Gliwice **2003.09.01**

SILESIAN UNIVERSITY OF TECHNOLOGY

FACULTY OF AUTOMATIC CONTROL, ELECTRONICS AND COMPUTER SCIENCE

MSc thesis

Automatic generation of virtual worlds from architectural and mechanical CAD models

Supervisor: **Prof. Józef Ober**
Consultant: **PhD Lars Hagge**

Author:
Daniel Szepielak

Table of content

1 Introduction

DESY in Hamburg is one of the world's leading centers for particle accelerator research. At DESY, physicists can study the world of elementary particles at the large underground accelerator ring HERA (Hadron-Electron Ring Accelerator). Second center of activity at DESY is the research with synchrotron radiation, a special form of light that is produced in particle accelerators. Synchrotron radiation is used for experiments on the physics of atoms and solid bodies, as well as in the fields of chemistry, geology, materials science, molecular biology and medicine [12].

In collaboration with institutes from all over the world, DESY is planning a research facility called TESLA (TeV-Energy Superconducting Linear Accelerator). TESLA comprises two facilities: a 33-kilometer-long linear accelerator, TESLA-LC, which will bring electrons into collision with positrons, and a 4-kilometer-long X-ray free electron laser TESLA-XFEL. The German government has recently approved the formation of a European collaboration to build the XFEL at DESY.

During the preparation of the TESLA project the need for fast and efficient 3D-visualization of accelerator components emerged. Because of its scale, the TESLA project involves extensive architectural and mechanical design work. People from various collaborative institutes are working on different parts of the project. They create CAD models which are stored in several different locations and which are thus difficult to access. Efficient work requires joining the models from different designers in order to check if all pieces fit together. The whole TESLA accelerator will consist of a great number of components placed over a broad area. A complete CAD model of the whole accelerator would be extremely huge and almost impossible to display. The possibility to select a desired area of an accelerator, create a 3D model of that selected area on the fly and then display it, would ensure that 3D models are small enough to display and explore them. If this functionality were accessible through a web-based interface, then each designer could have access to all 3D models regardless of where they were created.

To provide the described functionality, a new system shall be developed. This system is in the following referred to as “Automated Virtual Reality Creator” (AVC). An application called View Loader, whose development will be described as a part of this thesis, will be the core of this system.

1.1 Purpose and benefits

The main goal of the project is to provide a tool that could visualize a selected area of an accelerator and its surroundings as a virtual world. This Automated Virtual Reality Creator should be accessible to users at DESY and it should be intuitive to use.

Figure 1-1 shows the intended operation principle of AVC. AVC combines geographical, engineering and architectural data from different sources into a single integrated data set. Sources of these data are systems like a Geographic Information System (GIS) and CAD systems for architectural and mechanical design (ACAD, MCAD). Using these integrated data, AVC constructs a virtual world of the desired area. Users will be able to explore accelerator components, buildings and all other objects for which CAD models are available.

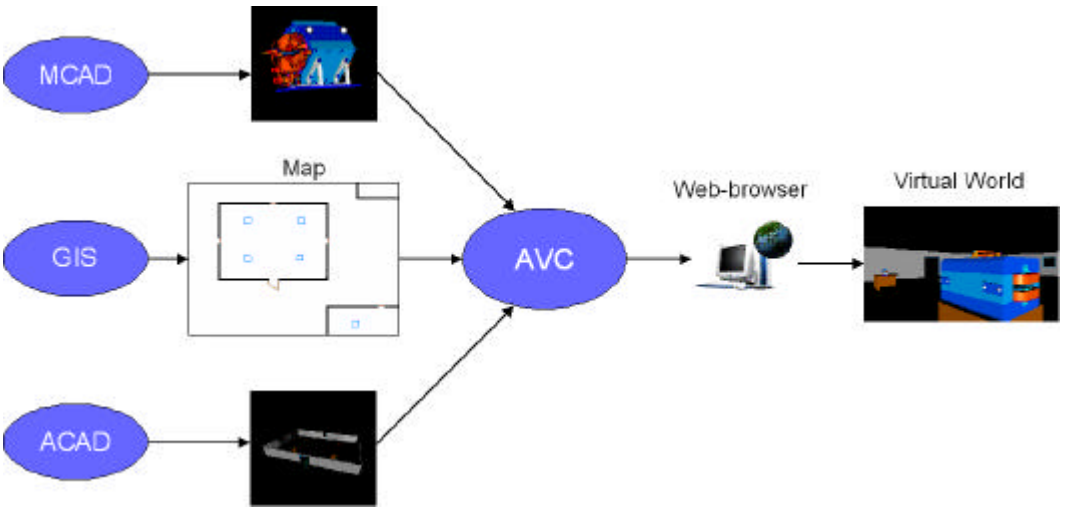


Figure 1-1 High-level AVC structure.

Many departments, including e.g. engineering, safety and transportation, could benefit from the use of AVC: the easy access to information offered by AVC makes e.g. planning of work easier. For example, workers who need to install, transport or work on components may check the available installation space only on location in the accelerator tunnels. With AVC, the installation spaces could be remotely explored. This is additionally desirable as the accelerators are not accessible during operation.

1.2 Overview

This thesis will first discuss AVC requirement analysis. Then it will describe theoretical issues of data visualization. This will include a short explanation of purposes and advantages of data visualization as well as a description of methods and problems related to combining data from different CAD sources. The thesis will then explain the architecture of ViewLoader, the core component of AVC and the used algorithms and technologies. At the end it will describe two case studies which will show the capabilities of the system.

2 Envisioning

2.1 Perspective

3D visualization is a powerful way of presenting information that otherwise would be difficult to analyze and understand.

Engineers who design accelerator components and need to check if there is enough installation space for a new designed component have to personally verify the installation place and make sure that the component will properly fit with rest of the components. Similarly, the available space for transporting components through the tunnel or for performing maintenance and installation work has to be verified before the work begins. With AVC, engineers would be able to see the installation place in virtual reality, put their components at the desired positions, and check for possible collisions or other shortcomings.

3D visualization is already in use at DESY but it has some limitations. The current visualization methods do not allow for selection of the area to be visualized. All the existing 3D models are static and because of that often inconvenient to use. Figure 2-1 shows the official 3D model of the TTF-2 accelerator, including the injector hall (top), the main accelerator tunnel and the "Expo" hall which was previously housing an exhibition and which will now be used for experiments. The TTF-2 accelerator is a test facility for TESLA technology.

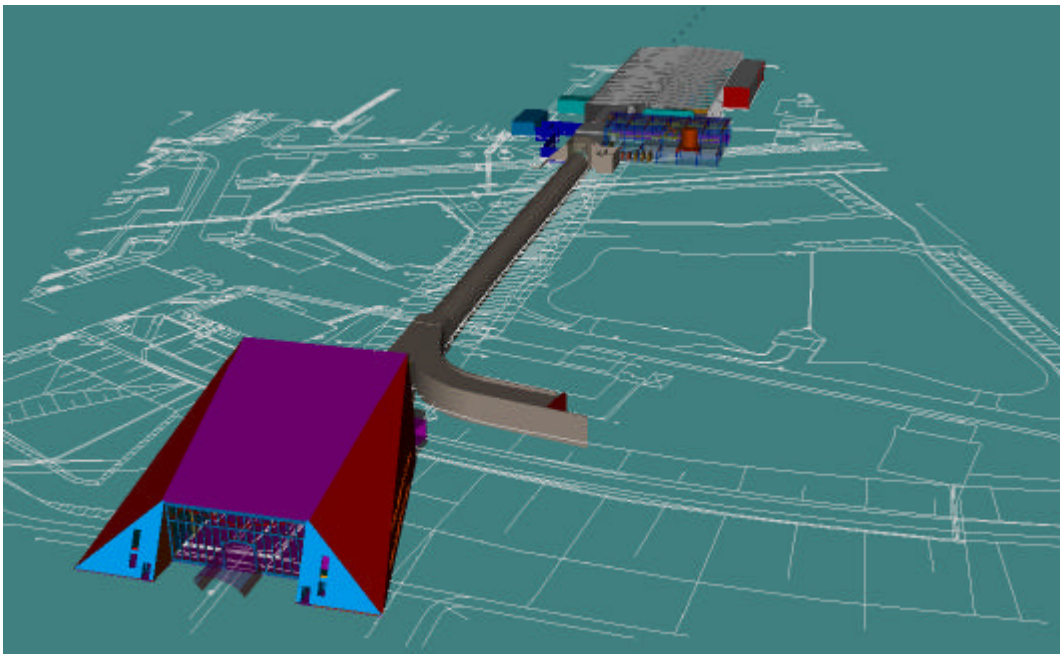


Figure 2-1 The TTF-2 accelerator installation in Virtual Reality as displayed by a dedicated 3D Mock-Up tool.

Figure 2-2 shows how an installation can be explored using virtual reality techniques: zooming to an area of the main tunnel (1) and deselecting the tunnel building from the scene (2) reveals the accelerator installations (3).

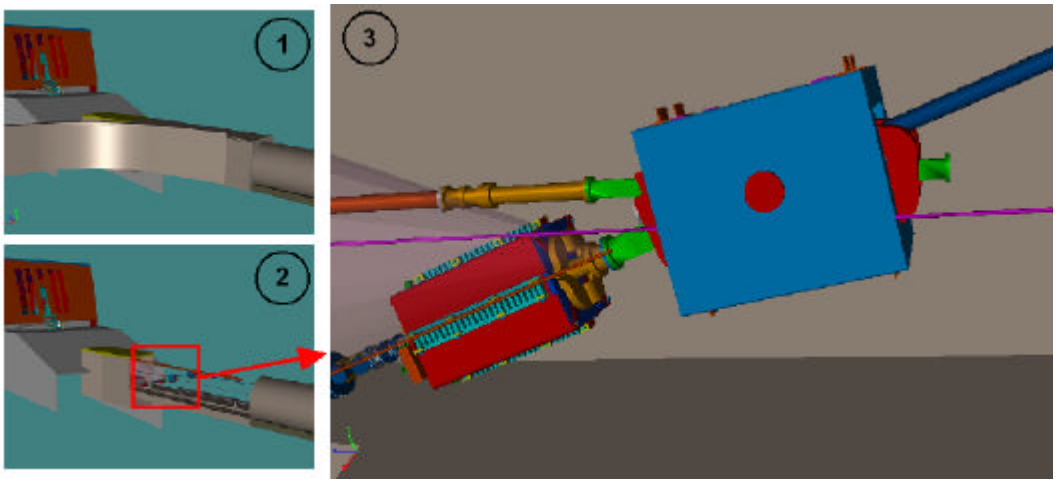


Figure 2-2 Exploring TTF-2 using VR techniques.

The displayed model enables engineers to visualize and explore the accelerator but it contains the entire TTF-2 accelerator and is therefore inconvenient to handle: movements in the viewer are slow and navigation is complicated as it is done on the part breakdown structure of the accelerator. With AVC it is possible to create a virtual world only for a selected area of an accelerator. The model thus remains small and easier to handle. Navigation is eased as it is based on a map of the accelerator.

Currently access to data from GIS and CAD systems is limited to particular user groups which may result in situations where users who do not have access to the specific data have problems in getting the information they need. The functionality of AVC will ensure the availability of data to any user in the DESY intranet. All this will enable a faster and more efficient flow of information between separate user groups.

2.2 General requirements and constraints

The most important requirement for AVC is the possibility to visualize only a selected part of a map. This feature will ensure that there will be no unnecessary operations during the creation of the virtual world and the performance of the 3D-viewers will not be degraded by processing unwanted data. The system should also allow users to move inside the virtual world, switch the graphical representation of components on and off, and interact with the

virtual world in general. Another useful feature would be the possibility to access database information about visualized components by simply clicking on them.

The final solution has to use data formats and tools which are available at DESY or which are free or relatively easy to obtain. The amount of 3D data has to be kept small for visualization tools. Some strictly defined design rules for CAD models have to be applied, covering e.g. the choice of coordinate systems or the structuring of designs. The system has to be available on the Web and intuitive to use for casual users.

2.3 User groups

Generally the AVC users can be divided into two groups, casual users and engineers.

The first group includes workers from e.g. survey, administration and safety departments who need to access the installations, but usually do not require detailed technical information from mechanical engineering. Casual users usually know how to handle Office tools, Web-Browsers, and e-mail.

Engineers are additionally trained in handling CAD systems such as I-DEAS or AutoCAD, and other specific tools, including special purpose viewers.

Both user groups do not necessarily have an in-depth knowledge of information technology.

2.4 Environment

There are several operating systems in use at DESY including a variety of Windows, UNIX and Linux platforms.

The following Web-Browsers are supported for the operating systems:

- Internet Explorer 5.5 SP1 / 6.0
- Mozilla 1.3

It should be possible to use VRML clients such as CosmoPlayer 2.1 or similar with the above mentioned operating systems and Web-Browsers.

2.5 Dependencies, risks, assumptions

The operation of AVC depends on many factors. All of the system requirements have to be met, otherwise AVC will not be complete. The biggest problem is caused by the existence of different coordinate systems which are used in separate components of AVC. All these coordinate systems have to be combined in order to create a complete virtual world. Clear transformation rules

from one coordinate system to another have to be defined and applied to all parts of the system.

A complete 3D world can only be obtained if CAD models are available for all components in the scene. Currently some components like e.g. support structures are not available. The models have to be provided from engineering departments.

Because AVC is built from a number of components there is a risk that future versions of one or more of them will not be compatible. The highest risk is involved with 3D format translators.

It is uncertain when particular components of AVC will be available, implying that some of the tasks will have to be done manually until then.

3 Requirements analysis

3.1 System use cases

The major application of AVC is the automated construction of virtual worlds from several separate input models.

The AVC system has a single main use case called *View 3D Model*. Its task is to create and display a virtual world. The created virtual world should contain only objects which are present inside an area selected on a map by the user. The possibility to select areas on a map is a very important aspect of the AVC. The completion if this task requires information about the spatial position of each object and 3D object representations of all the objects that are present in the selected area.

In the following chapters the UML notation [11] will be used to describe the requirements and system models.

The main use case includes three sub use cases: *create picklist*, *create 3D model*, *display 3D model*. Figure 3-1 shows the corresponding use case diagram.

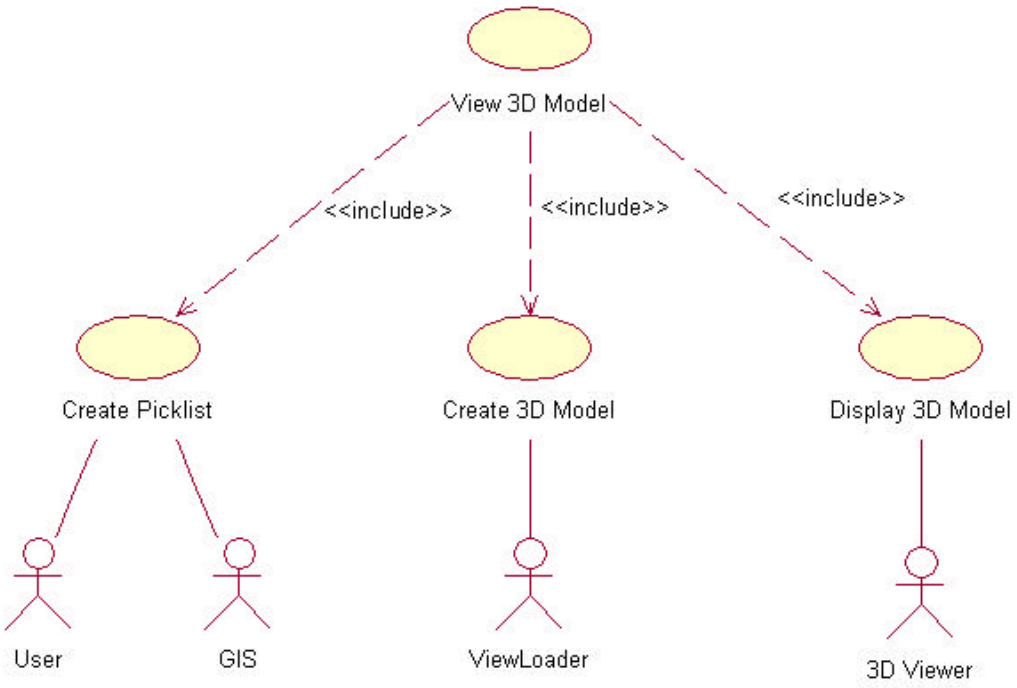


Figure 3-1 AVC use cases.

The sub use cases are defined as follows:

- *create picklist* starts when the user selects an area on a map in the GIS and delivers a list of all objects, including their positions, which are located in that area. This list is called *picklist*. The use case is initiated by the user and performed by the GIS.

- *create 3D model* interprets a *picklist* and combines 3D object representations of each object in their correct positions and alignments into a single 3D model for visualization purposes. It is initiated by the GIS whenever a new *picklist* is created and performed by an actor called ViewLoader, which is the core component of the AVC.

- *display 3D model* presents an interactive view of the virtual world which allows the user to explore it. It is performed by an actor called 3D viewer and is invoked each time when a new 3D model is created by ViewLoader.

3.2 Requirements

An important step in the process of building the system is the specification of its requirements. It enables determining what features the system should have, which of them are absolutely necessary, and which are only desirable. For each of the requirements there should be an assessment made of how difficult its implementation would be. Some of the requirements may change during the process of building the system which means the design should be flexible enough to allow replacing of existing elements.

To better understand how AVC works and which component is responsible for which of the requirements, the following activity diagram is presented. The diagram shown in Figure 3-2 refers to the *View 3D model* use case shown in Figure 3-1.

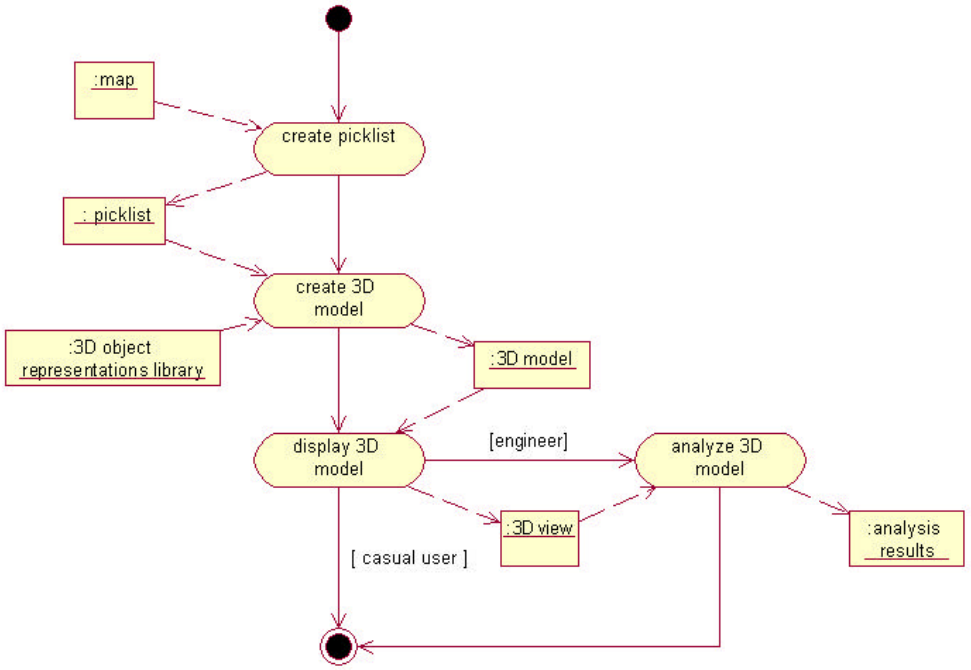


Figure 3-2 High-level AVC activity diagram.

The presented activity diagram explains the order of activities and shows all the objects used during the construction of the virtual world. The use case starts when the user selects an area on the map. The map object is presented to the user by the GIS which has data about the positions of all objects presented on the map. Using these data, the GIS produces the *picklist* – a list of spatial coordinates of objects included inside the selected area (*create picklist* activity). Information from the *picklist* is next used to place 3D object representations in their proper positions in the 3D model which contains the constructed virtual world (*create 3D model* activity). The source of 3D object representations is a library of objects which has to be provided in advance as a part of AVC. When the 3D model is completed it is displayed by the 3D viewer (*display 3D model* activity). Some 3D viewers allow to additionally perform analysis on the displayed model (*analyze 3D model* activity) like determining distances or checking for object collisions.

The requirements for AVC can be divided into two groups, user requirements and system requirements. The user requirements are gathered and presented in the following table:

No.	USER REQUIREMENT	PRIORITY	DIFFICULTY
UR1	The user should have the possibility to select in the GIS an area on a 2D map which should be visualized in 3D	HIGH	LOW
UR2	AVC should automatically create a 3D model of objects which are located in the selected area of the map	HIGH	MEDIUM
UR3	AVC should be able to use CAD models from various CAD applications (currently from I-DEAS and ADT)	MEDIUM	HIGH
UR4	The user should be able to interact with the created virtual world	MEDIUM	LOW
UR5	Engineers should have a possibility to analyze 3D models	MEDIUM	LOW
UR6	The user should have the possibility to display information about visualized components which is stored in other IT systems	MEDIUM	MEDIUM
UR7	AVC should allow using different positioning methods for different types of objects.	LOW	MEDIUM

The above mentioned requirements are later referred to as e.g. UR1 – user requirement 1.

Because the user has to be able to select an area of a map for 3D visualization (UR1), AVC should provide a graphical user interface. This interface should display a 2D map and allow the user to select the desired area using a mouse. The user interface should also provide buttons for actions such as confirming selection, zooming, shifting etc.

Since the construction of the virtual world should be done automatically (UR2) some conditions have to be fulfilled. Virtual worlds have to be constructed using information about the spatial position of objects from the selected area and the 3D-object representations of accelerator components, buildings etc. AVC should store only single instances of each type of object. These set of objects should be reusable. It means that a base object should be duplicated and placed in its spatial coordinates each time when it occurs in a 3D scene (this mechanism is described in section 4.3). This feature will ensure that there will be no redundancy of 3D data in AVC.

AVC will use as its input data the 3D object representations from multiple CAD sources (UR3). Because each CAD application creates its models differently it can be necessary to prepare a separate preprocessor for data from each CAD application.

Since the user should have the possibility to interact with a virtual world (UR4) AVC should support actions such as moving, rotating, shifting, switching on/off displayed components etc. Engineers additionally require performing additional tasks like e.g. measuring or collision analysis (UR5). This functionality is offered by the JT viewer VisView, because of which the JT format has to be supported.

Displaying information about visualized components (UR6) will require from AVC to connect with external databases that holds the wanted data.

The user wants to use different positioning methods for different object types (UR7). Currently two methods of positioning have to be supported by AVC – one for accelerator components and the second for any other object types. Both methods are described in section 4.2. The following table presents the system requirements described above:

No.	SYSTEM REQUIREMNT
SR1	AVC should provide a functional graphical user interface that allows e.g. for zooming and shifting maps …
SR2	Virtual worlds should be created using a set of base objects that can be used multiple times and placed at different spatial coordinates
SR3	Separate preprocessors have to be created for accepting the data from the different CAD applications
SR4	AVC should support actions such as moving, rotating, shifting or switching on/off displayed components
SR5	Because engineers require additional analysis functionality, the JT format has to be supported
SR6	AVC should be able to access external database in order to display information about the components
SR7	AVC should support two positioning methods, one for accelerator components, a second for any other type of objects

Analysis of the requirements shows that two separate user groups can be identified. The first group, casual users, needs only to observe and interact with the displayed virtual world while the second group, engineers, needs to additionally analyze the displayed 3D model. Figure 3-3 shows the *Analyze 3D model* extension of the *View 3D model* use case. To satisfy their needs both user groups will use a 3D viewer that best suits their task. Thus, the casual users will use a VRML viewer while the engineers will use the JT viewer which offers the extended analysis functionality.

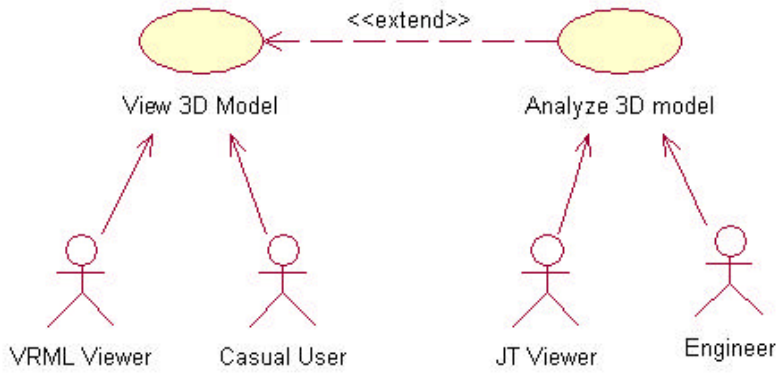


Figure 3-3 Two ways AVC can be used.

3.3 Constraints

The development of each computer system has its own constraint conditions. It can be a matter of limited funds, a specific hardware on which the system has to run or many other conditions. The development of the system which is described in this thesis is also restricted to specific conditions.

The currently used viewer for 3D models is VisView. The currently used design applications for 3D models are I-DEAS and ADT.

Each of these tools use specific data formats and those formats have to be supported by AVC. There are two different file formats which have to be supported: JT and VRML. The first one is being used by VisView, the second one can be viewed by viewers such as CosmoPlayer [21] or Cortona [26] which are free tools and can be downloaded from the Internet.

High detail CAD models cannot be used because the performance of 3D-viewers, format converters and other system components is limited. Models should be simplified for satisfactory AVC performance.

All CAD models have to use a specific coordinate system which is either compatible with the global coordinate system used in the GIS, or a coordinate system for which transformation rules are known.

4 System design

4.1 Background

This section presents the theoretical background needed for the later sections of this thesis. It defines and explains such concepts as Visualization, Virtual Reality and Geographic Information System.

4.1.1 Visualization and Virtual Reality

Visualization technology can be used to graphically illustrate various concepts in computer science. There are numerous visualization applications for geological, engineering, health, and other scientific endeavors. This section gives a brief introduction to visualization and virtual reality (VR).

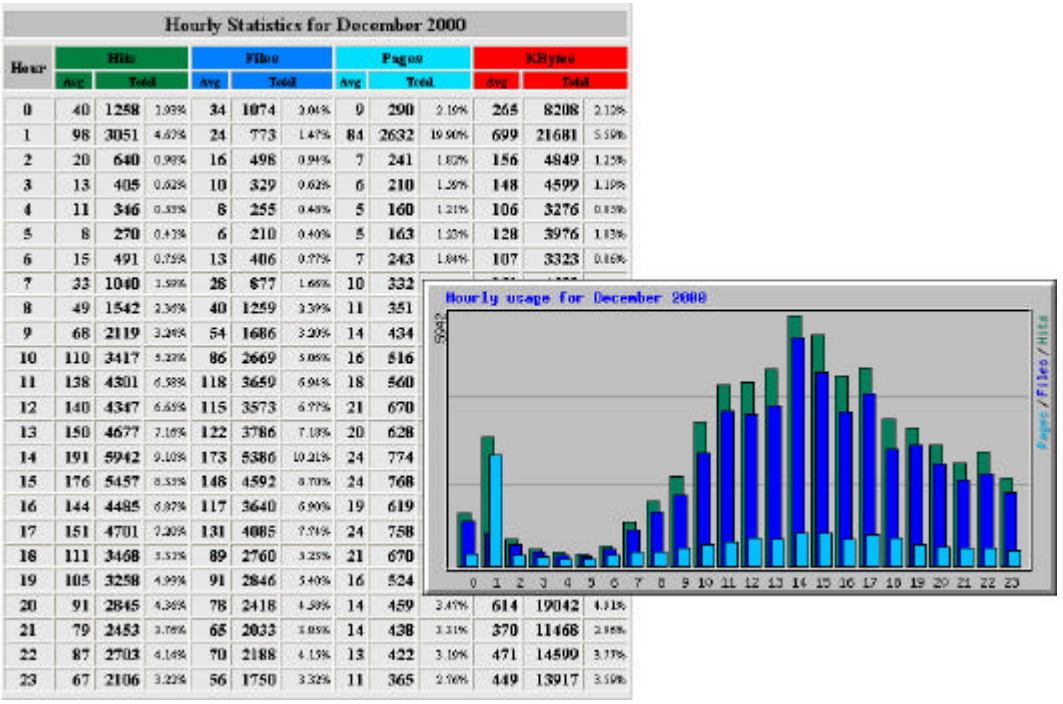
First the key concepts have to be defined:

- Visualization is the process of representing data as images that can aid in understanding the meaning of the data [13]. This might include anything from a simple X-Y graph of one dependent variable against one independent variable to complex visualizations such as virtual reality.

- Virtual reality is the simulation of a real or imagined environment that can be experienced visually in the three dimensions of width, height, and depth. The simplest form of virtual reality is a 3-D image that can be explored interactively at a personal computer, usually by manipulating keys or the mouse so that the content of the image moves in some direction or zooms in or out [14]. VR technology was originally developed by the military in the 1960s to help to train pilots.

The purpose of visualization is mapping any type of information onto graphical representations to gain insight for users and to help them understand the presented information.

The human eye allows seeing and interpreting objects or pictures of objects quite well and absorbing huge amount of information from that sight. Unfortunately, humans do not see nearly as well the information contained in masses of numbers or similar data [1]. Figure 1-1 shows the difference between different methods of presenting information.



Hourly Statistics for December 2000

Hour	Hits			Files			Pages			KBytes		
	Avg	Total		Avg	Total		Avg	Total		Avg	Total	
0	40	1258	1.93%	34	1074	2.04%	9	290	2.16%	265	8208	2.12%
1	98	3051	4.67%	24	773	1.47%	84	2632	19.90%	699	21681	5.59%
2	20	640	0.98%	16	498	0.94%	7	241	1.82%	156	4849	1.25%
3	13	405	0.62%	10	329	0.62%	6	210	1.58%	148	4599	1.18%
4	11	346	0.53%	8	255	0.48%	5	160	1.21%	106	3276	0.83%
5	8	270	0.41%	6	210	0.40%	5	163	1.23%	128	3976	1.03%
6	15	491	0.75%	13	406	0.77%	7	243	1.84%	107	3323	0.86%
7	33	1040	1.59%	28	877	1.66%	10	332				
8	49	1542	2.36%	40	1259	2.39%	11	351				
9	68	2119	3.24%	54	1686	3.20%	14	434				
10	110	3417	5.22%	86	2669	5.06%	16	516				
11	138	4301	6.58%	118	3659	6.94%	18	560				
12	140	4347	6.65%	115	3573	6.77%	21	670				
13	150	4677	7.16%	122	3786	7.18%	20	628				
14	191	5942	9.10%	173	5386	10.21%	24	774				
15	176	5457	8.35%	148	4592	8.70%	24	768				
16	144	4485	6.87%	117	3640	6.90%	19	619				
17	151	4701	7.20%	131	4085	7.74%	24	758				
18	111	3468	5.31%	89	2760	5.23%	21	670				
19	105	3258	4.99%	91	2846	5.40%	16	524				
20	91	2845	4.36%	78	2418	4.58%	14	459	3.47%	614	19042	4.81%
21	79	2453	3.76%	65	2033	3.85%	14	438	3.31%	370	11468	2.90%
22	87	2703	4.14%	70	2188	4.15%	13	422	3.19%	471	14599	3.77%
23	67	2106	3.22%	56	1750	3.32%	11	365	2.76%	449	13917	3.59%

Figure 4-1 Different methods of displaying the same data set.

Beyond the obvious differences in color and presentation, a graphic rendering of data helps users getting insight by selectively amplifying and diminishing different elements. The graph hides exact numbers (the reader will see approximated values) but explicitly reveals the differences between presented values and which value is the highest and which is the lowest. The same information could be figured out from the table, but exactness of the numbers makes relative comparisons more compute-intensive for the human brain [2].

Visualization is related to another concept - Human Computer Interface (HCI). HCI is the study of the interaction between humans and computers focused on the human element in computing. It includes all aspects of the human's experience from the obvious ones of screen layout and selection options to concepts such as input and output devices, reliability and accessibility [3]. HCI includes the issue of what the users need from the computer. Since most things people need from a computer actually originate from other people, interacting with people through the computer is also part of HCI. Human interaction means humans working together with humans and the proper medium needs to be built to support and facilitation and catalyze and enhance this. Therefore, communication and collaboration are part of HCI [15].

The purposes of HCI are presentation, navigation and interaction with information in the most effective, usable and functional way. To fulfill its task, HCI uses various visualization techniques. One of the advanced HCI techniques is Virtual Reality. An important advantage of VR is the possibility for users to interact with information presented by the computer in similar ways they are used to from the real world.

Virtual reality is an artificial environment with computer hardware and software and presented to the user in such a way that it appears and feels like a real environment.

It provides a way for people to visualize, manipulate, and interact with simulated environments through the use of computers and extremely complex data [4]. The user can interact with the presented information in natural way using specialized hardware devices such as VR gloves or VR glasses.

Virtual reality can simulate either reality or non-reality based data. When simulating reality, the objective is to recreate the real world as accurately and fully as possible [4].

Virtual reality is recent and is just now being used in many different areas of life. Among other things, VR is used in architecture and mechanical engineering, two fields which are in the area of interest of this thesis.

Today, architects are using VR for visualization and marketing. For example, VR is used as a promotional tool for both the architectural companies and their developer clients in retail and commercial architecture. Very importantly, VR can help architects and clients to better communicate about proposed projects. Clients who are not trained in reading architectural plans may gain the most from the more natural representation offered by VR displays. By making use of how people naturally perceive the world around them, VR technology enables both architects and clients to get a better understanding of a proposed building while it is in the design phase [5]. In Figure 4-2 interior of a virtual building is shown.

Figure 4-2 Inside a virtual building [16].

VR is also successfully used in engineering in preparing prototypes, planning processes for a product and manufacturing. In mechanical design, a solid model or virtual prototype of a product and its associated data can be processed using various software tools to analyze its performance or method of manufacture. Virtual reality can be used to complement this as a tool in collaborative work settings to allow several people to visualize and interact with the designed components and to simulate and verify the behavior of the product [6]. The ability for virtual teams to share a single virtual world would make it possible for participants at different locations to work together. This would allow, for example, a team member to test the fit of the component they are working on with other parts simply by assembling them in a VR world instead of sending drawings, CAD files or even physical prototypes between geographically remote sites. Figure 4-3 shows an example of a mechanical engineering design.

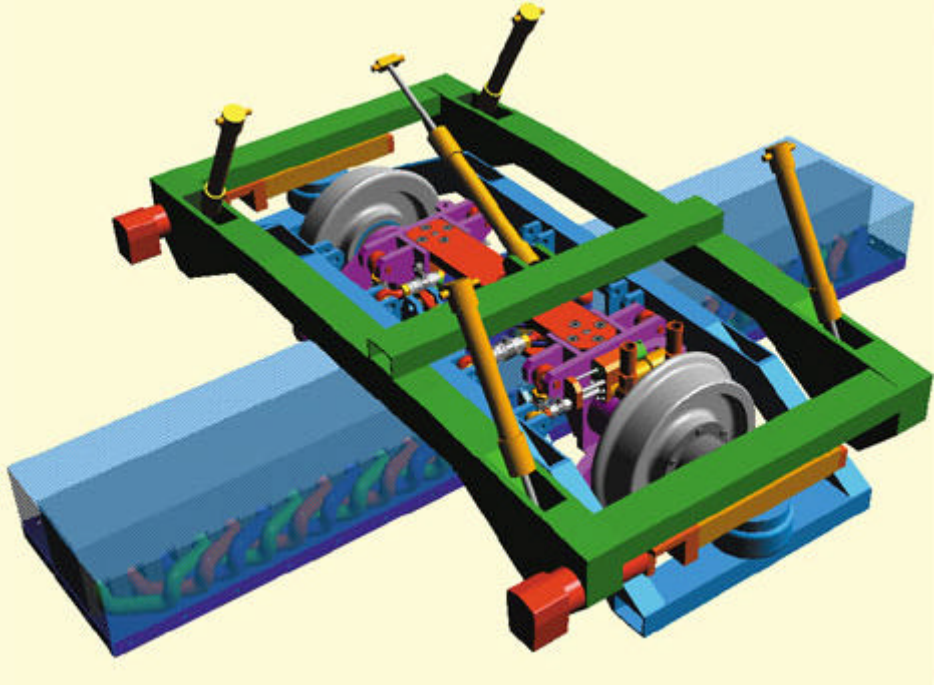


Figure 4-3 An example of a mechanical design in VR [17].

Common applications of VR deal with inherently spatial data such as mechanical, architectural, and scientific data sets. The technology handles a mapping from these domains to virtual worlds.

4.1.2 GIS

Development of computer technology in the last decades allowed to create new computer systems capable of analyzing huge amounts of data, the analysis of which would have been impossible with traditional methods. Geographic Information Systems are an example of such a systems.

A GIS is a computer system capable of capturing, storing, analyzing, and displaying geographically referenced information. Geographic Information System technology can be used for scientific investigations, resource management, and development planning. For example, a GIS might allow emergency planners to easily calculate emergency response times in the event of a natural disaster, or a GIS might be used to find wetlands that need protection from pollution [18]. GIS is an excellent technology to understand and solve problems associated with data whose common attributes are related to location and geography.

In recent years, the use of GIS has grown rapidly in both the public and private sectors. GIS is recognized for its ability to assist governments in economic development, tax assessment, infrastructure management, emergency management, delivery of health and human services, planning and zoning, environmental management, transportation studies and crime analysis. The private

sector is using GIS for such areas as market analysis, transportation routing, and insurance analysis [19].

Figure 4-4 shows the main GIS components. GIS stores two types of information: spatial data which contain the position of geographical reference points, and graphical data of objects which have to be displayed (e.g. points, lines, buildings, pictograms of objects, maps). The spatial data describe the spatial positions of the graphical objects. The central component is a GIS engine that performs all operations (accessing databases, converting data types, coordinate transformations, detecting intersecting areas etc.). GIS functionality is presented to the user through a desktop GIS client.

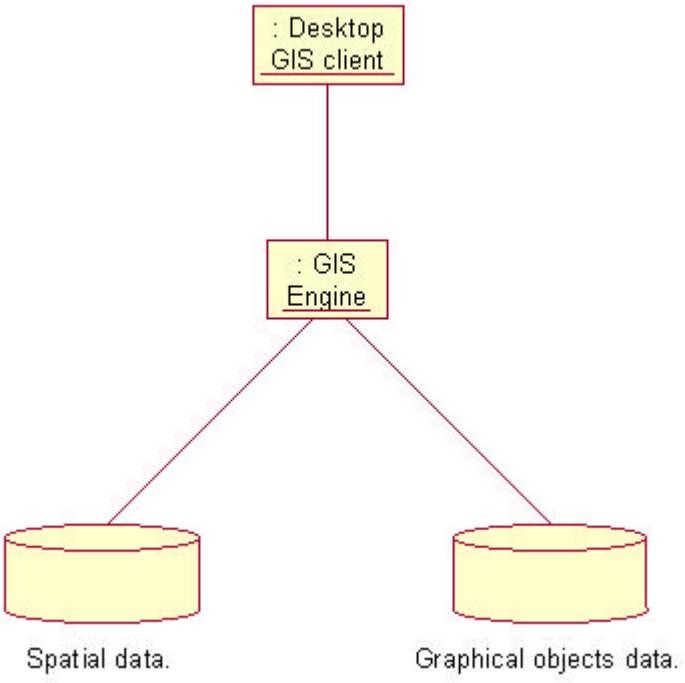


Figure 4-4 GIS architecture.

GIS systems have powerful visual display capabilities that present the results of analysis on maps of a wide variety of scales (Figure 4-5). Also all other types of objects presented in GIS can use different graphical representations in different scales, e.g. a tree could be presented as a simple circle when the scale used is big, and when the scale is smaller it can be represented as an icon showing a pictogram of a tree.

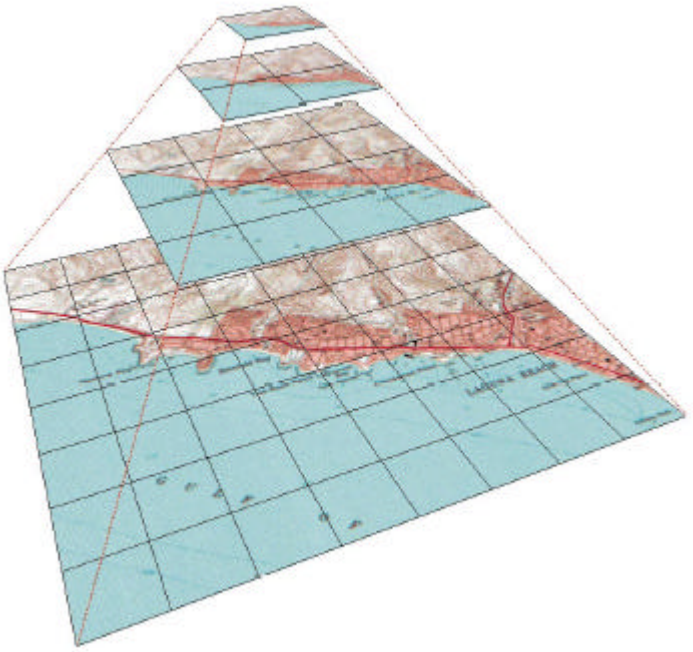


Figure 4-5 Different scales of a map.

There are two methods for the storage of spatial data in Geographical Information Systems (GIS). Raster formats cover a given area with regular pixels and associate attributes (for example, ownership or land use) with individual pixels. Alternatively, vector-topological formats describe the boundaries of irregular regions using strings of vectors. The basic structure of a vector format is therefore the polygon to which attributes are logically attached [7].

GIS makes it possible to link or integrate information that is difficult to associate through any other means. Thus, a GIS can use combinations of mapped variables to build and analyze new variables. Figure 4-6 shows examples for GIS data sources.

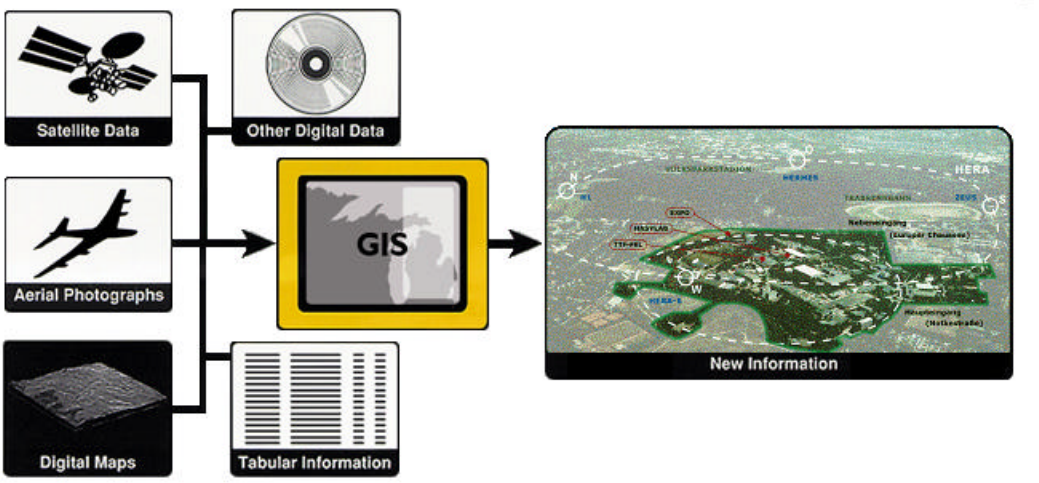


Figure 4-6 GIS data sources.

For most applications in the geographical sciences, two dimensional visualizations like maps are sufficient but in some cases 3D GIS can bring totally new possibilities for GIS users. Advances in computer graphics hardware and algorithms, visualization, and interactive techniques for analysis, offers the components for highly integrated, efficient real-time 3D Geographic Information System. Such 3D GIS is also often referred as Virtual GIS.

Virtual GIS can be used almost anywhere a traditional GIS can be used. In addition, the ability to have detailed 3D views and to jump to a different location to check the view opens new possibilities. Planners for new buildings or other facilities can see full 3D views from their prospective sites or can see the view from nearby existing buildings with their planned facility in place. Urban planners can see the layout of streets, buildings, and parks on their actual topography and can thus evaluate site lines, congestion, where sunlight strikes, etc. Emergency service providers could get immediate 3D views of the areas where they have to respond and, with the addition of appropriate event information in the GIS database, could also see where there are obstructions along their routes due to construction or heavy traffic. Virtual GIS is a general platform on which any of these services could be built [8].

In Figure 4-7 the DESY area is shown in 3D.

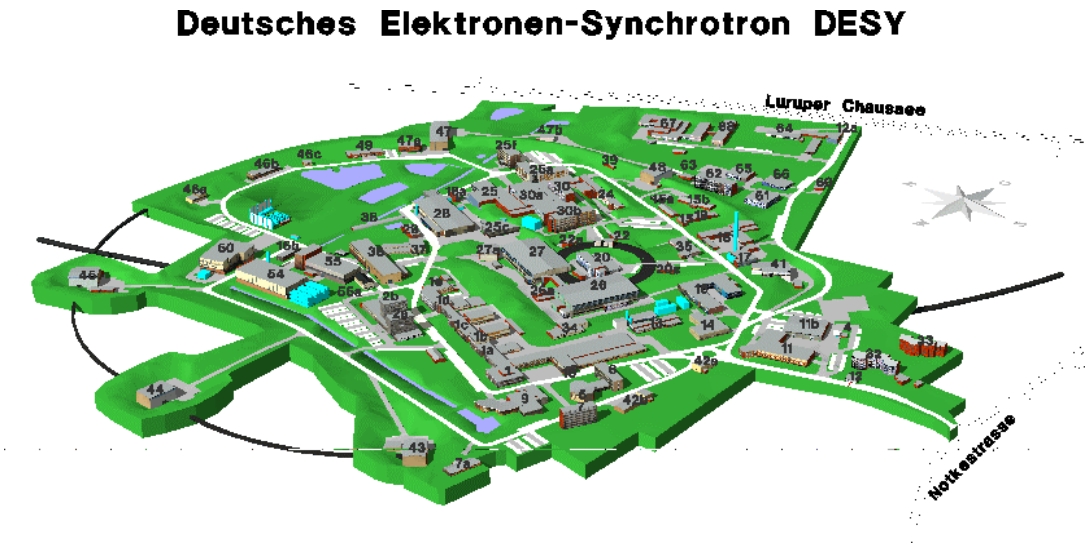


Figure 4-7 DESY area in 3D [20].

With VRML it is possible to create a Virtual GIS which can be accessed through the Internet with an ordinary Web-Browser. Virtual reality provides a suitable graphical environment for navigating spatial data, and most importantly, it can be used to provide a portable and cost-effective graphical user interface for spatial data processing. Traditional Web-Browsers, equipped with suitable plug-ins, are candidates for the next-generation interfaces to GIS systems or, better, to networks of distributed Virtual GIS components.

The flexibility of such systems will allow expert users to access only those data and models really needed by the application, having hopefully, a simpler

communication protocol with GIS components. Virtual reality, as a tool for delivering visual data over the network, will allow a widespread access to environmental information also to a larger community of non-expert users [9].

4.2 Object position information

The spatial position of each object in AVC is described using one of two positioning methods.

The first method describes objects using two reference points (R and L) and the rotation angle (N) with respect to the accelerator beam; reference points R and L set the direction of the beam - this method is used only for the accelerator components. The described method is shown in Figure 4-8.

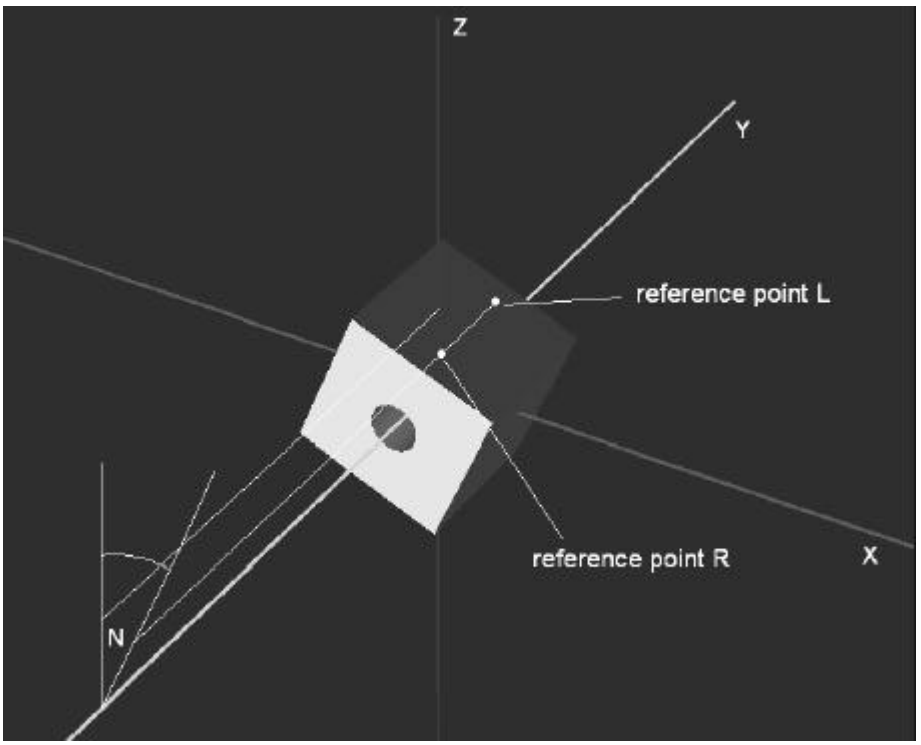


Figure 4-8 The positioning method for accelerator components.

The second method describes objects using three reference points (R, L and S) - it is used for the non-accelerator components (Figure 4-9).

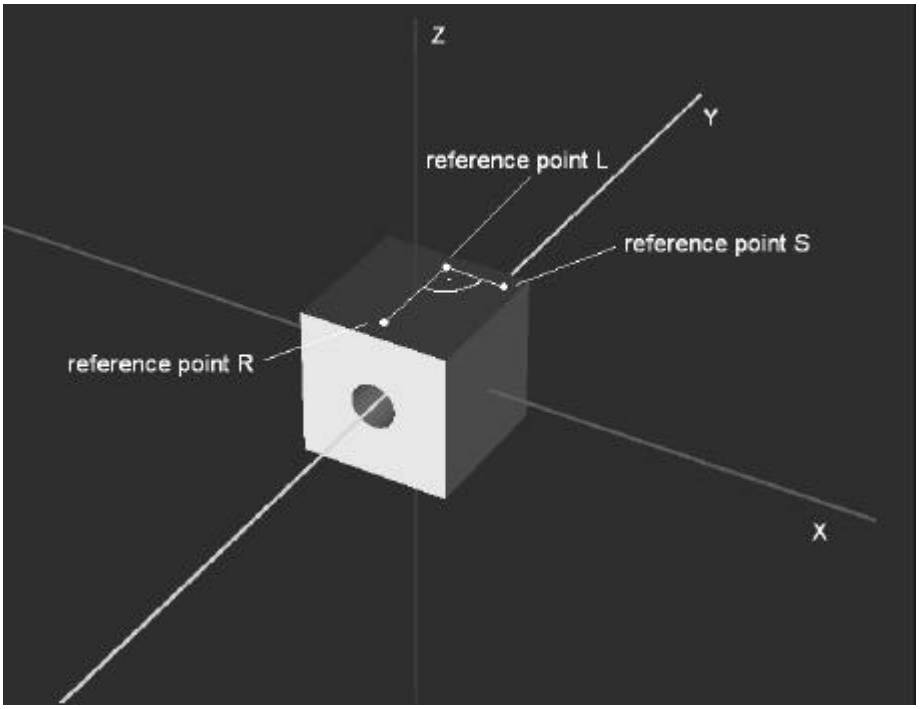


Figure 4-9 The positioning method used for non-accelerator components.

Reference points are always located at the top surface of the 3D object regardless of the used method.

The position information of objects consists of two lists called:

- *elementlist*
- *picklist*

The *elementlist* is a list of all types of 3D object representations used in AVC. Each object type is described in its own coordinate system using one of the previously described positioning methods (for the accelerator components the rotation angle N is always equal zero in the local coordinate system). An example of an *elementlist* is shown in Figure 4-10.

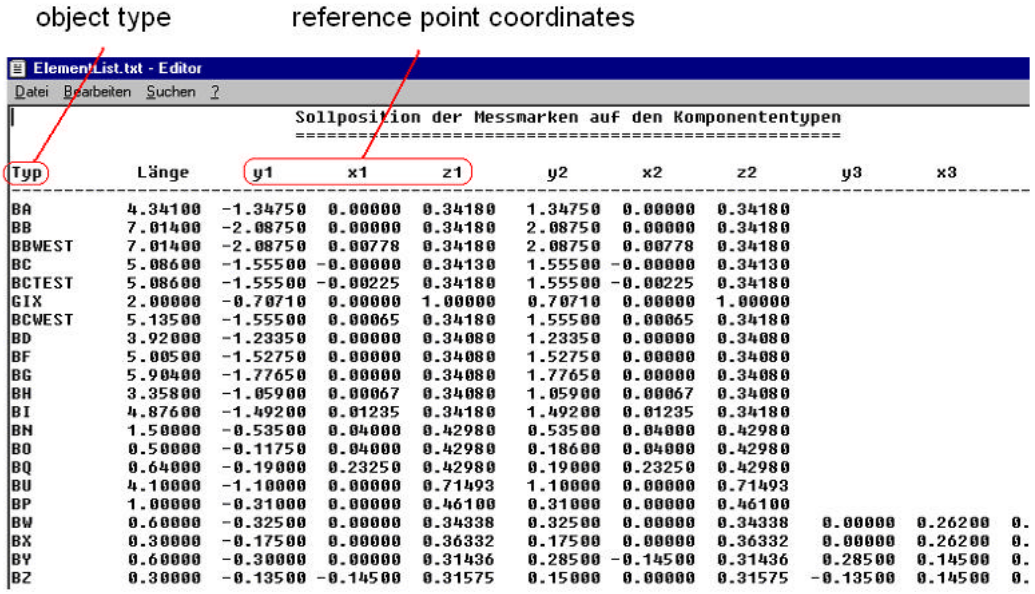


Figure 4-10 An example of the *elementlist*.

The *elementlist* contains the following columns:

- **Typ** – object type
- **Laenge** – object lenght
- **y1,x1,z1** – coordinates of the reference point #1
- further columns contain the coordinates of remaining reference points

The second list is called *picklist*. It contains a list of all objects located in the GIS selection box. All objects are described in the absolute coordinate system (the one used in GIS). An example of the *picklist* is shown in Figure 4-11.

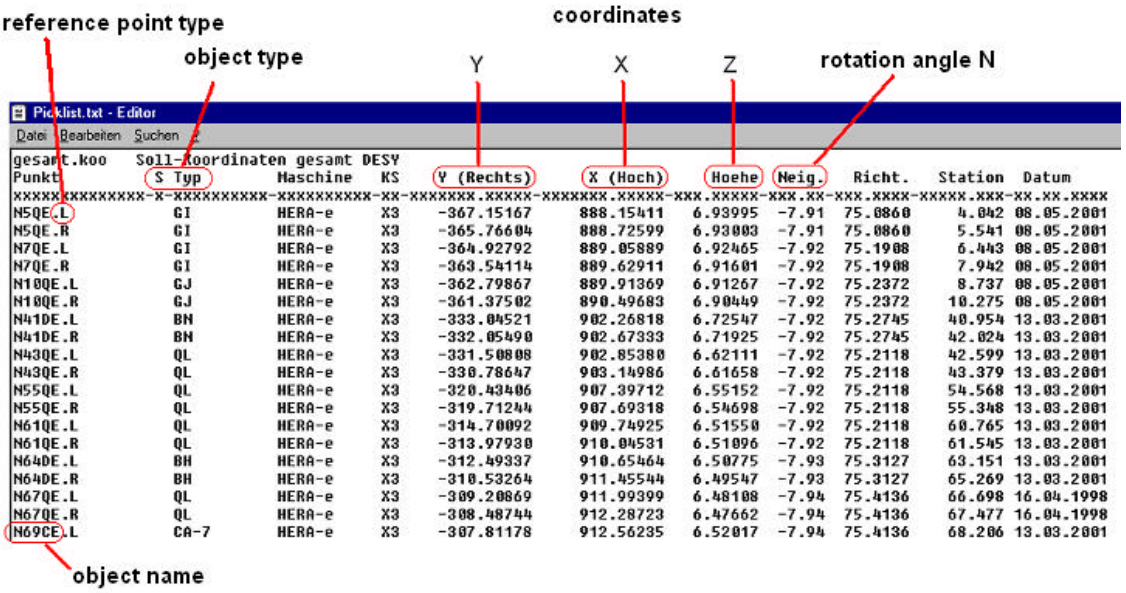


Figure 4-11 An example of the *picklist*.

The *picklist* contains the following columns:

- **Punkt** – object name followed by dot and the type of the reference point
- **S Typ** – type of object (it corresponds to column **Typ** in the *elementlist*)
- **Maschine** – not used in AVC
- **KS** – not used in AVC
- **Y (Rechts)** – y coordinate of the reference point
- **X (Hoch)** – x coordinate of the reference point
- **Hoehe** – z coordinate of the reference point
- **Neig.** – rotation angle
- further columns are not used in AVC

4.3 System architecture

The architecture of AVC consists of many components. It includes both, native and external components. The external components provide the input data for AVC, native components perform the operations and supply the result data. Figure 4-12 shows the dependencies of the AVC components.

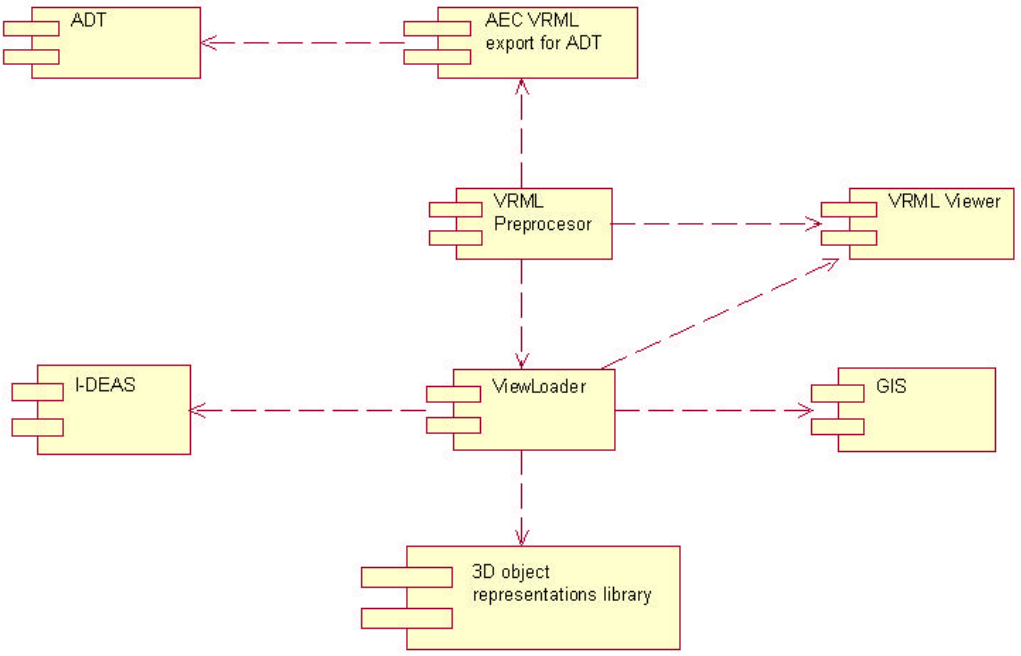


Figure 4-12 AVC component diagram.

The external AVC components are:

- I-DEAS
- ADT
- AEC VRML export for ADT
- VRML viewer
- GIS

The native AVC components are:

- ViewLoader
- VRML preprocessor
- 3D object representations library

Each component is responsible for some of AVC's requirements. Figure 4-13 shows the AVC architecture and explains how components interact with each other and lists, which requirement is handled by which component.

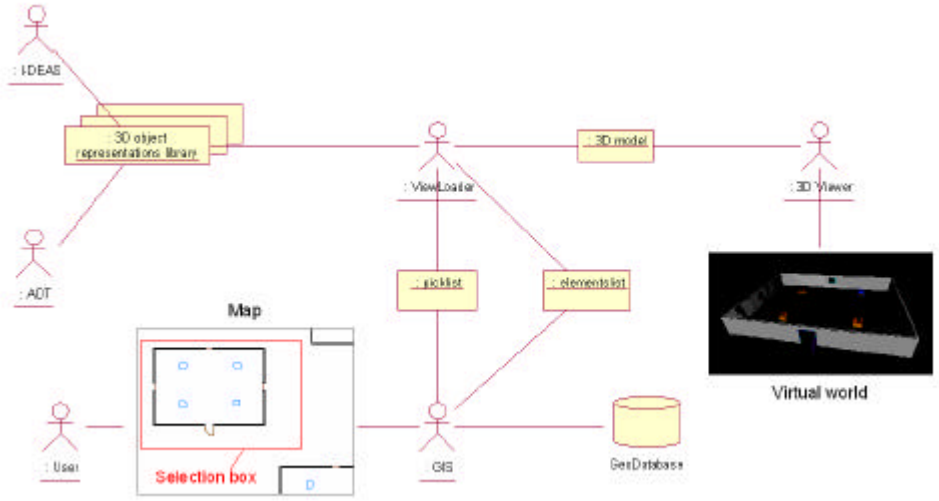


Figure 4-13 AVC architecture.

The most important user requirement, UR1, is fulfilled by the GIS. The GIS offers the possibility to select areas on maps for 3D visualization. The selection is available using a functional graphical user interface. The GIS cooperates with the GeoDB database which contains the spatial coordinates of all the objects which are present on a map. The GIS generates a list of all objects inside the selection box, including their spatial coordinates.

ADT and I-DEAS are the source of 3D data. ADT provides architectural CAD models and I-DEAS provides mechanical CAD models. To export 3D geometry, ADT needs to cooperate with external add-ons. All 3D object representations are stored as VRML files.

ViewLoader is the core component of AVC. Using data from the all other components, ViewLoader automatically generates virtual worlds (UR2). As input data ViewLoader uses information about the position of objects as delivered by the GIS, and 3D geometry data delivered by I-DEAS and ADT. ViewLoader supports the two positioning methods described in section 4.2 (UR6).

The resulting virtual world has VRML format and it can be viewed directly in the VRML viewer. It is also possible to convert the VRML to a JT and use the JT viewer. Both 3D viewers are responsible for the interaction between user and virtual world (UR4).

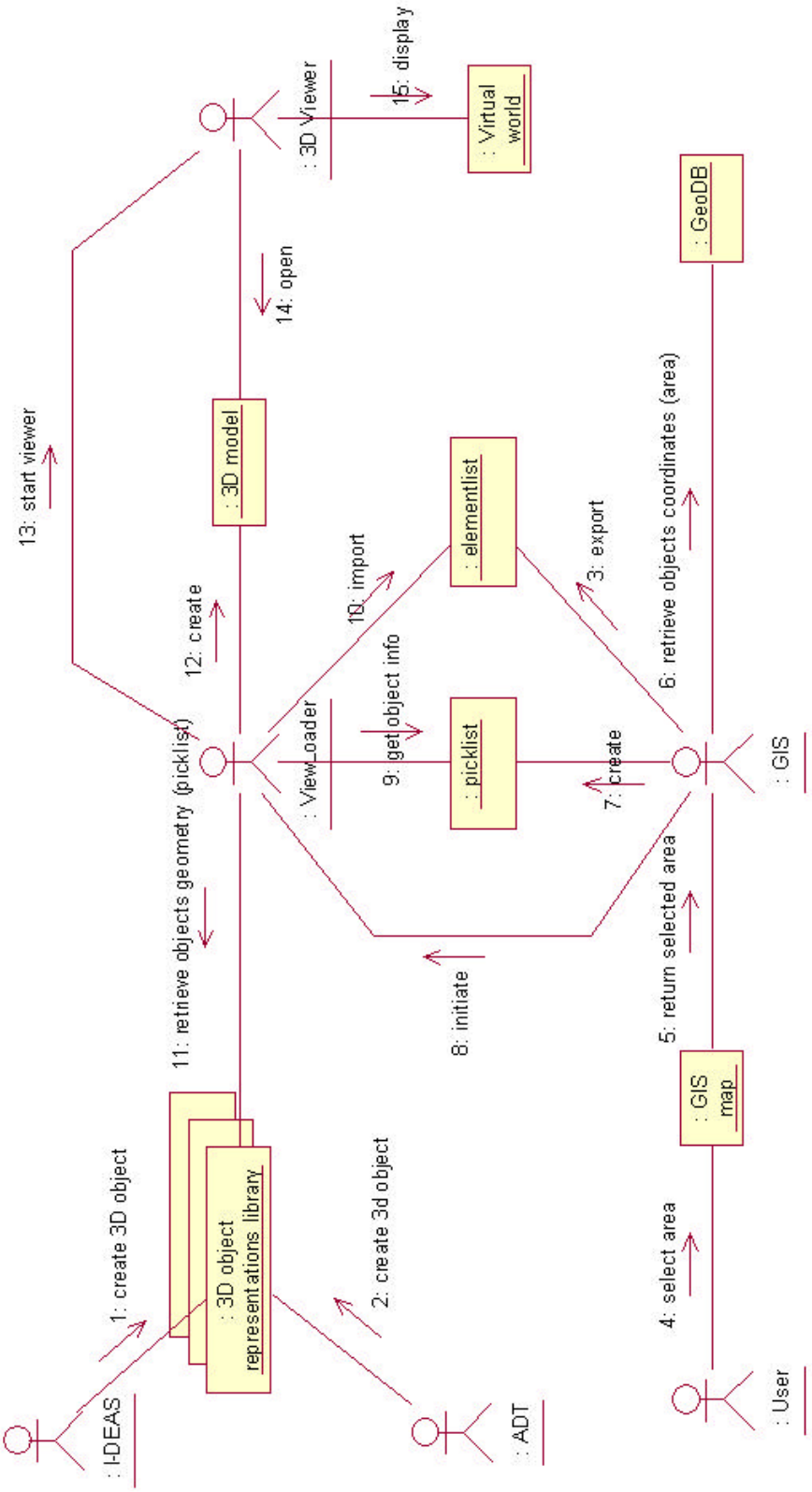


Figure 4-14 AVC collaboration diagram.

Figure 4-14 shows the AVC collaboration diagram. Before AVC can start its operation three preparation steps must be done:

- all 3D object representations of magnets and other mechanical components have to be exported from I-DEAS and added to the 3D object representations library;
- all 3D object representations of buildings have to be exported from ADT and added to the 3D object representation library;
- a list of all objects containing their coordinates in a local coordinate system (*elementlist*) has to be exported from GIS;

When these three steps are completed AVC is ready to start its operation.

Because AVC will be used by two different user groups AVC operation will be explained using two sequence diagrams showing separate scenarios for each group (Figure 4-15 and Figure 4-16). Each group has different needs and therefore each will use different viewers for the virtual world. Casual users need only to see the virtual world, and they will use the VRML client (probably CosmoPlayer) to access the virtual world directly through the Web-Browser. Engineers need to perform additional tasks (measurements, collision analysis etc.) and they will use a JT viewer called VisView [22].

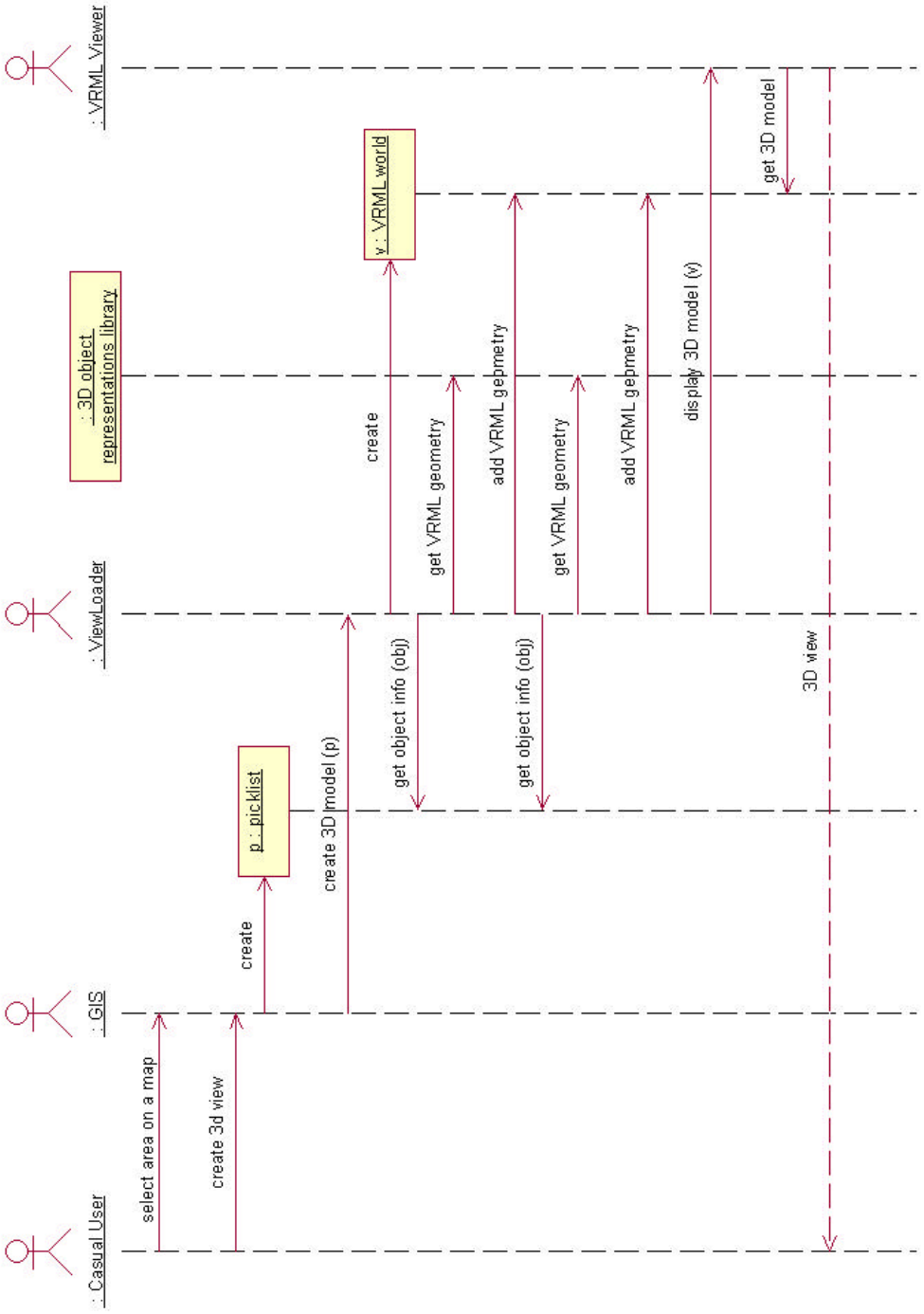


Figure 4-15 Low-level scenario of the *View 3D model* use case for casual users.

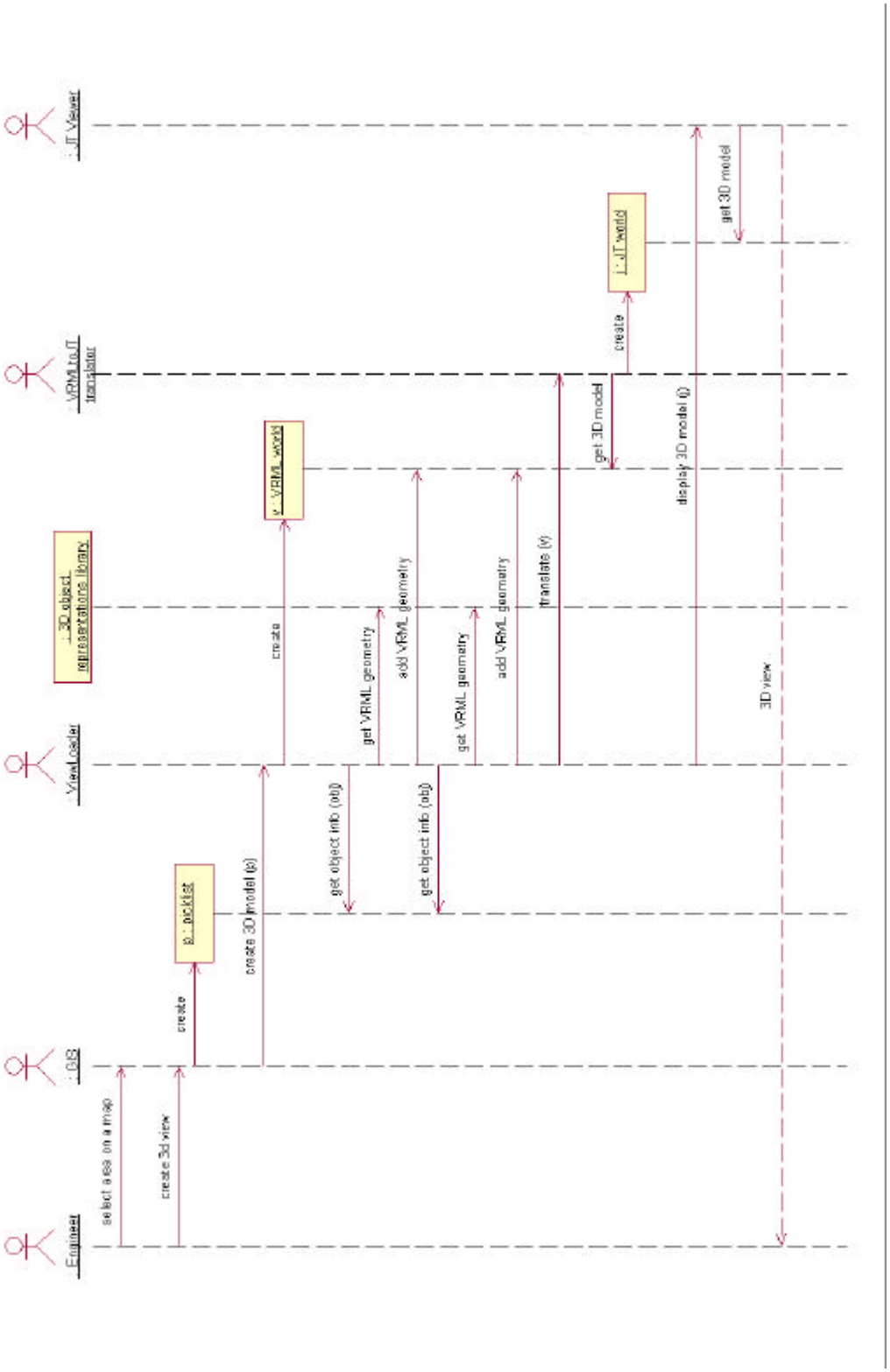


Figure 4-16 Low-level scenario of the *View 3D model* use case for engineers.

The sequence diagram on Figure 4-15 shows the *View 3D model* scenario for casual users. The scenario starts when the user selects a rectangular area of a GIS map and confirms the choice by the command “Create 3D view”. The GIS uses this selected map area as input information for producing the *picklist*. For each object from the selected area the GIS adds a single entry to the *picklist*. When the *picklist* is created ViewLoader can start constructing the virtual world. For each entry in the *picklist* ViewLoader adds a single object to the virtual world. The created VRML world is then visualized by the VRML client.

The sequence diagram on Figure 4-16 shows the *View 3D model* scenario for engineers. At the beginning the sequence of actions proceeds as in the previously described scenario. This time however, after the VRML world is created by ViewLoader, a VRMLtoJT translator has to convert the VRML world to JT. When the conversion is complete, the virtual world is displayed by the JT viewer.

The important aspect of the AVC architecture is the reusability of the 3D object representations. AVC uses a set of single instances of 3D objects to create the virtual reality. Figure 4-17 explains how AVC constructs virtual worlds.

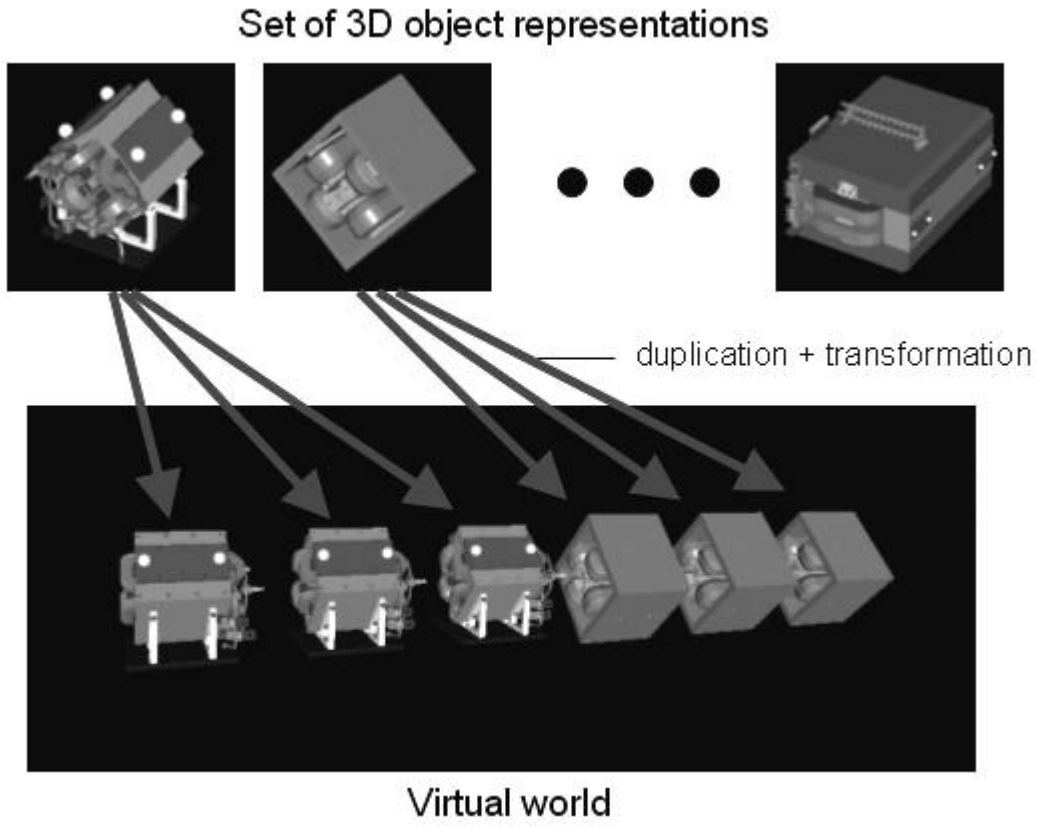


Figure 4-17 AVC functioning principle.

Each type of object is stored in AVC only as a single instance and every time this particular type of object has to be used it is duplicated and placed at the specific spatial position. The creation of the virtual world consists of a series of

duplication operations and placing the duplicated objects at their specified positions. When the created virtual world is no longer needed it can be destroyed. There is no need for storing the whole virtual world as it is possible to create on the fly the desired fragment of it from previously prepared objects. This way, the amount of data which have to be stored is minimized.

4.4 ViewLoader

ViewLoader is the core component of the AVC. The other AVC components serve mostly as data sources, whereas ViewLoader performs most of the operations in AVC.

The main task of ViewLoader is the construction of virtual worlds. The process is fully automated and no interaction with the user is required. ViewLoader has to be provided with two types of data:

- a set of 3D object representations supplied by CAD systems
- information about the spatial position of objects (*picklist* and *elementlist*) supplied by the GIS

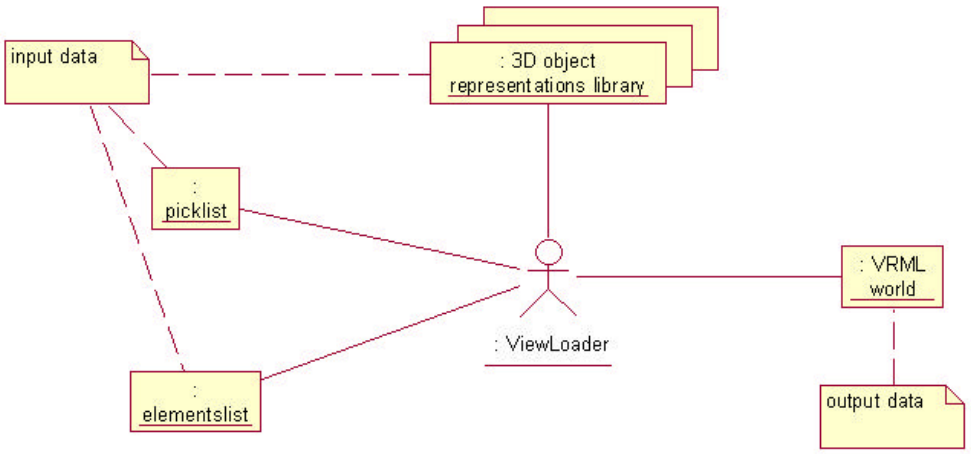


Figure 4-18 ViewLoader's input and output data.

ViewLoader uses 3D object representations in the VRML format. The output 3D models produced by ViewLoader are also in VRML format. Figure 4-18 shows ViewLoader's data sources and data output.

The set of VRML 3D object representations and the *elementlist* are constant – they change only when new object types are added or removed from AVC. For each entry in the *elementlist* a corresponding VRML object representation has to exist.

The *picklist* is the element which decides about the content of the virtual world. For each entry in the *picklist* a corresponding VRML object exists in the created virtual world.

The ViewLoader operation consists of a series of actions performed for every object included in the *picklist*. These actions are shown in Figure 4-19.

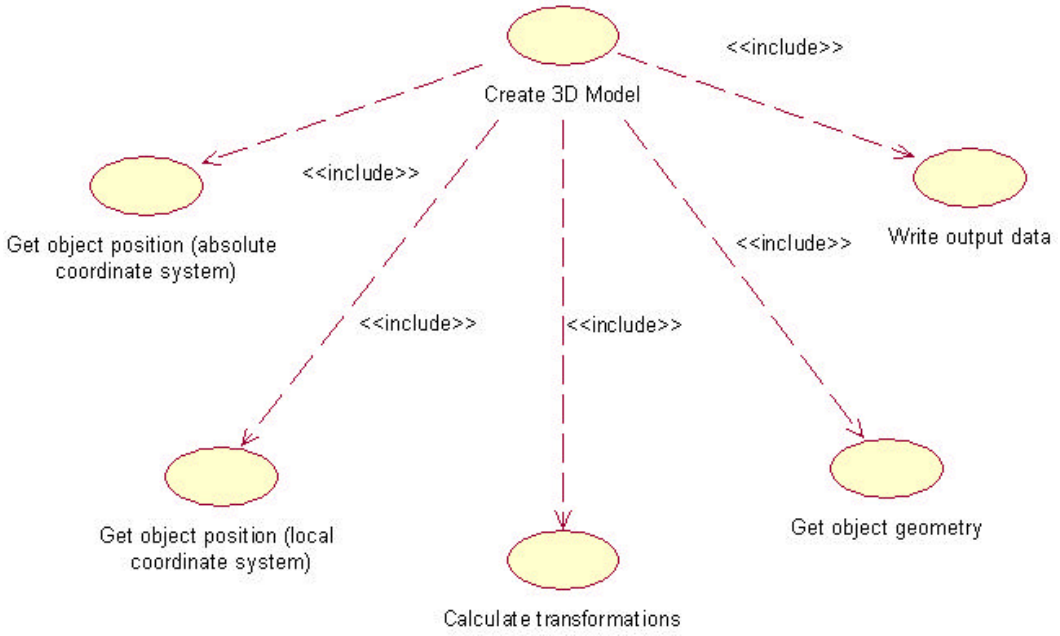


Figure 4-19 The set of operations performed during construction of a virtual world.

The diagram on Figure 4-20 explains the order of execution of these actions. At the beginning, information about the object is being read from the *picklist*. Information about the reference points' absolute coordinates and rotation angle are stored. For each object type included in the *picklist*, the corresponding information about the object is searched in the *elementlist*. The information about the position of reference points in the local coordinate system is being read from the *elementlist* and stored. At this point, the whole information about the spatial object position is known, and ViewLoader calculates the transformations from the local to the absolute coordinate system. In the next steps ViewLoader reads the object geometry from the VRML source file, adds previously calculated transformations, and writes the modified object geometry to the output file. This set of operations is repeated for each object in the *picklist*.

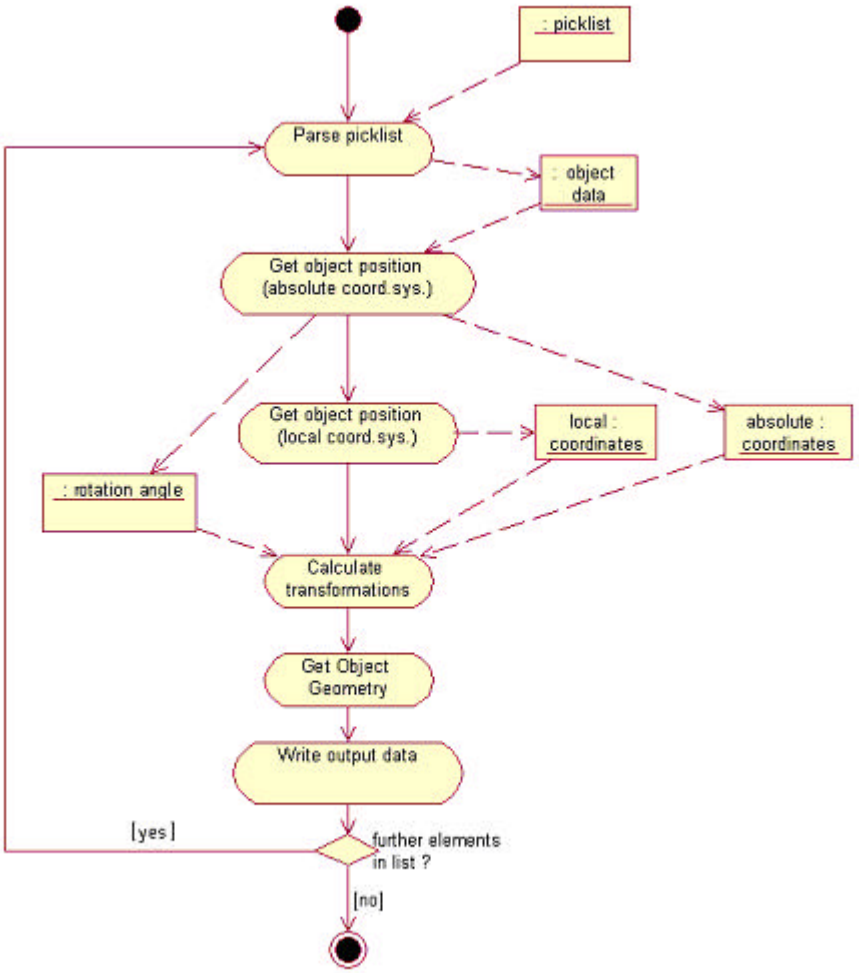


Figure 4-20 High-level ViewLoader's activity diagram.

5 Implementation

5.1 Used technologies and tools

The following section describes the technologies and tools used for implementing AVC.

5.1.1 Autodesk Architectural Desktop

Autodesk Architectural Desktop is an object oriented solution based on AutoCAD. The aim of ADT is to provide a solution that encompasses the entire architectural design process, from concept, to modeling, planning, analysis, detailing, documentation and presentation. ADT is capable to support the workflow of the design process to completion. Figure 5-1 presents the ADT user interface.

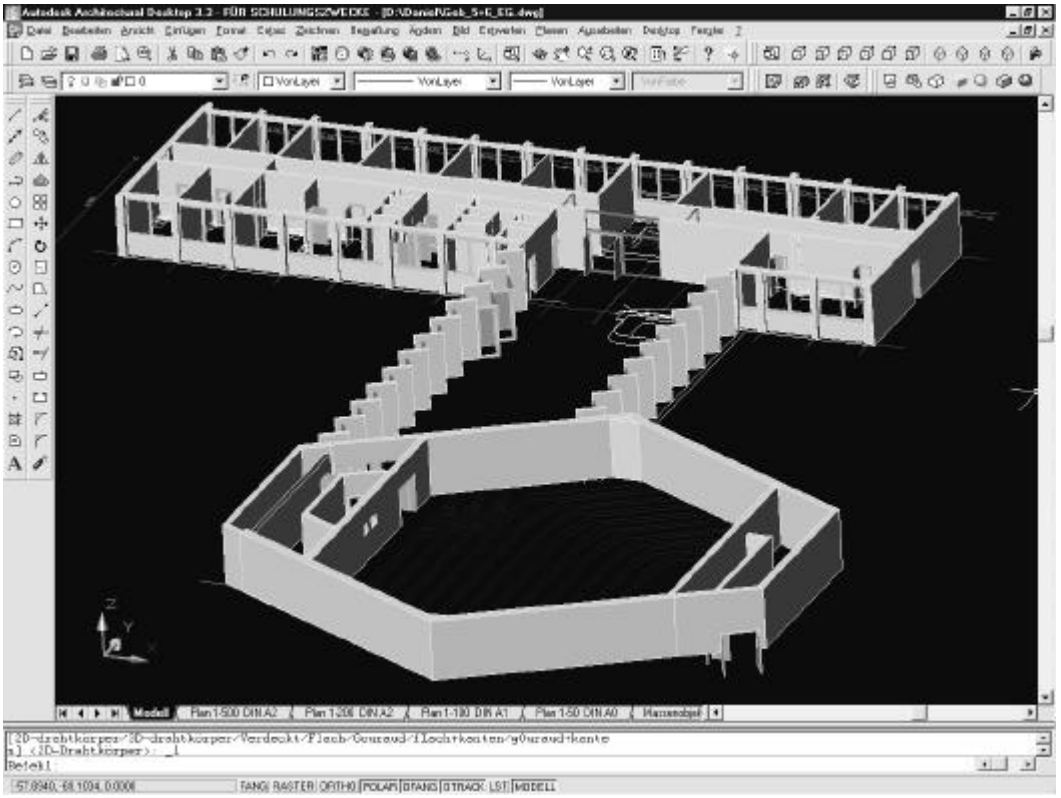


Figure 5-1 ADT user interface.

ADT combines AutoCAD drafting tools with architectural objects. Consequently, it can do anything that AutoCAD can do, and its user interface is much like AutoCAD's. It also uses AutoCAD commands for many basic drawing and editing operations. Using an object oriented approach, individual building objects relate intelligently with one another in a way that was not possible in traditional CAD environment. Object maintain data across a building life cycle and perform more functions without the need to reconstruct them. ADT uses intelligent architectural objects that behave according to real-world properties. Among these objects are walls, windows, stairs, and roofs.

ADT’s role in AVC is to provide 3D object representations of buildings which are present at the DESY area. Created designs are stored as VRML files. At present moment these designs are not available and they must be created in the near future.

5.1.2 I-DEAS

I-DEAS (Integrated Design Engineering Analysis Software) is an integrated package of mechanical engineering software tools. This software was designed to facilitate a collaborative concurrent engineering approach to mechanical engineering product design and analysis. It allows different groups in

a company to share design geometry and exchange information freely for a variety of applications [23]. I-DEAS is composed of many different „application“ modules, including: Design (Solid Modeling), Drafting, Simulation, Test, Manufacturing, Management, and Geometry Translators.

The modules are subdivided into lower-level tasks, e.g. Design application includes tasks for part modeling, drafting, assembly modeling, mechanism design and others.

All these modules share a common repository and I-DEAS enables sharing design geometry between different groups. There are several ways these tools may interact as a design evolves. Figure 5-2 presents the I-DEAS user interface.

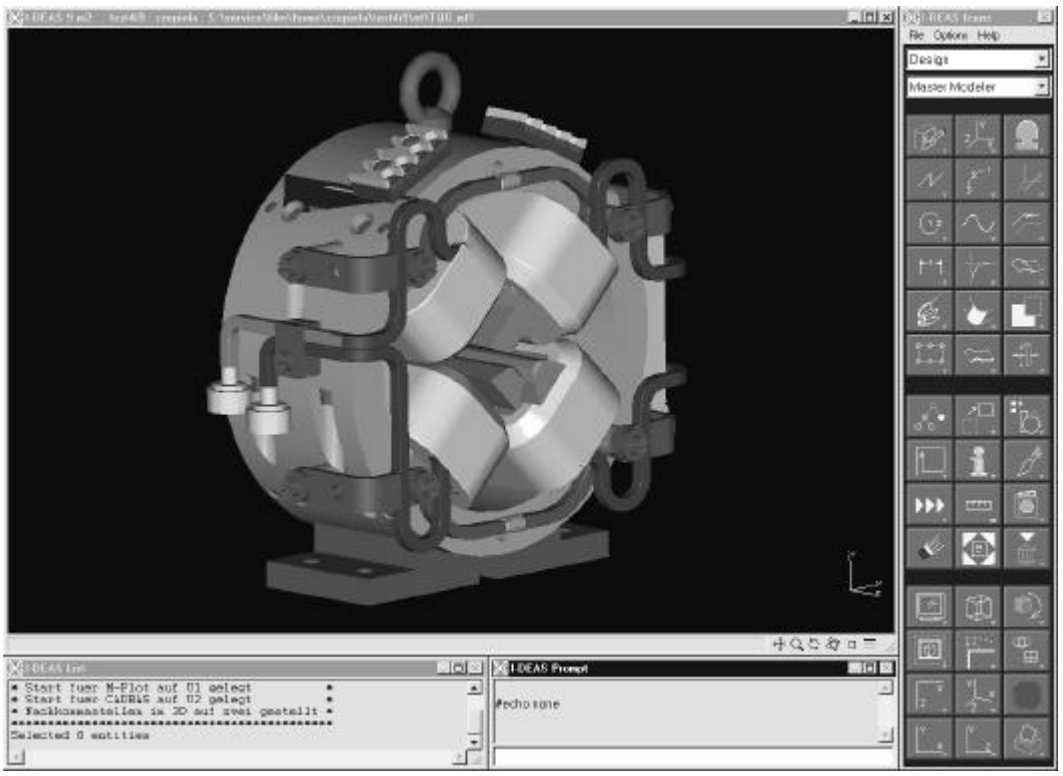


Figure 5-2 The I-DEAS user interface.

The philosophy behind I-DEAS is to encourage collaborative engineering by offering an integrated set of design automation tools. Each of the applications can be used by itself, but the real advantages occur when these tools are used together allowing closer communication between different disciplines working together in a design project [23].

AVC needs I-DEAS as a source of mechanical CAD models. The 3D object representations of magnets and other mechanical components are created and exported from I-DEAS to VRML files. These files are stored as an input data for ViewLoader.

5.1.3 VRML

VRML is an acronym for "Virtual Reality Modeling Language". The first release of the VRML 1.0 specification was created by Silicon Graphics, Inc. and based on the Open Inventor file format. The second release of VRML added significantly more interactive capabilities. It was primarily designed by the Silicon Graphics VRML team with contributions from Sony Research, Mitra, and many others. VRML 2.0 was reviewed by the VRML moderated email discussion group (www-vrml@vrml.org) and later adopted and endorsed by many companies and individuals. In December 1997, VRML97 replaced VRML 2.0 and was formally released as International Standard ISO/IEC 14772 [24].

The VRML is a file format for describing interactive 3D objects and worlds. VRML is designed to be used on the Internet, intranets, and local client systems. VRML is also intended to be a universal interchange format for integrated 3D graphics and multimedia. VRML may be used in a variety of application areas such as engineering and scientific visualization, multimedia presentations, entertainment and educational titles, web pages, and shared virtual worlds [10].

VRML has three main features which distinguish it from other 3D graphic tools and they decide about its attractiveness:

- availability – virtual worlds can be accessed through the Internet at any time what mean that it can be accessed by a very wide community
- possibility to move inside a virtual world – it is possible to see a virtual scene from any viewpoint, the user can decide how to move camera in a real time
- interactivity – the user can interact with objects, move them and change their state like e.g. turning the light on or off by clicking on a virtual button

A virtual world can be created in different ways:

- it can be constructed by VRML programmers as an ASCII text file
- it can be created in applications like 3D Studio and then converted to VRML format
- it can be created directly using applications like V-Real Builder, Home Space Designer or VRCreator

Three concepts are basic in VRML: nodes, fields and events. Nodes, which represent the entities in the scene graph, can be geometric primitives, appearance properties, sounds, videos, lights, different grouping nodes, sensors, and interpolators. Nodes can contain other nodes (some types of nodes may have

children) and may be contained in more than one node (they may have more than one parent) but a node cannot contain itself. Fields are node attributes and can be used to store everything, from a single number to a rotation matrix. Events model an architecture in which nodes can communicate with each other (message-passing mechanism) in the scene graph. This allows animation and interaction [9]. Figure 5-3 shows a sample VRML source code describing a scene that contains a green cube (*emissiveColor 0 1 0, specularColor 0 1 0*) of size (4,4,4), translated along vector (0,0,-5) and rotated 0,5 radian about the X axis.

```
#VRML V2.0 utf8

Transform {
    rotation    1 0 0 0.5
    translation 0 0 -5
    children [
            Shape {
                appearance Appearance {
                            material Material {
                                        emissiveColor 0 1 0
                                        transparency  0.2
                                        specularColor 0 1 0
                                    }
                           }
                geometry Box {
                     size 4 4 4
                }
            }
    ]
}
```

Figure 5-3 A sample of the VRML source code.

These interactive behaviors with the scene are implemented with the VRML animation. VRML allows to define sensors in the scene and to create event paths (ROUTE statement) between the sensors and the geometric objects, associating the event in the virtual world with particular behaviors. These sensor nodes are able to sense changes of the device mouse (*TouchSensor* node), and they can be associated to a particular geometry in the scene [9].

VRML is used in AVC as a main format of 3D data. All CAD designs are stored as VRML files. Also the main component of AVC, ViewLoader, uses this format to create virtual worlds.

5.1.4 CosmoPlayer

CosmoPlayer is the VRML browser from Cosmo Software. CosmoPlayer is a plug-in to Web-Browsers which makes it possible to navigate and manipulate 3D words and objects created with the VRML97 language. It is available for Windows and Mac-OS operating systems. Users can explore virtual worlds using a simple and intuitive interface (Figure 5-4).

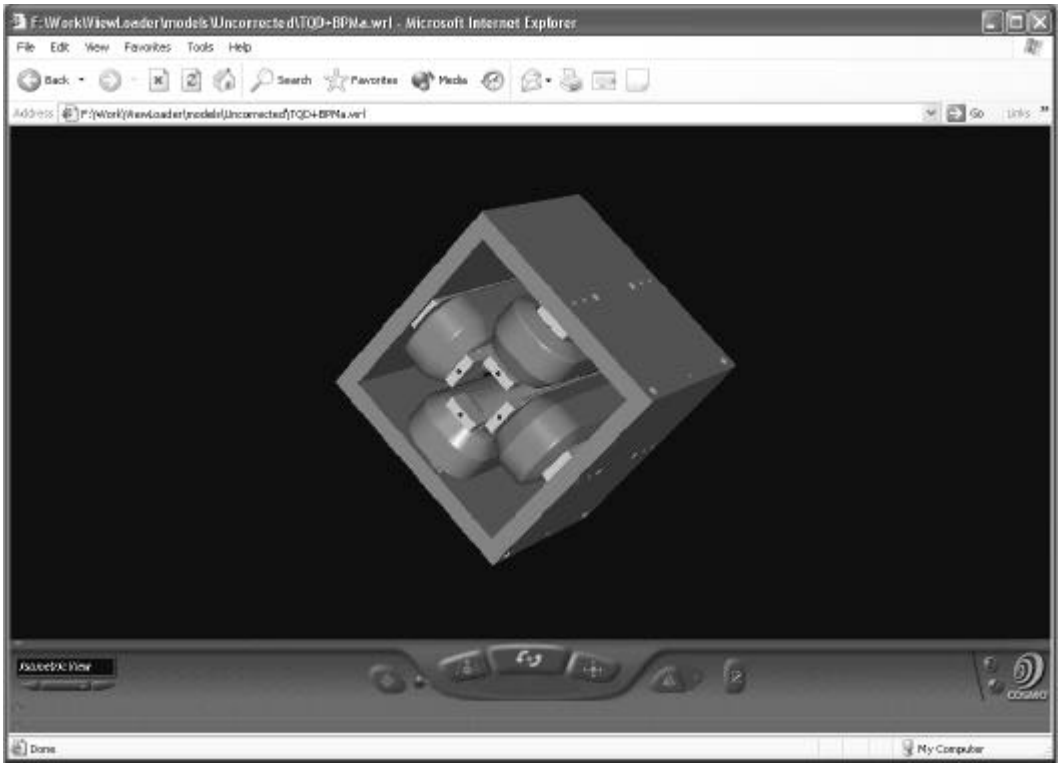


Figure 5-4 A sample 3D object in CosmoPlayer.

CosmoPlayer is able to use two different rendering APIs. It can use either OpenGL or DirectX enabling it to be used on a variety of platforms. CosmoPlayer has probably the most effective rendering engine of the available VRML clients.

CosmoPlayer offers users the basic interaction capabilities (graphical interaction) with the scene: rotating, zooming in/out and panning the model in the scene, clicking an object to move closer to it, moving around in the three-dimensional world using the flythrough commands. Since the user can easily get disoriented by switching from one control to another, a set of pre-chosen viewpoints (*ViewPoint* node) can be selected.

CosmoPlayer is one of the 3D viewers used in AVC. It is used by casual users to view virtual worlds created by ViewLoader.

5.1.5 VisView

VisView is a 3D-viewer by EDS (Electronic Data Systems) which is tightly integrated with many types of product data. The viewer shows a tree-view of the product breakdown structure (PBS) and displays views (mock-up) of the components which are selected in the PBS.

VisView enables users to easily interact with digital product data and to view the latest drawings, 3D models and documents in a single visual environment [25].

VisView has a high-performance rendering architecture which allows users to view large models with satisfactory performance. VisView uses many techniques to increase the rendering performance, e.g. when moving the model, very small details are represented only as bounding boxes, what greatly reduces amount of computations which are necessary to display complicated and large models. Figure 5-5 presents the VisView user interface.

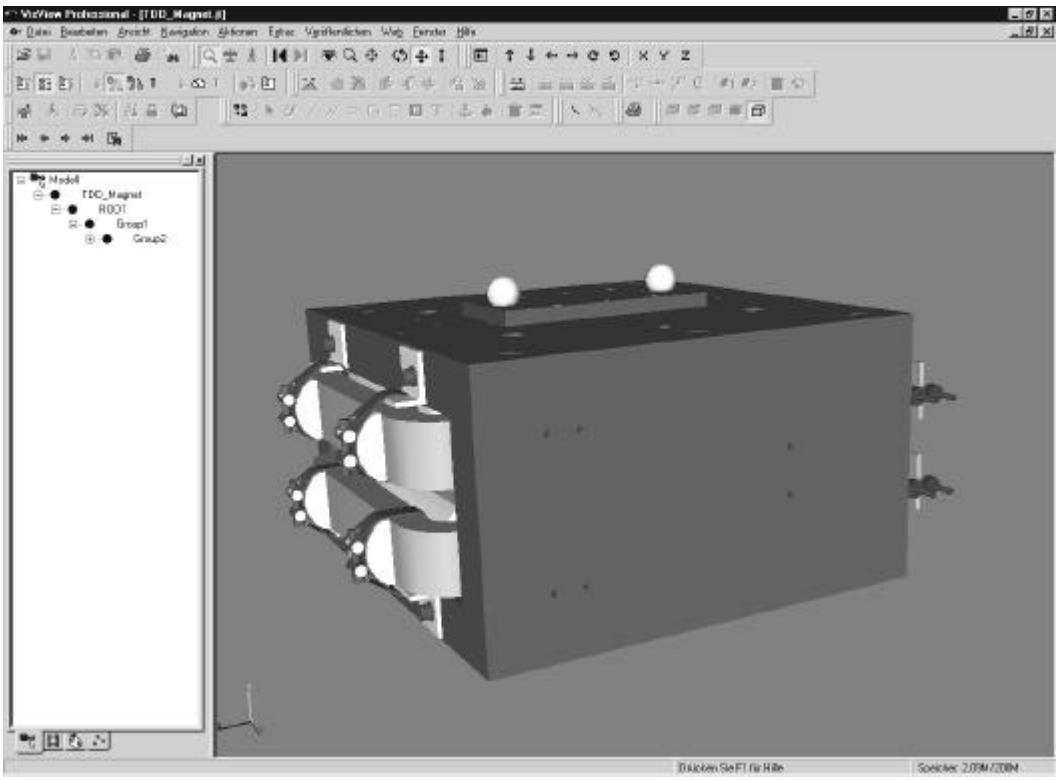


Figure 5-5 The VisView user interface.

VisView is available in two versions: Standard and Professional. The standard version is the base solution for viewing 2D drawings and 3D models, the professional version adds advanced measurement and analysis capabilities. The base functionality of VisView can be also expanded by many add-on modules which are available for this product.

Because VisView is a powerful viewer which offers good performance and analysis capabilities it is used in AVC as a 3D viewer for engineers.

5.2 VRML preprocessor

VRML files from different sources are not always compatible and they cannot be directly combined. AVC uses two CAD applications as its source of 3D object representations:

- I-DEAS 9
- Autodesk Architectural Desktop 3.3

I-DEAS cooperates with two different VRML translators, basic and extended. The basic VRML translator exports the geometry with default settings and offers only few simple options. The advanced VRML translator offers many additional options and features like special effects (e.g. fog, horizon), compression, conversion one type of geometry to another (e.g. splines to polylines), precision settings etc. Currently AVC uses the basic VRML translator because it produces simpler 3D object representations and also because none of the advanced features is used in AVC.

ADT is not able to export the 3D geometry to VRML or any other industrial standard format. To export 3D data ADT needs to cooperate with external add-ons. One of such add-ons “EAC VRML Export”, allows ADT to export the geometry to VRML format.

Because AVC has to use more than one source of 3D data (UR3) the VRML preprocessor is essential for the AVC operation. The VRML preprocessor is used to prepare 3D object representations from ADT before they can be used by ViewLoader. 3D object representations from I-DEAS do not require any preprocessing because they are directly supported by ViewLoader.

Both applications produce slightly different VRML output files and at least one of them has to be modified to be compatible with other one.

With different CAD application it may be required to modify files from all sources. Fortunately in AVC it is enough when VRML files from ADT will be modified, files from I-DEAS can be used directly without any changes. Figure 5-6 illustrates this concept.

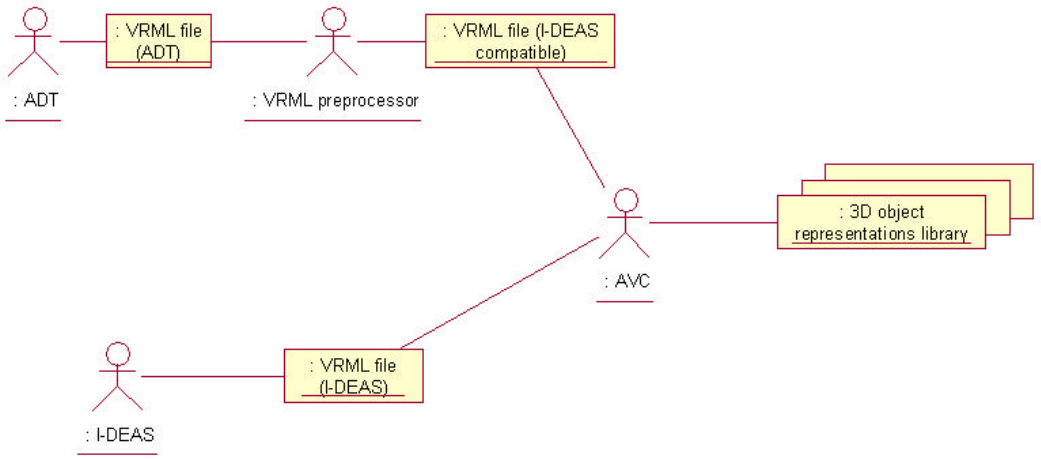


Figure 5-6 The preparation of VRML files.

There are three required modifications for ADT VRML files. The first one involves minor changes of text coding (removing blank spaces, changing sign of end of the line) to the one used in I-DEAS VRML. Fragment samples of VRML files from both systems are shown in Figure 5-7.

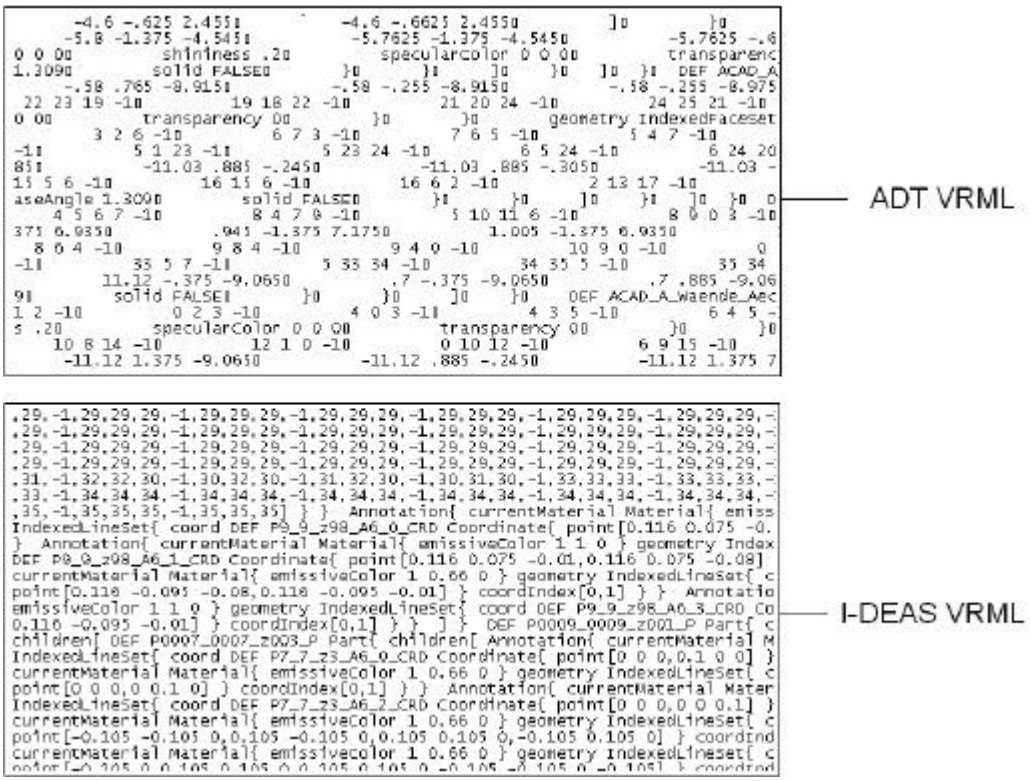


Figure 5-7 Fragments of ADT and I-DEAS VRML files.

The second modification involves adding definitions of objects used in I-DEAS VRML but not included in ADT VRML. These definitions are included in the header of I-DEAS VRML files (Figure 5-8).

```
PROTO Entity [ field SFNode geometry NULL
field SFNode currentMaterial NULL ] { Group { children [ Shape { appearance Appearance { material IS currentMateria
} geometry IS geometry } ] } } PROTO Face [ field SFFloat ambientIntensity .2 field SFColor diffusecolor .8 .8 .8
field SFColor emissiveColor 0 0 0 field SFFloat shininess .2 field SFColor specularColor 0 0 0
field SFNode geometry NULL field SFFloat transparency 0 ] { Group { children [ Shape { appearance Appearance {
material Material { ambientIntensity IS ambientIntensity diffusecolor IS diffusecolor emissivecolor IS emissivecolor
shininess IS shininess specularColor IS specularColor transparency IS transparency } } geometry IS geometry } ] } }
PROTO Annotation [ field SFNode geometry NULL field MFNode BBChild [Group{}] field SFNode currentMaterial NULL ] {
Group { children [ Shape { appearance Appearance { material IS currentMaterial } geometry IS geometry } Group {
children IS BBChild } ] } } PROTO Part [ field SFVec3f translation 0 0 0 field SFRotation rotation 0 0 1 0
field MFNode children [Group{}] ] { Transform { translation IS translation rotation IS rotation children IS children
} PROTO BBoard [ field SFRotation rotation 0 0 1 0 field SFVec3f translation 0 0 0
field SFVec3f axisofRotation 0 0 0 field MFNode children [Group{}] ] { Transform { rotation IS rotation
translation IS translation children [ Billboard { axisofRotation IS axisofRotation children IS children } ] } }
PROTO Assembly [ field MFNode configList [Group{}] ] { Group{ children IS configList } } PROTO Configuration [
field MFNode children [Group{}] ] { Group{ children IS children } } PROTO configInstance [
field SFVec3f translation 0 0 0 field SFRotation rotation 0 0 1 0 field MFNode children [Group{}] ] { Transform{
translation IS translation rotation IS rotation children IS children } }
```

Figure 5-8 Definitions of new object types in I-DEAS VRML header.

The third modification involves changing the order of elements inside ADT VRML files. Originally, the navigation information (*NavigationInfo* node) is included at the beginning of ADT VRML file. In I-DEAS this information is contained at the end of file. Because the structures of the VRML files from both CAD sources have to be identical, *NavigationInfo* node has to be moved from the beginning to the end of ADT VRML files.

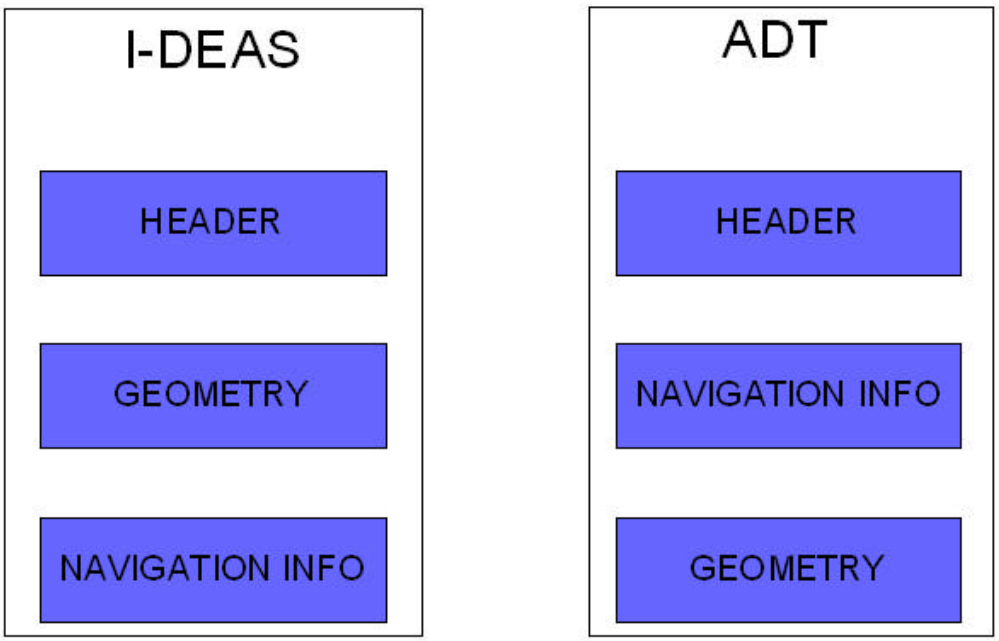


Figure 5-9 The structure of ADT and I-DEAS VRML files.

5.3 ViewLoader

This section presents the ViewLoader's implementation. It explains the internal structure of ViewLoader and the way the VRML is parsed by it.

5.3.1 Internal structure

The ViewLoader's structure consists of several classes which are shown in Figure 5-10.

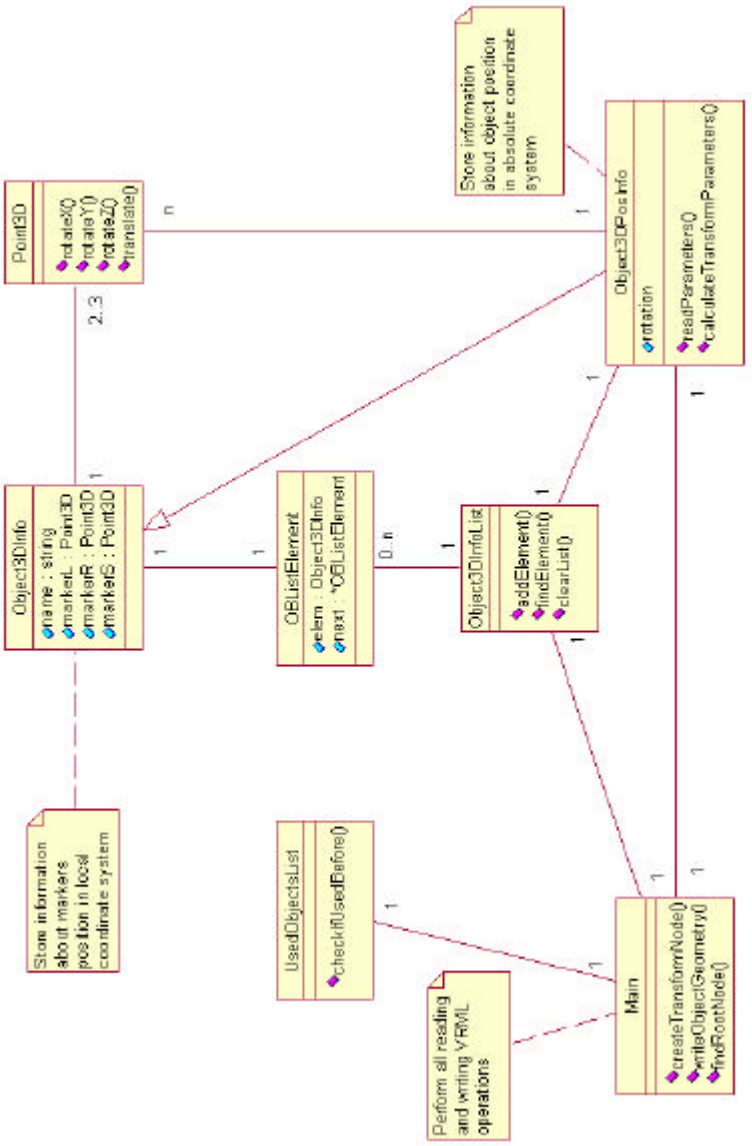


Figure 5-10 The ViewLoader's class diagram.

The class *Point3D* contains methods for all basic geometry operations supported by VRML language:

- rotations around axes
- translation about vector

These operations are often used by methods in other classes. Instances of this class represent reference points in ViewLoader.

The class *Object3DInfo* represents information about objects which is stored in the *elementlist*. This information includes:

- type of the object
- data about position of object reference points in local coordinate system

There are as many *Object3DInfo* instances in ViewLoader as many different types of objects exist in AVC.

Instances of the class *Object3DInfo* are encapsulated inside the class called *OBListElement*. Instances of the *OBListElement* class are used as list elements in class *Object3DInfoList*.

Object3DInfoList class supply methods for:

- searching for a list element
- adding an element to the list
- removing an element from the list

The class *Object3DPosInfo* extends the functionality of the class *Object3DInfo* with:

- attribute *rotation*
- others supporting attributes (not presented on class diagram)
- method *calculateTransormParameters* - used to calculate transformations between local absolute coordinate systems
- method readParameters – used to initialize attributes

Only one instance of this class exists in ViewLoader and it represents the currently parsed object from the *picklist.* When next object is going to be parsed attributes are filled with new values using the *readParameters* method.

The main application thread handle all input/output operations including parsing of VRML files. It also cooperates with list called *UsedObjectList* in order to minimize size of output files. The *UsedObjectList* contains data about all previously used object types. Objects from this list can be reused with a single VRML command without defining the geometry from the beginning (see section 5.2.2).

Figure 5-11 explains how particular classes interact with the input and output data objects, and with each other (high level of abstraction).

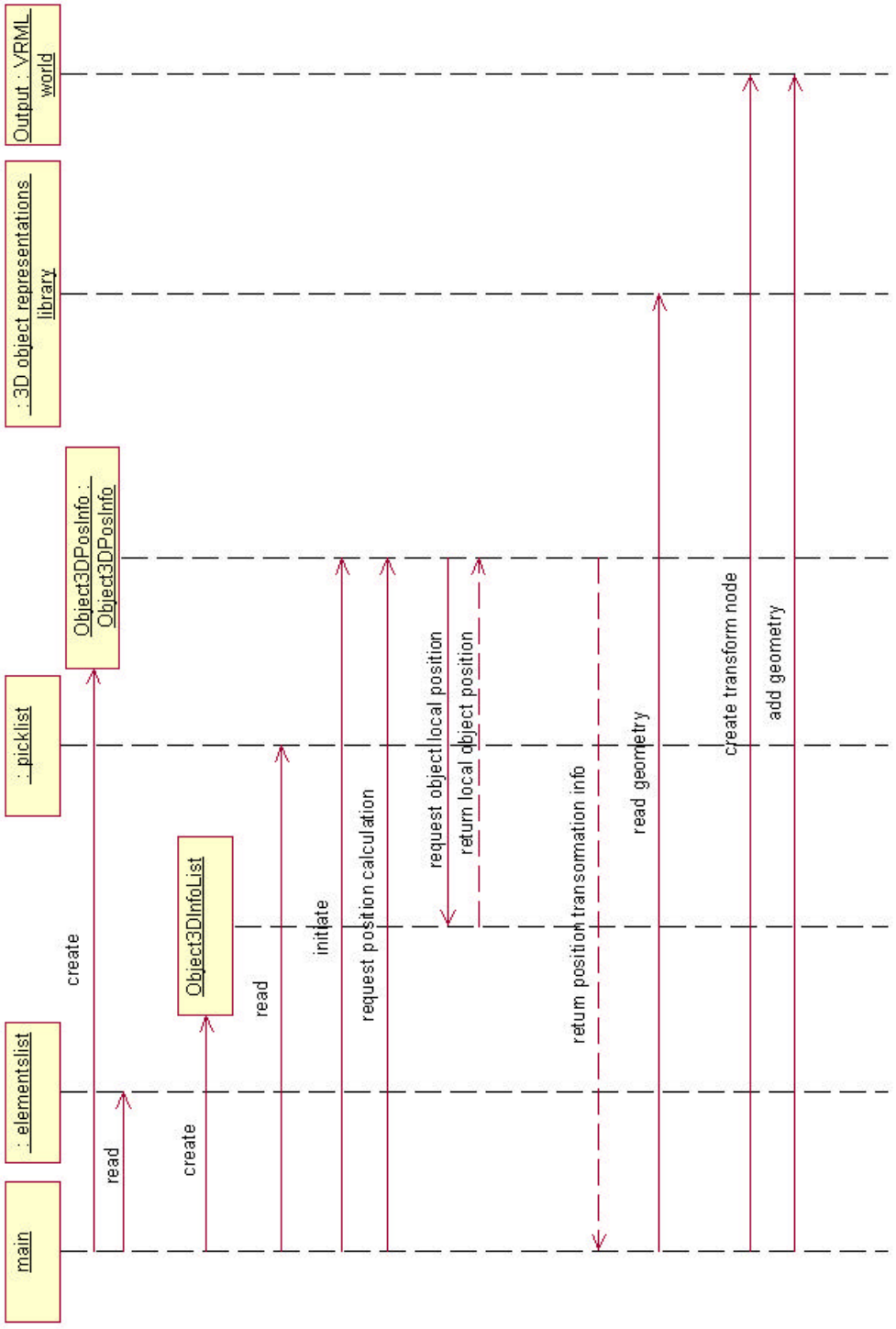


Figure 5-11 High level ViewLoader's sequence diagram.

At the beginning the *elementlist* is analyzed. Using data about position of objects in local coordinate system from the *elementlist,* the instance of *Object3DInfoList* is created (it contains a single instance of Object3DInfo for each entry in the *elementlist*). In the next step data about the position of the object in an absolute coordinate system is read from the *picklist*. The *Object3DPosInfo* instance is initiated with data from the *picklist.* In series of method calls object transformation parameters are calculated. Next, the geometry is read from the 3D object representation library. Finally, ViewLoader creates a series of transform nodes and puts there the object geometry.

5.3.2 VRML parsing

The full VRML specification is very vast and it includes things like animation, special effects (e.g. fog), sounds etc. All these features will not be used in AVC. For ViewLoader the only interesting information is geometry data. That is why there is no need of implementing the full VRML parser what makes the task much simpler. In addition to that, it is possible to take advantage of the fact that VRML is a node language. The geometry is contained inside the set of nested nodes. In order of getting geometry information, the most outer geometry node has to be found. All information inside that most outer node can be treated like a „black box“ (there are some exceptions which will be discussed later). An example of the most outer node is shown in Figure 5-12.

```
DEF BD3 Transform
{
children [
 DEF BD3ACAD_A_Fenster Transform {
 children [
  DEF BD3ACAD_A_Fenster_AecDbWindow_3401 Transform {
  children [
   Shape {
    appearance Appearance {
    material Material {
     ambientIntensity .2
     diffuseColor .4 .698 .8
     emissiveColor 0 0 0
          ...
          ...
          ...

    creaseAngle 1.0472
    solid FALSE
   }
   }
  ]
  }
 ]
 }
]
}
```

Figure 5-12 Information inside B3D node can be treated like "black box".

VRML allows to define new node types, data types and objects. It also supports reusability of objects. I-DEAS VRML files contain set of instructions which define many new nodes types and object definitions. Information about all those new node types and objects has to be included in output file which combine all objects included in the *picklist*.

```
PROTO Assembly
[
        field MFNode configList
        [
                Group{}
        ]
]
{
        Group
        {
                children IS configList
        }
}
```

Figure 5-13 Defining new node type called *Assembly*.

Because ViewLoader uses a set of separate VRML files, where each of the files represents a single 3D-object, there is a danger that different VRML objects will have the same name (I-DEAS VRML translator names entities randomly as series of letters and numbers). Such situation causes that objects are redefined and old objects with the same name are lost. To solve that problem ViewLoader renames all objects by adding different prefix to them. This is also only situation when information inside the most outer geometry node has to be modified in any way.

The possibility of defining objects has a huge advantage, it allows to use the previously defined object with a single instruction instead of writing the whole geometry from the beginning. This feature allows to receive much smaller output files and it also increases performance of ViewLoader by reducing the amount of data to process.

6 Results

To better understand the capabilities of the AVC, two case studies will be presented in the following section. During the work on AVC some difficulties caused by CAD models have been identified. In order to avoid these difficulties in the future, general guidelines for creating CAD models will be presented.

6.1 Case study: Building a virtual world of a workshop

In order to test ViewLoader's capabilities of combining CAD models from different sources a 3D model of a "Virtual Workshop" has been constructed. The "Virtual workshop" 3D model contains four 3D object representations of magnets from I-DEAS and one 3D object representation of a building from ADT. The following paragraphs illustrate how the virtual world is created and which preparatory steps are required. Figure 6-1 shows the three steps needed to create a virtual world.

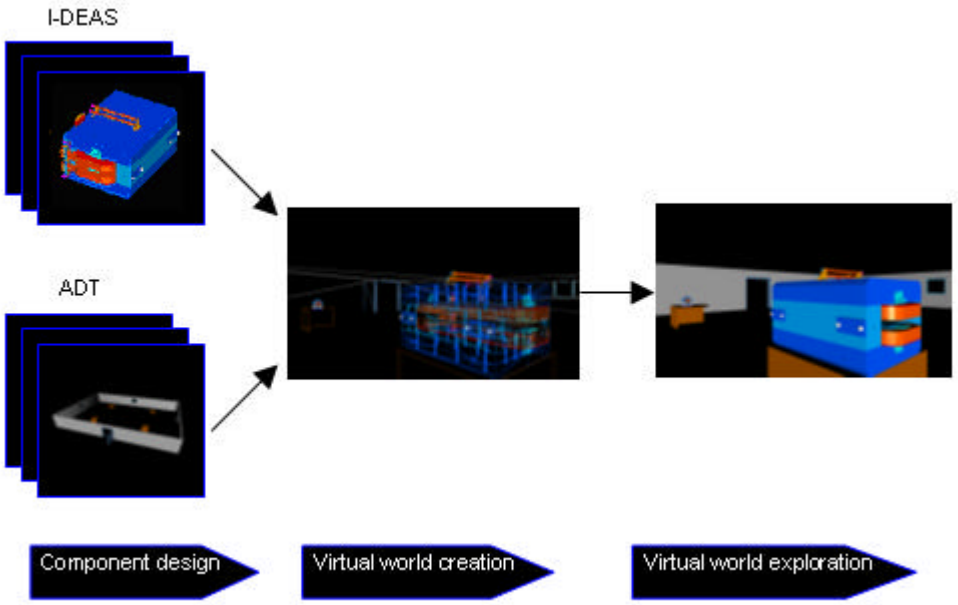


Figure 6-1 Phases of a virtual world construction.

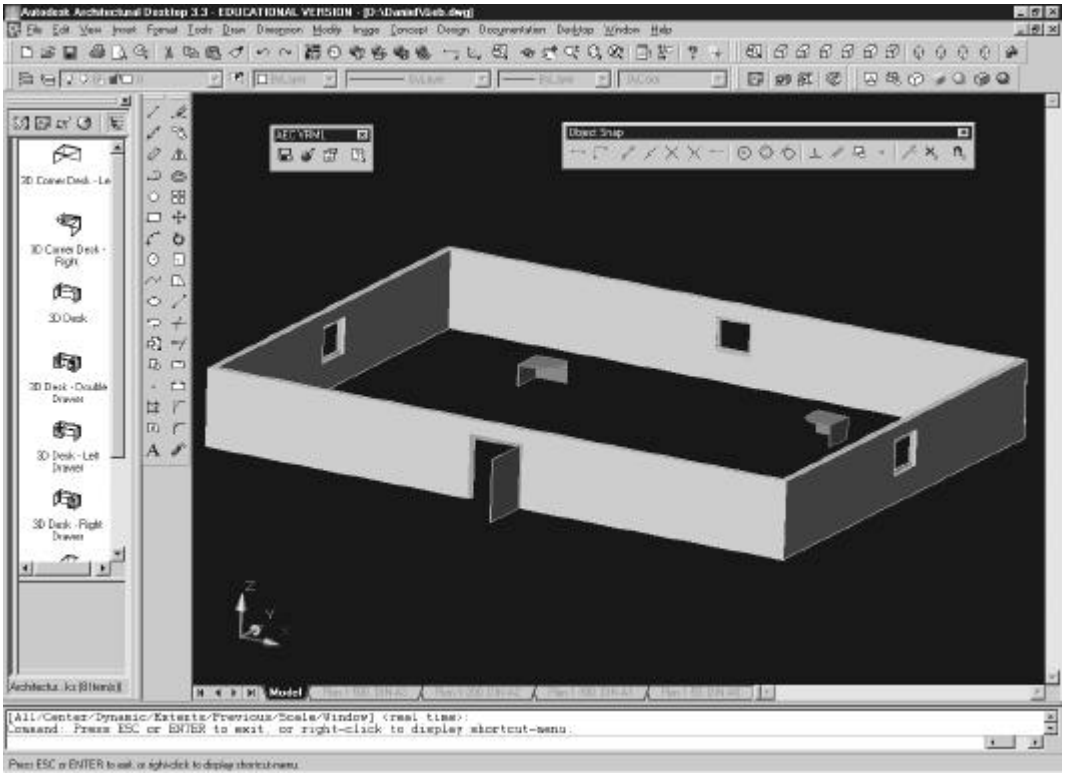


Figure 6-2 Model of "Virtual workshop" building in ADT.

The 3D object representation of the building (Figure 6-2) has been constructed using ADT. The prepared model has been exported from ADT to the VRML using an external add-on “EAC VRML export“. Figure 6-3 presents the export options of the add-on. The geometry can be exported using three different coordinate systems: user coordinate system (Transform to UCS coordinates option), world coordinate system (No transformation option), or centered coordinate system (Center objects option). For the purpose of this case study the centered coordinate system has been chosen because it fits the best to the positioning methods used in AVC. Axis Y and Z have been switched (Flip YZ coordinates option). Other export parameters were set on default.

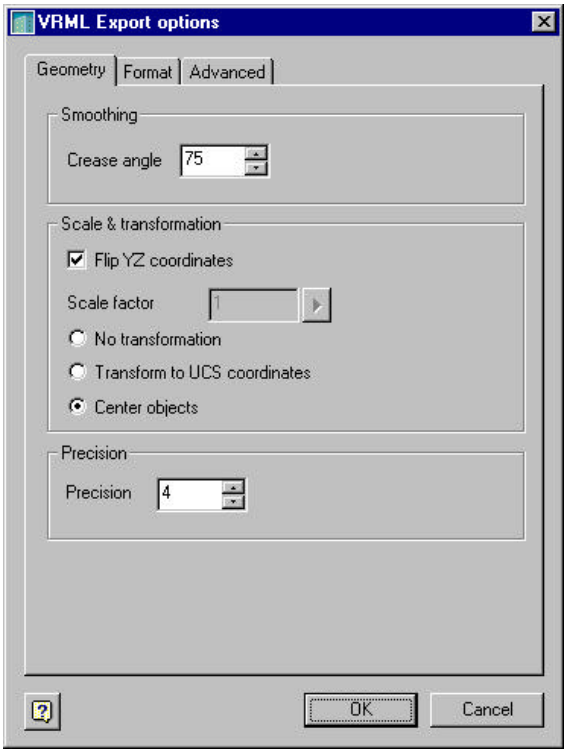


Figure 6-3 ADT VRML export options.

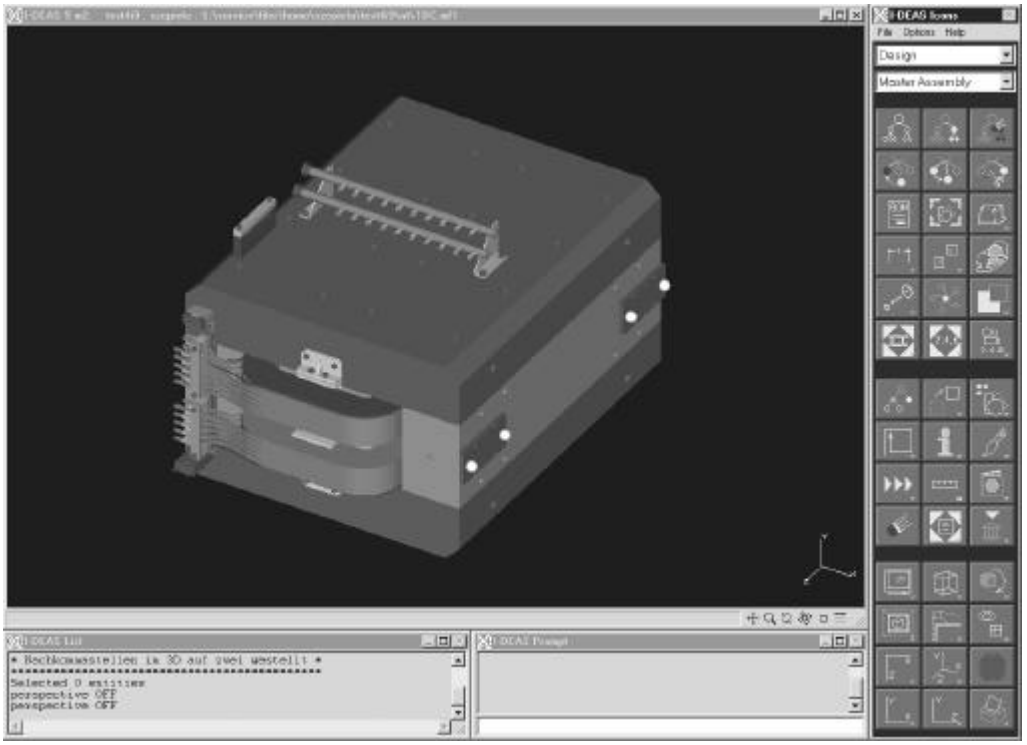


Figure 6-4 One of the magnets used in the "Virtual workshop".

Figure 6-4 shows the CAD model of a magnet presented in the “Virtual workshop”. The 3D object representations of the magnets have been exported from I-DEAS using the “Basic VRML translator” (Figure 6-5).

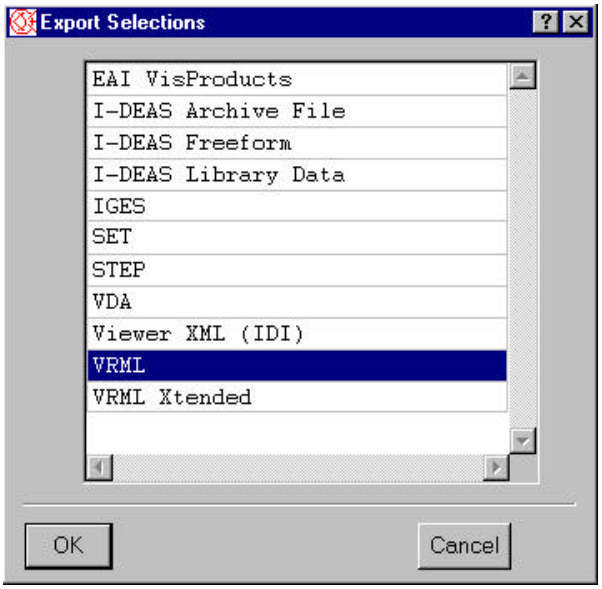


Figure 6-5 The list of available I-DEAS translators.

The translator has been set to “Geometry only” mode in order to produce a VRML output which contains only geometry data (Figure 6-6). The I-DEAS VRML translator does not allow choosing the type of coordinate system or switching axes.

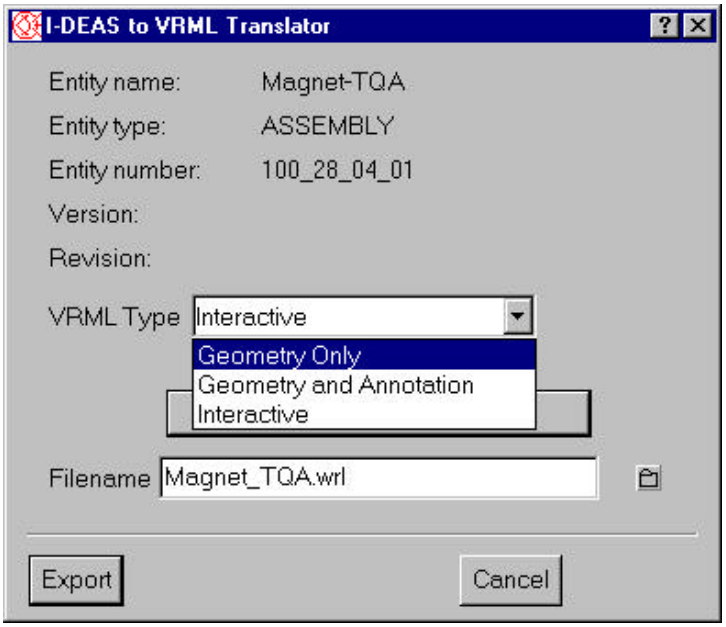


Figure 6-6 The chose of export type.

The *elementlist* and the *picklist* have been prepared manually. The coordinates of all magnets and the building have been determined experimentally. Figure 6-7 presents the *elemetslist* used in the experiment, Figure 6-8 shows the *picklist*.

Sollposition der Messmarken auf den Komponententypen

Typ	Länge	y1	x1	z1	y2	x2	z2	y3	x3	z3
TQA	0.25000	-0.10000	0.10000	0.10000	0.10000	0.10000	0.10000			
TQD	0.30000	-0.10000	0.00000	0.28800	0.10000	0.00000	0.28800			
TDA	0.40000	-0.10000	0.00000	0.17800	0.10000	0.00000	0.17800			
TDC	0.40000	-0.10000	0.00000	0.17800	0.10000	0.00000	0.17800			
GF	1.00000	-0.10000	0.00000	0.00000	0.10000	0.00000	0.00000	-0.10000	0.10000	0.00000

Figure 6-7 The *elementlist* used in the case study.

gesamt.koo Soll-Koordinaten gesamt DESY

Punkt	S	Typ	Maschine	KS	Y (Rechts)	X (Hoch)	Hoehe	Neig.	Richt.
TQAT.L		TQA	TF1	VW	-0.52500	5.65000	2.68000	0.00	75.0860
TQAT.R		TQA	TF1	VW	-0.32500	5.65000	2.68000	0.00	75.0860
TDAT.L		TDA	TF1	VW	-0.52500	-5.30000	-4.72000	0.00	75.0860
TDAT.R		TDA	TF1	VW	-0.32500	-5.30000	-4.72000	0.00	75.0860
TDCT.L		TDC	TF1	VW	-0.37000	5.65000	-4.36500	0.00	75.0860
TDCT.R		TDC	TF1	VW	-0.17000	5.65000	-4.36500	0.00	75.0860
TQDT.L		TQD	TF1	VW	-0.48600	-5.30000	2.98000	0.00	75.0860
TQDT.R		TQD	TF1	VW	-0.28600	-5.30000	2.98000	0.00	75.0860
GFT.S		GF	TF1	VW	-0.10000	0.10000	0.00000	0.00	75.0860
GFT.L		GF	TF1	VW	-0.10000	0.00000	0.00000	0.00	75.0860
GFT.R		GF	TF1	VW	0.10000	0.00000	0.00000	0.00	75.0860

Figure 6-8 The *picklist* used in the case study.

The prepared 3D object representations and position information lists have been used by ViewLoader to create the virtual world. ViewLoader is invoked from the command line using the command:

```
viewloader.exe  <picklist> <elementlist>
```

where <picklist> is the name of the *picklist* file and the <elementlist> is the name of the *elementlist* file. As result the VRML file *output.wrl* has been created.

Following Figure 6-9 shows the resulting virtual world displayed by CosmoPlayer. The virtual world has been interactively explored in order to check if all the objects have been properly created.

Figure 6-9 The virtual world of a workshop.

In the next step, the *output.wrl* has been converted to the JT format using VisView's VRML to JT translator with following command:

```
wrltojt.exe output.wrl
```

As result the JT world which is presented in Figure 6-10 has been obtained. The "Virtual Workshop" world has been interactively explored and analyzed using VisView. The figure shows an example for the extended functionality of VisView – the measures which have been added to the scene.

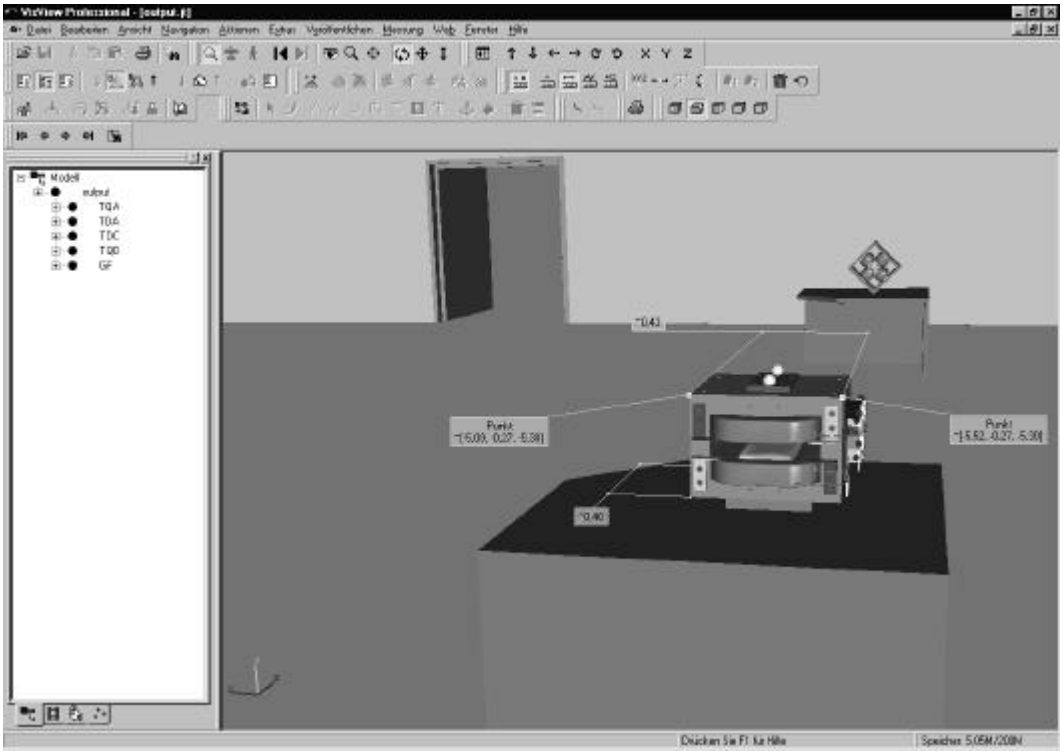


Figure 6-10 The virtual workshop explored using VisView.

6.2 Case study: Visualization of a part of TTF-2

TTF-2 is an accelerator using TESLA technology which has been used as a testbed and which will in the future start operation as a free electron laser. As a one of the results of this thesis a visualization of a part of the 3D model of TTF-2 is presented.

In this case study real TTF-2 accelerator data has been used to create a beam line which consists of a series of magnets. Unfortunately, only a limited number of CAD models for different type of magnets was available at the time of writing this thesis. The CAD models for the tunnel and other types of supporting objects were also not available.

All the 3D object representations have been created in exactly the same way as described in the previous case study.

Both position information lists (*picklist* and *elementlist*) for the TTF-2 3D model have been extracted manually from an Excel worksheet which contains the full data about positions of the TTF-2 magnets.

The 3D model of TTF-2 has been constructed by ViewLoader and then explored in the interactive mode. Figure 6-11 and Figure 6-12 present the TTF-2 accelerator beam line.

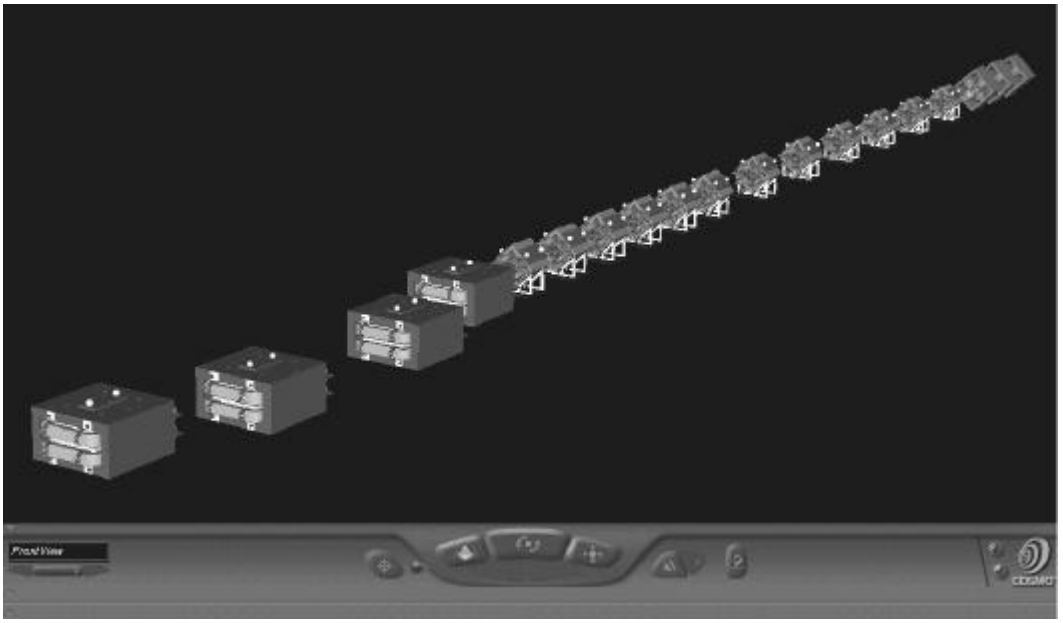


Figure 6-11 The part of TTF-2 in CosmoPlayer.

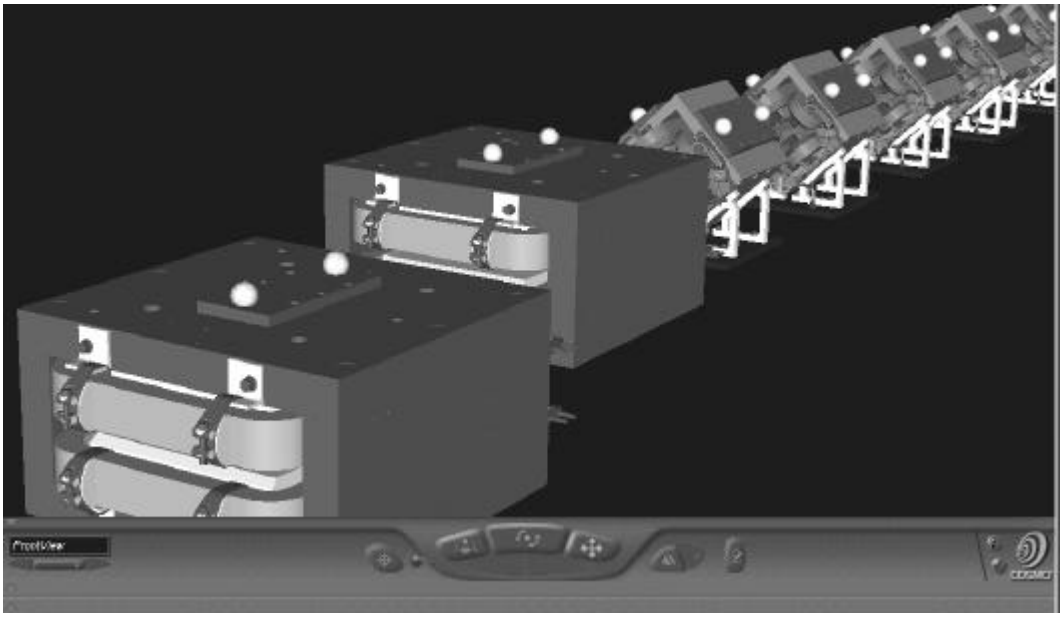


Figure 6-12 A close-up of the previous view.

6.3 General rules for building CAD models.

For efficient and correct work AVC requires specially prepared CAD models. The most important requirement for building CAD models for AVC is the choice of a proper coordinate system. All CAD models should be created using a local coordinate system with its origin in the center of the created model (Figure 6-13).

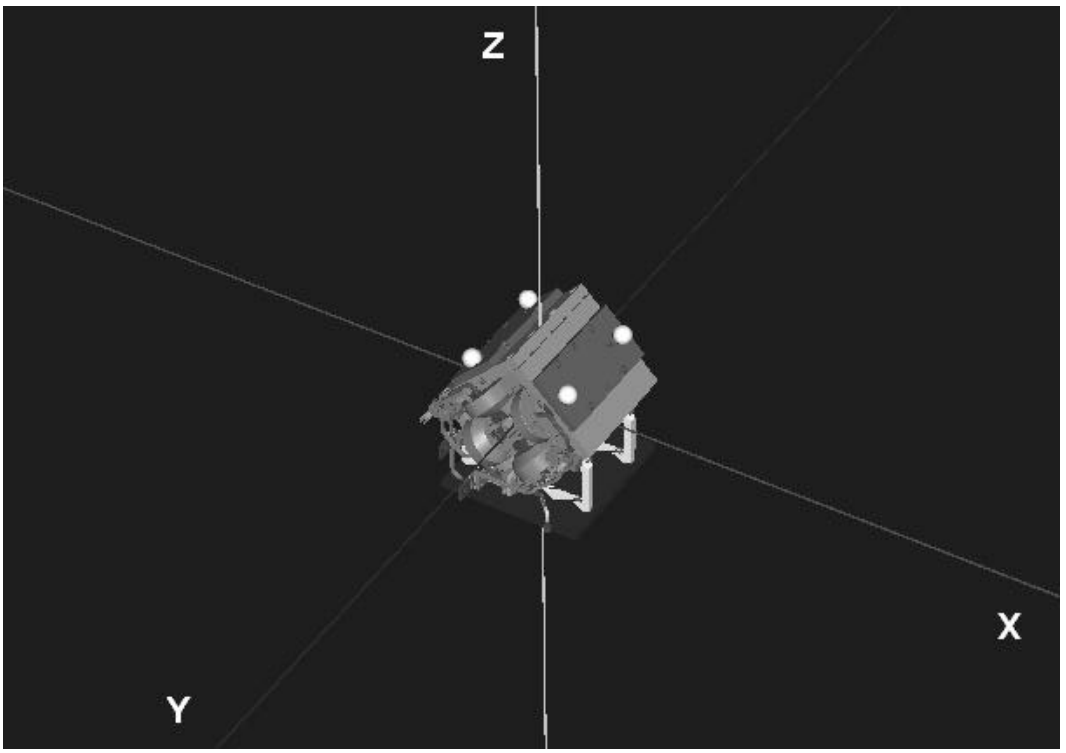


Figure 6-13 The coordinate system with origin the center of the model.

There are two possible approaches to the coordinate system problem. In the first one, CAD models from ADT and I-DEAS could use the same coordinate system shown in Figure 6-14 (and above in Figure 6-13) where the Z axis represents height and the Y axis sets the beam direction.

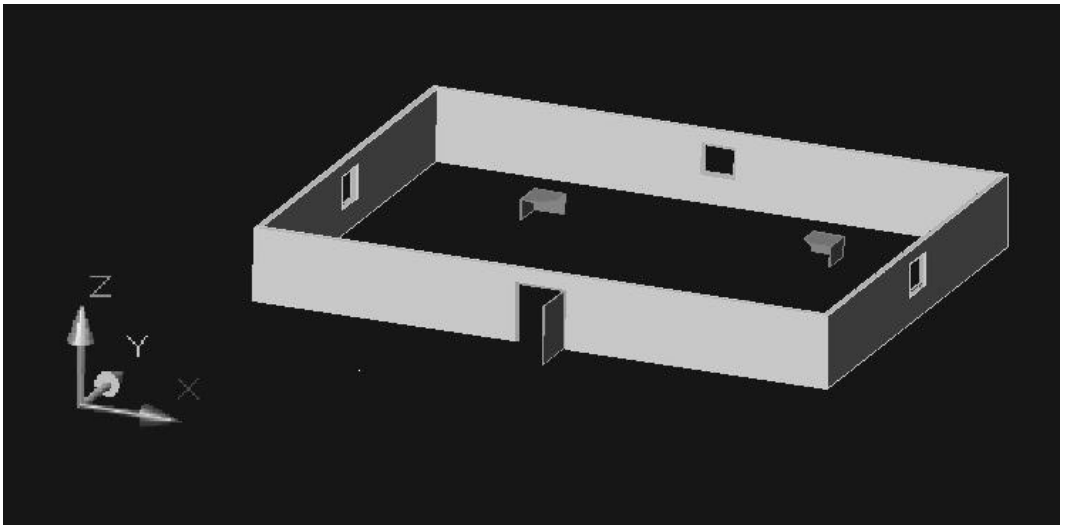


Figure 6-14 Unified coordinate system.

In the second one, I-DEAS could use a coordinate system with switched axes Y and Z (currently used) where the Y axis represents height and the Z axis sets a beam direction. ADT would use the same coordinate system as presented in Figure 6-14 – axes Y and Z would be switched during the export of the ADT models.

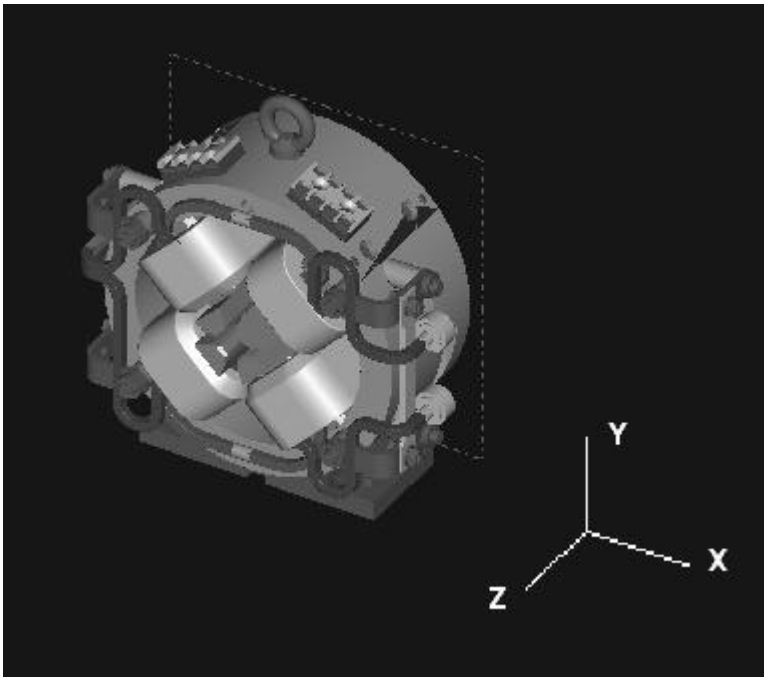


Figure 6-15 Coordinate system currently used in I-DEAS.

The performance of VRML viewers with standard CAD models is not satisfactory, therefore the CAD models used to create virtual worlds for VRML viewers should be simplified. All small elements should be removed from the

model so that only the main shape solids remain. The I-DEAS VRML translator exports spherical geometry as an approximation. Because of that, output files can contain large numbers of vertices. To minimize the size of output files, models should contain as little spherical surfaces as possible.

The CAD models intended for creating virtual worlds for VisView can remain intact – the performance of VisView is satisfactory with the currently used models as long as the number of elements shown in the scene is kept within reasonable limits.

6.4 Experience and lessons learned

During the development of AVC some novel technologies have been used, the most important are VRML, GIS and the CAD systems, I-DEAS and ADT.

VRML has proven to be a good choice for projects involving interference with the internal data of 3D-models:

- it is an ASCII format that allows to easily operate on internal data
- it is an open standard – documentation, tutorials and sample files are commonly available
- it can be viewed in almost any available Web-Browser with an installed free VRML client

Combining 3D data from different sources can be complicated for the reason that each source of 3D data can use its own coordinate system. This creates necessity to transform all 3D data into the one common coordinate system. The task is not necessarily straight forward.

The work on AVC was sometimes not easy because the requirements were very vague and the development process lacked the feedback from end users. This resulted in the frequent changes of the design decisions and lengthened the time of the development process.

At present AVC's applicability is limited as some of the supporting components are not yet completed or available. The GIS system is currently under development and the *picklist* creator has to be developed as a part of this system. A number of necessary CAD models also is not available and has to be created. An algorithm for simplifying CAD models for viewing purposes could be developed as a supplement to AVC.

7 References

[1] R.Kay, “Map and graphics applications can make massive amounts of information readily understandable”, **Government Computer News,** February 1999, online at www.gcn.com/archives/gcn/1999/February22/43.htm 2003.08.22

[2] R.Carter, J.Young, “Data Visualization Made Easy Part I: Preparing Your Data”, Microsoft Corporation, January 1998, online at www.msdn.microsoft.com 2003.08.22

[3] B. Shneiderman **Designing the User Interface: Strategies for Effective Human-Computer Interaction** Addison Wesley 1997

[4] G.Reichbach, “Virtual Reality Offers Potential, Raises Ethical Questions”, **Technology in action**, Vol.5, No. 7, September 1996

[5] M.Good, L.Tan, “VR in Architecture: Today's Use and Tomorrow's Promise”, **Virtual Reality World,** November 1994

[6] A.Rowell, “The design benefit of group VR”, **Computer Graphics World**, Vol.20, No.2, 1997

[7] M.Mineter, B.Gittings, “Parallel GIS algorithms”, The University of Edinburgh 2000, online at www.geo.ed.ac.uk/home/research/gispal/pap1.html 2003.08.22

[8] D.Koller, at al. “Virtual GIS: A real-time 3D Geographic Information System”, In proceedings if Visualization’95 GVU – Technical Report 95-14

[9] F.Rossi, M.Spagnuolo, “Web-based Modeling Techniques Providing Interactive Views of Geographical Data with VRML”, Istituto per la Matematica Applicata Consiglio Nazionale delle Ricerche 1999

[10] R.Carey, G.Bell, C.Marrin, “Virtual Reality Modeling Language (VRML97)” ISO/IEC 14772-1:1997, online at www.vrml.org/Specifications/VRML97 2003.08.22

[11] G.Booch, J.Rumbaugh, I.Jacobson **The Unified Modeling Language User Guide** Addison Wesley 1999

Internet resources

All resources were available at 2003.08.22

[12] www.desy.de/pr-info/desyhome/html/presse/

[13] www.openchannelfoundation.org/discipline/Visualization_and_Virtual Reality/

[14] www.whatis.com

[15] Jon Grover, “Principles in Multimedia technology “, online at www.kiva.net/~jgrover/hci/HCIReport4N501.html

[16] www.vr.clemson.edu/vr/arch/arch.html

[17] www.hni.uni-paderborn.de/vr/nbp/indexe.php3

[18] www.erg.usgs.gov

[19] James G. Natoli, “The Wave of the Future for Information Analysis” online at www.nysgis.state.ny.us/news/gis_art.htm

[20] www.desy.de/dtg/printing/gel3d.gif

[21] www.cai.com/cosmo

[22] www.eds.com

[23] www.cad.luth.se/help/swdocuments/IdeasStudentGuide/SG01chap.pdf

[24] www.web3d.org

[25] www.eds.com

[26] www.parallelgraphics.com/products/cortona

A. Glossary

3D object representation:
is a 3D representation of a single object like a magnet; each 3D object is stored in a separate file;

3D model (virtual world):
is a set of 3D objects combined inside a single file by ViewLoader;

3D viewer: is a viewer for displaying VRML or JT files;

API (Application Program Interface) The interface (calling conventions) by which an application program accesses operating system and other services. An API is defined at source code level and provides a level of abstraction between the application and the kernel (or other privileged utilities) to ensure the portability of the code.

ACAD Architectural CAD; See CAD definition.

AVC: is an acronym for Automated Virtual Reality Creator;

CAD (Computer-aided design) software is used by architects, engineers, drafters, artists, and others to create precision drawings or technical illustrations. CAD software can be used to create two-dimensional (2-D) drawings or three-dimensional (3-D) models.

Class The prototype for an object in an object-oriented language; analogous to a derived type in a procedural language. A class may also be considered to be a set of objects which share a common structure and behavior. The structure of a class is determined by the class variables which represent the state of an object of that class and the behavior is given by a set of methods associated with the class. Classes are related in a class hierarchy. One class may be a specialization (a "subclass") of another (one of its "supercalsses") or it may be composed of other classes or it may use other classes in a client-server relationship. A class may be an abstract class or a concrete class.

Class diagram Diagram that shows the existence of classes and their relationships in the logical design of a system. A class

	diagram may represent all or part of the class structure of the system
elementlist:	is an ASCII text file which contains information about the positions of reference points in the local coordinate system of each object; it contains information about all 3D object representations in the system;
GIS	(Geographic Information System) A computer system for capturing, storing, checking, integrating, manipulating, analyzing and displaying data related to positions on the Earth's surface. Typically, a GIS is used for handling maps of one kind or another. These might be represented as several different layers where each layer holds data about a particular kind of feature (e.g. roads). Each feature is linked to a position on the graphical image of a map.
JT	Also known as DirectModel; JT is a mature lightweight data format for CAD applications.
MCAD	Mechanical CAD; See CAD definition.
picklist:	is an ASCII text file which contains information about the positions of reference points that are inside the “selection box”; reference points are described in the absolute coordinate system;
reference points:	are 3D points placed on 3D objects; they are used to define the position of a 3D object in space;
Scenarios	A use case realization is a graphic sequence of events, also referred to as a scenario or an instance of a use case. These realizations or scenarios are depicted in either a sequence or collaboration diagram.
selection box:	a rectangular region of a GIS map, selected by the user in order to receive a 3D view of the selected area;
UML	(Unified Modeling Language) A non-proprietary, third generation modeling language. The Unified Modeling Language is an open method used to specify, visualize, construct and document the artifacts of an object-oriented software-intensive system under development. The UML represents a compilation of "best engineering practices" which have proven successful in modeling large, complex systems. UML succeeds the concepts of Booch, OMT and OOSE by fusing them into a single, common and widely

	usable modeling language. UML aims to be a standard modeling language which can model concurrent and distributed systems.
VR	Virtual reality is the simulation of a real or imagined environment that can be experienced visually in the three dimensions of width, height, and depth and that may additionally provide an interactive experience visually in full real-time motion with sound and possibly with tactile and other forms of feedback.
VRML	(Virtual Reality Modeling Language) is a language for describing three-dimensional (3-D) image sequences and possible user interactions to go with them. Using VRML, you can build a sequence of visual images into Web settings with which a user can interact by viewing, moving, rotating, and otherwise interacting with an apparently 3-D scene.
VRML Client	Also known as VRML Browser and VRML Player; is a plug-in to Web-Browser that allows to display VRML files and interact with virtual world.

B. Acknowledgments

The author would like to acknowledge the possibility to create this thesis at DESY and use the provided resources, in particular the engineering CAD designs including the model of TTF-2. Special thanks should be given to Prof. Józef Ober and to Dr. Lars Hagge for supervising this work and for their useful and helpful comments and suggestions.