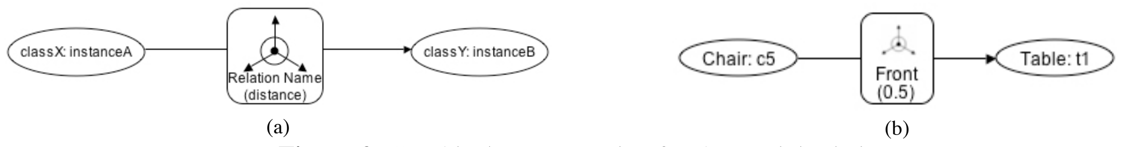


Figure 3: Graphical representation for the spatial relation

The use of Spatial Relations presumes that it is known what (for example) the front, left and above-direction of an object is. To define this, the designer has the possibility to name the axes of the reference frame of the object. For example the X-axis may indicate the left side of the object, the Y-axis the backside and the Z-axis the upper side.
The spatial relations will also be used to position the parts of an articulated body in the VE. In that case they may also be used at the instance or world specification level.

c. Orientation Relations

Spatial Relations provide an intuitive way to a non VR-expert to position objects relative to each other. However these relations do not involve the orientation of the objects. Suppose that a chair is positioned in front of a desk. Then, we still need to specify how these two objects are oriented with respect to each other. Therefore the 'Orientation Relation' (see figure 4(a) for its graphical representation) is introduced. For example, with this modeling concept we can specify that the chair is oriented with its right side towards the front side of the desk (specified in figure 4(b)). Note that we also provide an alternative Orientation Relation in which degrees can be used for orienting the object. Similar as for the Spatial Relations, Orientation Relations can be used at the type level as well as at the instance level.

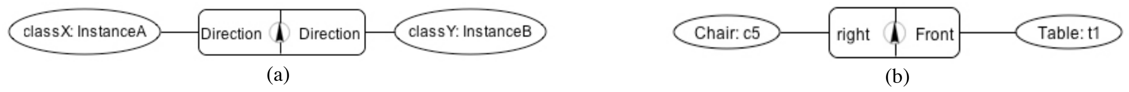


Figure 4: Example of the use of an orientation relation

2.2. *The Mapping Step*

The mapping step is the second step in VR-WISE. In this step the mapping from the conceptual level to the implementation level is specified. Similar to the specification, the mapping is defined at two levels. The *Domain Mapping* defines the mappings from the

concepts in the Domain Ontology to VR implementation primitives. The purpose of this mapping is to specify how a domain concept should be represented in the VE. This is a kind of default mapping that is also very useful for rapid prototyping. For example, in the architectural domain, a Door could be mapped onto a box. In this way each instantiation in the World Specification of the Door concept can be represented as a box.
Although instances may be of the same type (concept), they may in some cases require different representations. Therefore, the *World Mapping* allows defining the mappings from the instances to VR implementation primitives. In this way the default mappings, specified in the Domain Mapping, can be overwritten.

2.3. The Generation Step

This step generates the actual code for the VE specified in the specification step using the mappings defined in the mapping step, i.e. the conceptual specifications given by means of the Domain Ontology and the World Specification are converted into a working application by means of the mappings given by the Domain Mapping and the World Mapping.

3. Connection Relations

This section introduces the high-level modeling concepts defined in VR-WISE for connecting two bodies in order to form an articulated body. These connections will ensure that the connected bodies obey certain constraints in their position with respect to each other. The way the connection is established will be reflected in the behavior. Three different high-level Connection Relations are defined and discussed in this section. These are the 'Connection Point Relation', the 'Connection Axis Relation', and the 'Connection Surface Relation'.

3.1. Connection Point Relation

This type of Connection Relation connects two objects over a center of motion. This means that there is a point on both objects that, at any time, has to be positioned on the same coordinate in the global reference frame of the VE. Note that this point can also be a point outside the object.

The graphical notation for the Connection Point Relation is presented in figure 5. It is a rounded rectangle divided into two parts, each of which should be connected to a certain concept (or instance on the level of the World Specification). The icon in the middle makes it easier to recognize the relation. In each part of the rectangle, the connection point on the relevant object should be specified. By default (if nothing is specified) the middle point of the relevant surface of the relevant concept or instance will be used as connection point.
Although coordinates are not very suitable for non VR-experts, our approach also provides an alternative way to specify a connection point, namely by means of coordinates. We introduced this because it better fits the requirements of more advanced VE developers. The graphical representation for this is similar to the one in figure 5. The difference is that the coordinates of the connection point are given in the relevant part of the rectangle (see figure 6).

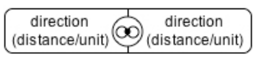


Figure 5: Graphical representation for the Connection Point Relation

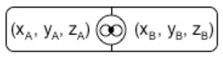


Figure 6: Alternative connection point relation

We will illustrate the Connection Point Relation with a simple example. We will model a compound concept Door that exists of a DoorHandle connected to a DoorBoard and a

DoorPost to which the DoorBoard is connected. The complete compound object could then be instantiated straight away in a virtual scene. Connecting the DoorHandle to the DoorBoard is done by means of a connection point. Here, the specification is done at the concept-level. This means that later on different instances can be created of this Door concept. The first step in the specification is to give the DoorBoard and the DoorHandle the correct position and orientation. E.g. the DoorHandle is in front of the DoorBoard with a distance of 0 meter, so it is against the DoorBoard (specified by the Spatial Relation 'Front' between the concepts DoorBoard and DoorHandle in figure 7). From this information the side of each object on which the connection point will be positioned can be derived. In the example we know that the DoorHandle is in front of (against) the DoorBoard, therefore the connection point on the DoorBoard has to be in the front side of it and for the DoorHandle it has to be on the back side. Next, we define the connection point on the DoorBoard: it is defined as 0.4 meter right from the middle point of the front surface. For the DoorHandle the default (being the middle point) is used. Figure 7 shows the complete specification of the Door concept. It also shows the graphical representation of a compound object which is a rectangle containing the model of the compound object.

Note that to position the connection point relative to the middle point, the Spatial Relations introduced in section 2.1 can be used. These Spatial Relations can be combined in three dimensions so it is possible to define a connection point inside, but also outside an object as objects may move with respect to each other over a center of motion lying outside the objects.

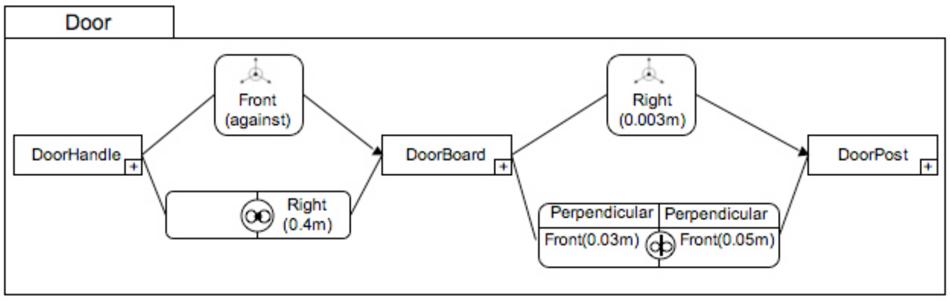


Figure 7: Conceptual model for the compound class "Door"

3.2. Connection Axis Relation

Connecting objects using a connection point leaves three degrees of freedom in their movement according to each other. They can rotate around the three axis of the reference frame. Using a connection axis, the movement of the objects according to each other is limited to the movement around or along this axis. Therefore, our approach provides the Connection Axis Relation. In this case, instead of defining a point on each object that has to be at the same position at all times, an axis on each object need to be defined that had to coincide at all times.

To define the axis we use a plane through the object. For this, we provide three different planes intersecting the object: the horizontal plane (defined by the front-back and left-right directions), the vertical plane (defined by the front-back and top-bottom directions) and the perpendicular plane (defined by the left-right and the top-bottom directions). The connection axis is defined as the intersection of the plane and the casing of the object on the side of the connection. Our approach provides the ability to translate or rotate the plane so that more different possible connection axes can be defined. This principle is demonstrated in figure 8(a). In this figure, the horizontal plane is taken and it is shifted one meter to the bottom. When the connection is on the front side of the object, it results in the connection axis shown in figure 8(b).

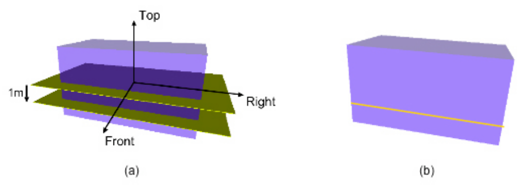


Figure 8: Defining a connection axis using the horizontal plane and a translation

To continue with the door example we will now explain how the DoorBoard is connected to the DoorPost. First we position the Door-Board and the DoorPost with respect to each other. E.g. the Door-Board is right of the DoorPost with a distance of 3 millimeters (specified by the Spatial Relation 'Right' between the concepts DoorBoard and DoorPost in figure 7). Next we specify the connection axis on both the DoorBoard and the DoorPost. For the DoorBoard we take the perpendicular plane (which lies parallel to the front plane of the DoorBoard and goes through the middle point of the DoorBoard) and shift it with half the depth of the DoorBoard, 3 centimeter, to the front. The intersection between left side of the DoorBoard (this is the side where the connection will take place) and the shifted plane defines the connection axis on the DoorBoard. The same is done for the DoorPost. Figure 7 shows the complete specification for this.
In addition to defining the connection axes, we also need to be able to position the objects on the defined connection axes. By default, the middle point of the axes defined on the objects that need to coincide will fall together. However, it is also possible to shift the connection points. By shifting these points, the objects can be positioned along the axis. In figure 9 we see two desks. If we want that the legs can be folded then we need to connect the tabletop and the legs by means of a connection axis. The difference between the desk in figure 9(a) and the one in figure 9(b) is that the legs are positioned more to the back of the desk in figure 9 (b). How this is realized is shown in figure 10. Figure 10(a) shows the default situation that corresponds to figure 9(a) while 10(b) corresponds to the situation of figure 9(b).

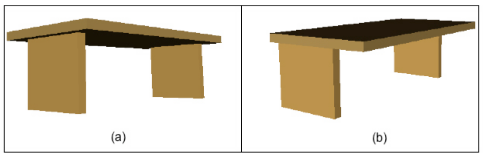
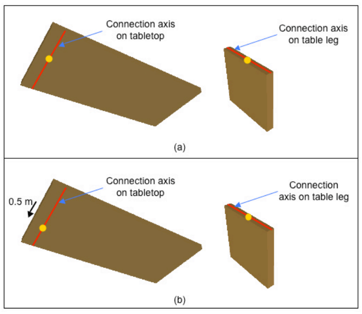


Figure 9: Different positions on the same connection axis


Connection axis on tabletop
Connection axis on table leg
(a)
0.5 m
Connection axis on tabletop
Connection axis on table leg
(b)
Figure 10: Shifting the connection point of a connection axis

The graphical representation for the Connection Axis Relation is also a rounded rectangle divided into two parts. Again there is a specific icon for this relation. Each part is connected to an object from the articulated body. Inside each part, the plane that should be used to define the connection axis for the relevant object is specified, as well as the rotations and translations on the plane. The graphical notation is illustrated in figure 11. For the connection axis on the object connected to the left part of the relation we have taken the vertical plane and shifted it left over a distance of 1.5 meter. This translation is expressed by means of the function *trans(direction, distance)*. We have also shifted the connection point of the connection axis to the front over a distance of 0.5 meter. This is indicated by the function *cp(direction, distance)*. For the connection axis on the object connected to the right part of the relation we have taken the perpendicular plane and rotated it over the top to bottom axis over 30 degrees. The rotation axis can be denoted by 'tb' (top to bottom), 'lr' (left to right) and 'fb' (front to back).

The rotation itself is expressed by means of the function *rotate(axis, degrees)*. In section 3.4 we will provide an example of the usage of this relation.

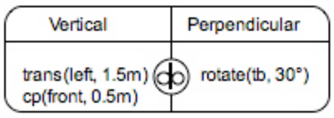


Figure 11: Graphical notation for the Connection Axis Relation

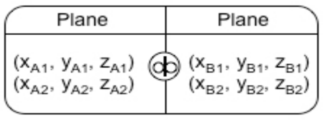


Figure 12: Alternative Connection Axis Relation

Also here, we provide a method for defining a connection axis by means of two coordinates (see figure 12). In that case, the line through these coordinates forms the connection axis.

3.3. Connection Surface Relation

The third way two connect two objects is over a surface of motion. Using a connection surface, all degrees of freedom for the movement of the connected objects according to each other are gone. For this we provide the Connection Surface Relation. The mechanism is the same as for the Connection Point Relation. In fact, a connection point needs to be defined. The intersection of the surfaces touching each other when the connection point falls together will be the connection surface. For example when connecting a table leg to a table, the touching surface of the table leg to the table will be the connection surface. The graphical notation for this relation is similar to the one for the Connection Point Relation but with a different icon. This is shown in figure 13.

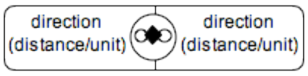


Figure 13: Connection Surface Relation

3.4. Roles

The same concept can be used in different places in an articulated body, e.g. a Table has a left and a right TableLeg. In the definition of the Table concept we need a connection with the TableTop for each TableLeg. To differentiate between these two TableLegs we use the concept of 'Role'. A Concept can play different Roles in the definition of an articulated body.

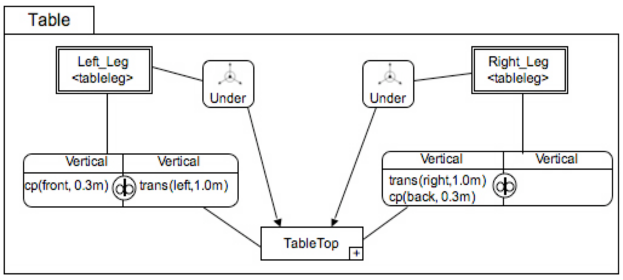


Figure 14: the use of roles in the conceptual model

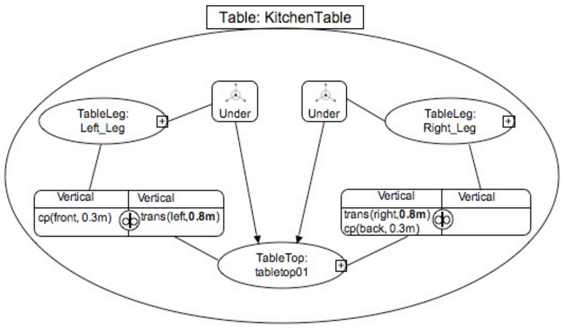


Figure 15: instantiation of a compound object

Graphically, a Role is represented by a double-sided rectangle. Inside the rectangle the name of the Role together 170with the name of the concept playing the Role is written. Figure 14 shows two Roles for the concept TableLeg, a Role Left_Leg and a Role Right_Leg. Note that both are connected to the TableTop by means of a Connection Axis Relation.

3.5. Instantiating an Articulated Body

So far, we described how to model an articulated body at the Concept level (being the Domain Ontology). In the World Specification Model we may instantiate these bodies. This will happen by instantiating each Concept (and each Role) used in the definition of the articulated body. In the Domain Ontology, default values are specified for the properties of these Concepts. At instantiation (at the World Specification Level), the default properties inherited from the Concept definition can be overwritten for each instance. Figure 15 shows the example of a kitchen table being an instance of the concept Table from figure 14. Note that some (inherited) property values are overwritten. Instead of positioning the legs 1 meter to the left and the right side of the table, they are positioned at 0.8 meter (the property values inside the Connection Axis Relations are changed from 1 to 0.8 meter).

4. Tool support

To support the VR-WISE approach, we developed a prototype tool called OntoWorld. The tool enables a designer to make a conceptual design, specify the desired mappings and finally generate a VR application from the specifications. In the current version VRML/X3D code is generated.
We have extended the prototype tool with the Conceptual Specification Generator (CSG). This is a graphical diagram editor supporting the modeling of the articulated bodies as described. The tool has been implemented as an extension to Microsoft Visio [9]. Consistency between the diagrams and the ontologies is maintained. The CSG can be considered as a graphical interface for the specification phase (first step in the VR-WISE approach).

5. Related work

First, we will discuss some approaches that were developed to connect rigid bodies. The Cody Virtual Constructor (COAR) [10] is a knowledge-based system that enables the interactive assembly of 3D visualized mechanical parts to complex and novel aggregates in a Virtual Environment. The user may either directly manipulate the virtual scene by grasping, moving and assembling parts using the mouse or he can instruct the system using simple commands in natural language. This approach makes use of specific connection points that are predefined. For very complex objects it may become difficult to remember these connection points. In our approach, however, the connections are defined in a more intuitive way.
The Virtual Assembly Design Environment (VADE) [11] is a VR based engineering application that allows engineers to evaluate, analyze, and plan the assembly of mechanical systems. This system focuses on utilizing an immersive virtual environment tightly coupled with commercial Computer Aided Design (CAD) systems. However, the COAR and the VADE approaches are very much oriented towards the mechanics and engineering domain and thus not conceptual in a way that the designer can use his own domain terminology to describe a VE.
We have also looked at 3D engines like Vortex[1] and ODE [12]. Inside these programming libraries some joint constraints are predefined. If we compare the possibilities of our approach at this stage of the research and the possibilities provided by such toolkits, then we see that our approach is able to model most of the joint constraints on a conceptual level. Examples of these joint constraints are the ball and socket joint, hinge joint and prismatic joint. In Vortex the programmer can place some extra constraints on top of the actual connection. In the near future we will also provide this by allowing to define constraints on our Connection Relations. With some minor extensions our approach will offer at least the same functionality as most widely used toolkits. However, the advantage of our approach is that the design is at a conceptual level. This makes it easier to communicate about the design with clients and will save time because miscommunications can be avoided and fast prototyping is possible. Having an underlying semantic model for the virtual environment will allow querying the virtual environment at a conceptual level using domain terminology.

6. Conclusions and Future Work

Our ultimate goal is to make the development of VR applications available to a much broader public. We want to realize this by introducing an explicit conceptual design phase as an initial step in the development process of a VR application. The use of a conceptual modeling phase with high-level modeling concepts and in terms of domain concepts allows a better participation of domain experts into the development of a VR application. In this paper we have presented the modeling concepts developed for the specification of complex or articulated bodies. We have presented three different Connection Relations that can be used to connect rigid bodies. These are the Connection Point Relation, the Connection Axis Relation and the Connection Surface Relation.

A prototype tool called OntoWorld was already available to support the overall approach. We extended it with a graphical front-end (build in Microsoft Visio) to allow creating articulated bodies using intuitive diagrams.

Future work will focus on the ability to specify constraints on these connections. An example of such a constraint is the hinge constraint. This constraint can be placed on a Connection Axis Relation to specify how much the connected objects can turn around this axis in a particular direction and not move along the connection axis. We will also investigate on more advanced joint types. Next we will perform user experiments to evaluate the usability of the approach and the graphical notation.

7. Acknowledgements

This research is carried out in the context of the Ontobasis and the VR-DeMo project; both projects are funded by the Institute for the Promotion of Innovation by Science and Technology in Flanders (IWT). It is also partially funded by the FWO (Fund of Scientific Research – Flanders).

References

[1] CMLabs Simulations, Vortex Developer Guide: A Manual for the Vortex Simulation Toolkit, 2003.

[2] K.L. Murdock, 3ds max 5 Bible, ISBN: 0 7645 3703 2. Wiley Publishing Inc., New York, 2003.

[3] Web 3D Consortium (Web3D): Extensible 3D (X3D) International Standard http://www.web3d.org/x3d/specifications, 2003

[4] W. Bille, B. Pellens, F. Kleinermann and O. De Troyer, Intelligent Modelling of Virtual Worlds Using Domain Ontologies, *Proceedings of the Workshop of Intelligent Computing (WIC)*, Mexico City, Mexico 2004, pp. 272-279.

[5] W. Bille, O. De Troyer, F. Kleinermann, B. Pellens, and R. Romero, Using Ontologies to build Virtual Worlds for the Web, *Proceedings of the IADIS International WWW/Internet 2004 Conference*, Madrid, Spain, pp. 683-690

[6] T.R. Gruber, A translation approach to portable ontologies, *Journal of Knowledge Acquisition*, 5(2). 1993, pp. 199-220.

[7] N. Guarino and P. Giaretta, Ontologies and knowledge bases: towards a terminological clarification, *Towards Very Large Knowledge Bases: Knowledge Building Knowledge Sharing*, IOS Press, Amsterdam, 1995, pp. 25-32.

[8] A. Valente and J. Breuker, Towards principled Core Ontologies, *Proceedings of the Tenth Knowledge Acquisition for Knowledge-Based Systems Workshop*, Alberta, Canada, 1996, pp.301-320.

[9] M. Walker, N.J. Eaton and N. Eaton, Microsoft Visio 2003 Inside Out, ISBN: 0 7356 1516 0. Microsoft Press, 2003

[10] B. Jung, M. Latoschik and I. Wachsmuth, Knowledge-Based Assembly Simulation for Virtual Prototype Modeling, *Proceedings of the 24th Annual Conference of the IEEE Industrial Electronics Society, Vol. 4,* IEEE, 1998, pp. 2152-2157.

[11] S. Jayaram, Y. Wang, U. Jayaram, K. Lyons and P. Hart, A Virtual Assembly Design Environment, *Proceedings of the Virtual Reality Conference*, Houston Texas, USA. 1999, pp. 172-179.

[12] Open Dynamics Engine, http://www.ode.org, last accessed on 26th of May 2005

[13] B. Pellens, O. De Troyer, W. Bille and F. Kleinermann, Conceptual Modeling of Object Behavior in a Virtual Environment, accepted for publication on Virtual Concepts 2005, Biarritz, France. Proceedings published by Springer-Verlag, 2005.