Vegowelt: A smart virtual playroom

Katy Börner
Indiana University, SLIS
10th Street & Jordan Avenue
Bloomington, IN 47405, USA
katy@indiana.edu

Abstract:

This paper describes VegoWelt a smart virtual environment that supports human design activities. The research combines Artificial Intelligence (AI) techniques and Virtual Reality (VR) technology. In particular the approach of *Conceptual Analogy* (Börner, 1997) is applied to provide generative and adaptive design support for human problem solving in an assistant-like fashion.

Introduction

By *smart* virtual environments we mean virtual worlds that *adapt* (or get *customized*, *personalized*) to the behavior of a particular user. Adaptive interaction entails the personalization of *content* and *presentation* of it (Langley, 1997). Content refers to the knowledge structures and inferences used to support human problem solving. The presentation of content refers to the implemented human-computer interface.

Artificial Intelligence (AI) techniques play an increasing role in the development of adaptive design systems (Börner, 1998). However, one of the major problems is the limited interaction capability of the used human-computer interfaces. Often, human-computer interaction (HCI) is restricted to a non intuitive use of keyboard, mouse, and screen. Extensive training is required to handle programs such as CAD tools etc. effectively. Virtual Reality (VR) computer interfaces such as the *Responsive Workbench* (Krüger and Fröhlich, 1994) or the *Automatic Virtual Environment (CAVE)* system (Cruz-Neira et al., 1993) support many of the perceptual channels of information that a person processes. They permit extensive tracing of human behavior that goes far beyond recording mouse events. For example, the position and the orientation of shutter glasses and the user's hand(s) as well as additional button states can be recorded. The resulting behavioral protocols can serve as a basis to support human problem solving in complex tasks involving spatially organized information. Additionally, VR interfaces allow multiple users to interact in a shared virtual and physical space.

There exists a number of VR systems like *Sculptor* (Kurmann, 1995) or *Sketch* (Zeleznik et al., 1996) that allow for intuitive, direct virtual interaction within a 3D world. However, these systems provide only rudimentary support on navigating and manipulating virtual worlds.
On the other hand, there are examples of generative and adaptive AI systems that support human users. *Clavier* (Hennessy and Hinkle, 1992) which supports the generation of autoclave loads was one of the very first systems. But most of these systems still use WIMP GUI's for human-computer interaction.

This paper proposes a system which uses a VR interface for human-computer interaction and which dynamically adapts its knowledge structures and reasoning as well as the presentation of support to its user. Next, we introduce *VegoWelt* (Virtual Lego World) a children's playroom environment that supports simple design tasks and formalize this design task. Subsequently, the approach of *Conceptual Analogy (CA)* is introduced that solves the task in a *knowledge lean, efficient* way. An example section applies the approach within *VegoWelt* to support design tasks by suggesting preferred subassemblies. The paper concludes with a discussion.

VegoWelt

VegoWelt is a smart virtual environment that resembles a children's playroom like setting as depicted in Figure 2. There is a table in the middle of the room allowing him/her to place virtual objects (uniform geometric solids). An unlimited amount of these objects can be picked up from the shelf on the right hand side by direct manipulation. Objects can be moved through so called color cubes in order to paint them in different colors. Simple snapping mechanisms by which objects are positioned in a three dimensional grid are employed to ease the adjustment of objects without requiring force feedback. In order to ease the process of mapping topologies onto each other during retrieval and solution generation, the first object of a new assembly has to be placed in the middle of the table (marked by a cross). Simple sounds are used to denote picking and dropping of objects as well as forbidden object manipulations such as placing an object inside another one. Buttons on the table allow the user to write out design solutions, query the AI system, and exit the application.

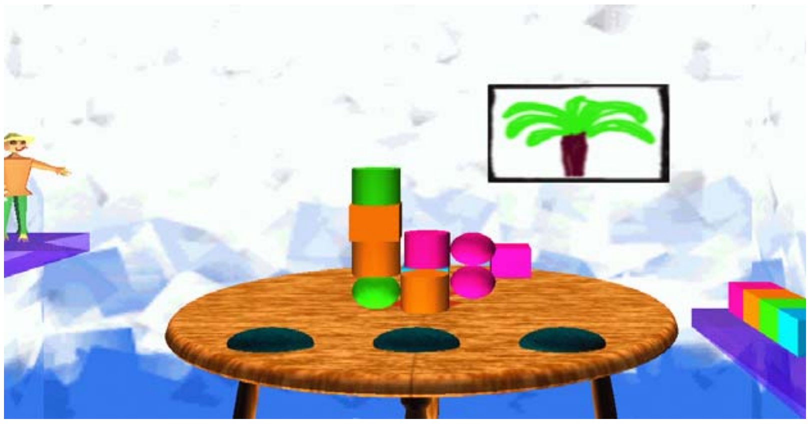

Figure 2: VegoWelt - A children's playroom that supports design tasks

The user's general task is to build a particular object-assembly, for example, to assemble an arch, pyramid, gallow, palm tree, ladder as shown in Figure 3 within a restricted time. The variety of different object-assemblies that can be generated is enormous. Different users select and assemble objects in particular sequences. Final object-assemblies can be saved as good examples. They are represented by the positions of their objects. In order to support the design of different assemblies, not the positions of single objects but the topological relations between objects are important. Therefore, a topological representation is derived which characterizes the set of objects plus the spatial relations among them.

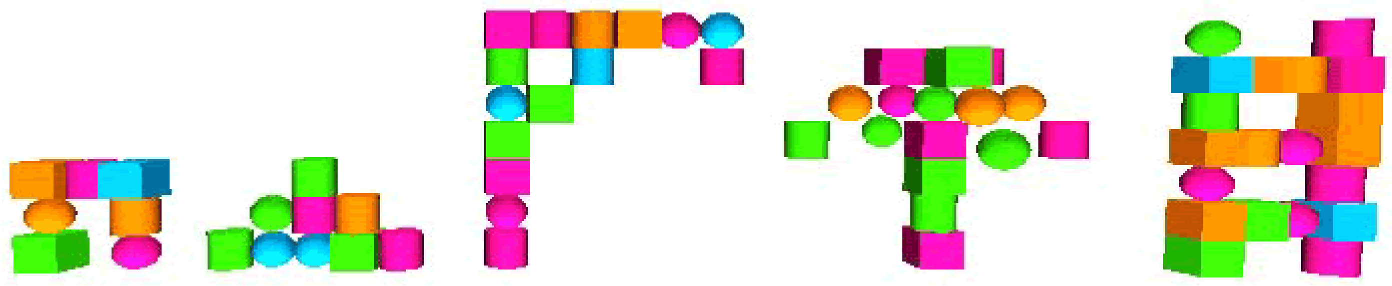

Figure 3: Five simple designs and their graph representations

VegoWelt exploits the CAVE at Indiana University's VRVE Lab to improve the match between the computer interface and the sensorimotoric capabilities of human users. In the CAVE, the user wears a set of shutter glasses that allow for time-multiplexing different images to both eyes. The glasses have a large angle of view, approximately 90 degrees per eye, and are synchronized by an infrared signal from emitters located near the scene. Using a viewer-centered perspective, each eye's image is computed for that eye's exact location. Other participants see stereo, but from the tracked person's perspective. Input devices include a tracked wand with three buttons as well as tracked shutter glasses. User and computer share this world and can manipulate it.

A sketch of the interconnection of the VR interface and the AI system is given in Figure 4 and will be detailed later. The user
designs object-assemblies in the CAVE. Whenever s/he finishes a design, the object-assembly can be stored as a case. Stored cases are used to generate **`Design Solution(s)`** for **`Design Problems`**.

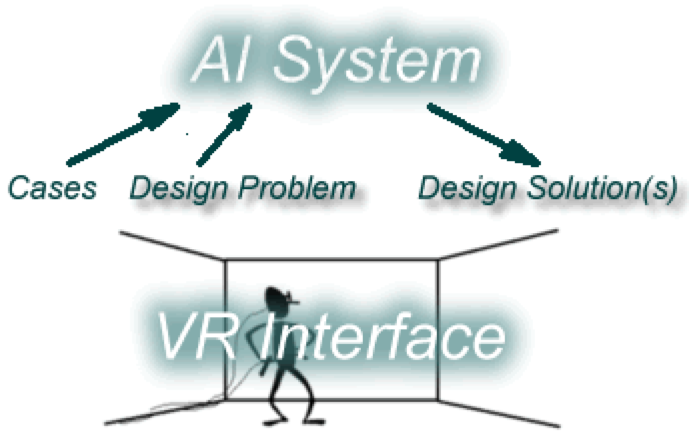

Figure 4: Integrating AI system and *VR interface*

When the system first interacts with a user, it has no knowledge (cases) about his/her preferences. During each session, new object-assemblies are acquired, and if they are similar to previously acquired assemblies, they are merged together to form more general ones. Later on these general assemblies are used to support reasoning in an analogous fashion.

Personalized presentation of content is achieved at increasing levels of generalization via the VR interface. In the beginning, human-computer interaction for novices proceeds at a very concrete level. For example, objects that are typically used are highlighted. During the systems usage, it collects data about user preferences. Based on this it can depict preferred object-assemblies. Instead of manipulating single objects the user can now select larger *object-assemblies*, thus delegating the concrete execution of the commands to the system. In such a way, the user's design activities are constrained by, but not restricted to, past interactions. The interfaces' expressiveness scales along with the users' skill. Personalized knowledge structures are grounded on concrete human-computer interactions and the concrete interaction changes with the knowledge structures that are built up.

Formalizing the Task

Given a large number of object-assemblies (also called cases) a *case-based approach* may be applied. A new design problem is solved by selecting similar past cases, transferring, and if needed, adapting their solution(s) to generate an appropriate solution. For detailed surveys of case-based reasoning see (Aamodt and Plaza, 1994; Kolodner, 1993).
However, to support this kind of design tasks, not the position or type of single objects but their spatial relations are important, and have to be considered during retrieval, adaptation, and evaluation. The structure of cases has to be represented by graphs. A structural similarity measure working over case sets has to be defined that guides the generation of high quality solutions.

For a formal definition of the task we are going to use the following graph-theoretic definitions:

A *graph* $g = (V^g, E^g)$ is an ordered pair of vertices V^g and edges E^g with $E^g \subseteq V^g \times V^g$. A *set of graphs* is denoted by *G*. The *entire graph* $g^{(G)}$ of a set *G* of graphs equals the union of the vertices/edges of the graphs in *G*, i.e., $g^{(G)} = (\cup_{i=1}^{|G|} V^{g_i}, \cup_{i=1}^{|G|} E^{g_i}) = (V^{(G)}, E^{(G)})$.

Based on this we can define a case, case-base as well as a problem and its solution:
A structurally represented *case* $c=(V^c,E^c)$ is a graph. A *case base CB* is a finite set of cases. A *problem* provides a set of vertices and perhaps some edges; i.e., it is a forest. A *solution* of a problem contains the problem vertices and edges and adds those vertices and edges from structurally similar cases that are required to connect all problem vertices and edges.
In the given domain and task hardly any information about the relevance of features guiding the selection of similar cases is available. The adaptation of prior object-assemblies mainly corresponds to adding, eliminating, or substituting objects and their relations. Because of the variety and the possible number of combinations of these modifications, adaptation knowledge is difficult to acquire by hand. Often, several cases have to be combined to generate solutions.

On the assumption that solutions which share a large structure that occurred often in the cases of a case class lead to solutions of higher *quality*, edges of high *relative frequency* are preferred to generate problem solutions. The *relative frequency* P_G of an edge (v_i,v_j) of the entire graph of a set *G* of graphs equals the cardinality of a set of graphs containing this edge divided by the number of graphs in *G*:

$$P_G((v_i, v_j)) := \frac{|\{g \in G \mid (v_i, v_j) \in E^g\}|}{|G|}.$$

The *relative frequency* P_G of a set of edges *E* relative to a set of graphs *G* equals:

$$P_G(E) := \frac{1}{|E|} \sum_{(v_i,v_j)\in E} P_G((v_i, v_j)).$$

Based on this we can define the *quality* μ of a solution. It equals the relative frequency of its edges with regard to the case class *CC* used to generate it:

$$\mu(CC, s) := \frac{\sum_{i=1}^{|CC|} |E^s \cap E_i^{(CC)}| * \frac{i}{|CC|}}{|E^s|} = P_{CC}(E^s) \in [0,1].$$

Aiming at combinations of cases that share structure, structural similarity is defined via the relative frequency of case edges. The *structural similarity* σ maps a set *G* of graphs into the interval [0,1]:

$$\sigma(G) := \frac{\sum_{i=1}^{|G|} |E_i^{(G)}| * \frac{i}{|G|}}{|E^{(G)}|} \in [0,1],$$

were $E^{(G)}_i$, i=1,..., $|G|$ is defined

$$E_i^{(G)} = \{(v_l, v_k) \mid (v_l, v_k) \in E^{(G)} \ \wedge \ P_G((v_l, v_k)) = \frac{i}{|G|}\}$$

as .

Note that the structural similarity function is commutative and associative. Thus it may be applied to a pair of cases as well as to a set of cases.

A *case class CC* is a non-empty subset of *CB* that groups cases of high structural similarity. Assuming that *structurally similar* cases meet all constraints of a task type, a new problem is solved (completed) by transferring the vertices and edges of the most similar *case class*. That way, constraints on designs are transferred implicitly. The solutions are correct with regard to the cases combined.

Taken together, solution generation resembles case combination, where the spatial structure inherent in the cases as well as its relative frequency needs to be considered. Complex case representations, however, require increased computational expense to retrieve, match, and adapt cases. To guarantee answer times that are practical for real world applications, efficient memory organizations directly tailored to the applied reasoning mechanisms are essential. Problems that relate to the amount and the structural complexity of the knowledge have to be addressed.

The Approach of Conceptual Analogy

Conceptual analogy (CA) is a general approach that relies on conceptual clustering to facilitate the efficient selection and combination of complex cases in analogous situations (Börner, 1997). Originally, the approach was developed to support the design of complex installation infrastructures for industrial buildings. Here, cases correspond to pipe systems that connect a given set of outlets to the main access. The approach has been fully implemented in SYN, a module within a highly interactive, adaptive design assistant system that interacts with users via the manipulation of CAD layouts describing pipe systems for fresh and return air, electrical circuits, computer networks, phone cables, etc. of real buildings. See (Börner, 1997) for a detailed description of the implementation. *Conceptual analogy* divides the overall task into *memory organization* and *analogical reasoning*. Both subtasks process graph representations but are grounded on attribute-value representation of past cases. They are explained in detail subsequently.

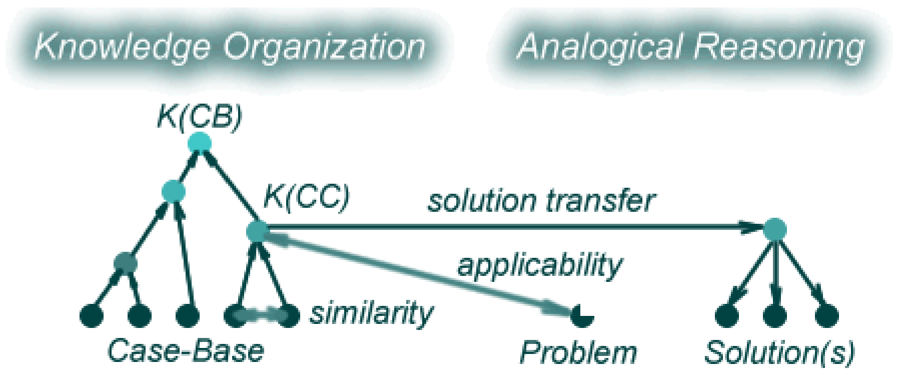

Figure 5: Conceptual Analogy - Overview

Memory Organization starts with a case base $CB = \{c_1, .., c_N\}$ providing a significant number of structurally represented cases as well as a structural similarity function σ. Nearest-neighbor-based, agglomerative, unsupervised conceptual clustering is applied to create a hierarchy of case classes grouping cases of similar structure. Conceptual clustering starts with a set of singleton vertices representing *case classes*, each containing a single case c_i, i=1, ..., |*CB*|. The two most similar case classes CC_k and CC_l over the entire set are merged to form a new case class $CC = CC_k \cup CC_l$ that covers both. This process is repeated for each of the remaining *N*-1 case classes, where *N*=|*CB*|. Merging of case classes continues until a single, all-inclusive cluster remains. At termination, a uniform, binary *hierarchy of case classes* is left.

In order to make the selection of applicable concepts as well as the evaluation of generated solutions more efficient, a concept description *K*(*CC*) is assigned to each case class *CC* in the case class hierarchy. Result is a so called *concept hierarchy*. The concept of a case class *CC* equals *n*=|*CC*| (possibly empty) graphs showing the same relative frequency of their edges relative to the cases in *CC*. Formally, a *concept K*(*CC*) is defined as

$$K(CC) := \{m_i^{(CC)} \mid i = 1, ..., |CC|\}$$

where the vertices and edges of the graphs $m_i^{(CC)}=(V_i^{(CC)},E_i^{(CC)})$ equal

$$E_i^{(CC)} = \{(v_l, v_k) \mid (v_l, v_k) \in E^{(CC)} \ \wedge \ P_{CC}((v_l, v_k)) = \frac{i}{|CC|}\},$$

and

$$V_i^{(CC)} = \{v \mid \exists (v_l, v_k) \in E_i^{(CC)} \ (v = v_l \vee v = v_k)\}.$$

In such a way, large numbers of cases with many details can be reduced to a number of hierarchically organized concepts. The concrete cases, however, are stored to enable the dynamic reorganization and update of concepts.
--> **Example**

Analogical Reasoning is based on concepts exclusively. Given a new problem, the most *applicable* concept containing all problem objects is determined by searching the concept hierarchy in a top-down fashion. If the entire graph of a concept does not contain the problem, it cannot complete it and consequently the *applicability* α of this concept $K(CC)$ to solve the problem $p=(V^p,E^p)$ equals -1. Otherwise it equals the similarity of the case class CC:

$$\alpha(K(CC),p) := \begin{cases} -1 & iff\ V^p \not\subseteq V^{(K(CC))} \wedge E^p \not\subseteq E^{(K(CC))} \\ \sigma(CC) & otherwise \end{cases} .$$

If $0 \leq \alpha(K(CC),p) \leq 1$ holds, then $K(CC)$ will allow to generate at least one solution of p. The concept showing the highest α value is called the *most applicable concept.* It shows the highest structural similarity and solves the problem.

--> Note that the most similar concept may be too concrete to allow the generation of a solution, i.e., its entire graph may not contain the problem.
Instead of *adapting* one or more cases to solve the problem, the concept representation $K(CC)$ of a case class CC is used to generate a set of adapted solutions $S_{CC,p}$ for a problem p. In general, there exists more than one applicable concept. The set of all solutions $S_{CB,p}$ of a CB for a problem p equals the union of solution sets $S_{CC,p}$.

Finally, the set of solutions may be ordered corresponding to their quality. The quality μ of a solution $s=(V^s,E^s)$ depends on the relative frequency of its edges with regard to the case class used. In order to determine $P_{CC}(E^s)$ every solution edge E^s needs to be mapped with every edge of a case. Given the representation of the applied concept

$$K(CC) = \{(V_1^{(CC)}, E_1^{(CC)}), ..., (V_{|CC-1|}^{(CC)}, E_{|CC-1|}^{(CC)}), (V^{PT(CC)}, E^{PT(CC)})\}$$

the solution quality equals the sum of the relative frequency of solution edges divided by the total number of solution edges:

$$\mu(K(CC), s) := \frac{\sum_{i=1}^{|CC|-1} |E^s \cap E_i^{(CC)}| * \frac{i}{|CC|}}{|E^s|} \in [0,1].$$

In this way the concept representation reduces the number of required mappings considerably. If the solution was accepted by the user, its incorporation into an existing concept changes at least the relative frequency of edges. If the solution was not accepted, the case memory needs to be reorganized to incorporate the solution provided by the user.
--> **Example**

Be aware, that system support is based on the users past actions exclusively. So if s/he always goes for bad designs the system will support this behavior and it will assign a high quality to these solutions.

The concrete interconnection of the VR interface and the AI system is shown in Figure 4, revised. Terms that refer to procedures are shaded. Arrows denote the sequence of steps as well as data flows.

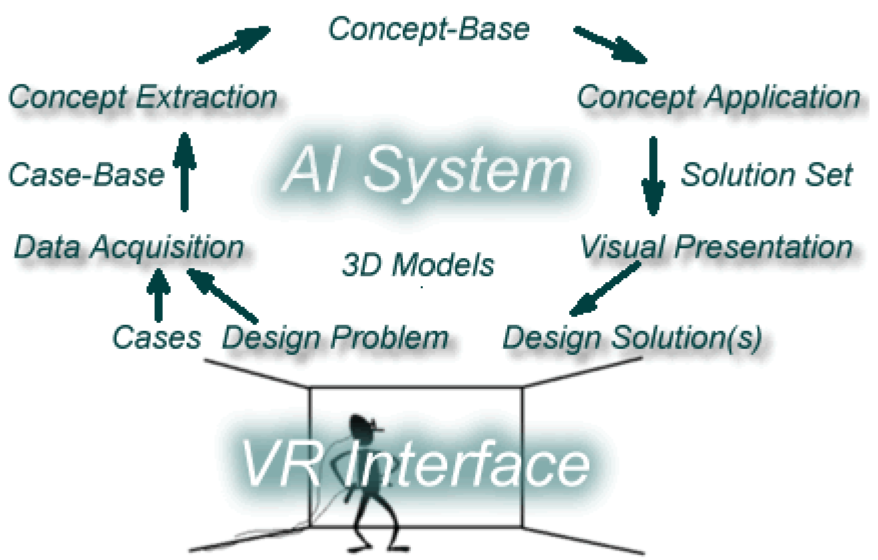

Figure 4, revised: Integrating AI system and *VR interface*

Object-assemblies designed in the CAVE are stored as **`Cases`** in the **`Case-Base`**. Taking into account information about the **`3D Models`** of the virtual world, the cases are represented as graphs. The entire process is called **`Data Acquisition`**. Conceptual clustering is applied to extract a hierarchy of concepts representing knowledge about preferred object-assemblies. The result of this so called **`Concept Extraction`** is a **`Concept-Base`** containing information about the design behavior of a particular user. Support will be provided by applying concepts to generate solution sets, called **`Concept Application`**. Finally, **`Solutions`** are presented as object-assemblies in the CAVE. Again, the **`3D Models`** of the virtual objects have to be considered to compute the correct placement of objects.

Example

Using the approach of conceptual analogy to generate design solutions, the case-base first has to be organized. Given the five cases shown in Figure 3 and asking the user to design a standing lamp, memory organization and analogical reasoning proceed as follows.

Memory Organization: Figure 6 shows the organization of the five cases into a concept hierarchy. N cases are represented by $2N$-1 case classes as well as their respective concepts $K(CC)$. Leaf vertices correspond to concrete cases. Generalized concepts in the concept hierarchy are labeled K_6 to K_9 and are characterized by sets of graphs. Edges with a relative frequency of 1 are shown in black. Edges with lower relative frequencies are drawn in lighter colors.

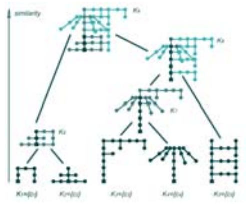

Figure 6: Resulting concept hierarchy
(click figure for larger image)

The structural similarity of these concepts corresponds to the sum of the relative frequencies of edges divided by the number of edges of the entire graph. For example, concept K_6 has two edges with a relative frequency of one, and 10 edges with a relative frequency of one half, divided by twelve edges results in a similarity of (2*1+10*1/2)/12=7/12.
Concepts are represented by sets of vertices and edges. The edges in each set have identical relative frequency. Concept K_7 that represents c_1 and c_2 consists of two graphs with a relative frequency of 1 and 1/2. It can be applied to generate the two cases as well as combinations thereof.

Analogical Reasoning: Typically, in order to design a lamp one would use a case-base that contains designs of lamps. However, let's see which design(s) the system proposes given the concept hierarchy above.

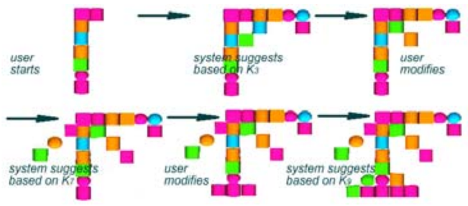
user starts
system suggests based on K3
user modifies
system suggests based on K7
user modifies
system suggests based on K9

Figure 7: Process to design a standing lamp
(click figure for larger image)

The design process is sketched in Figure 7. The user started the design of the standing lamp and sends the partial design (shown in the left upper corner) as a query to the system. The system retrieves C_3 as the most applicable concept and displays it to the user. The user likes the proposed design in general. S/he deletes some objects, adds a reading lamp part to the design and sends it to the system. Using concept C_7, the system proposes a design combination. Being interested in a more stable lamp, the user adds some objects and sends the resulting design back to the system. This time, only the most general concept C_9 is applicable resulting in the design as shown in the lower right corner of Figure 7.

Discussion

Taken together the research applies the approach of *Conceptual Analogy* to build virtual environments that support design tasks. The approach itself can be very well used with todays WIMP GUI's. In fact, its first implementation used a CAD interface to support architectural design tasks. However, design tasks in three dimensions are hard to solve if two dimensional input and output devices are used. VR technology allows to use 3D input and output devices and thus to merge visual, acoustic, and haptic space. The combination of VR interfaces with *Conceptual Analogy* enables generative, adaptive human-computer interaction that feels more intuitive and may provide easy-to-use support for novices and high effectiveness for experts. The toy domain *VegoWelt* resembles real world tasks like the design of houses or cars out of preconfigured parts in that users have to pick appropriate objects in a certain sequence and place them in correct topological relations to one another.

Current work deals with different ways of representing design support visually and acoustically at increasing levels of generalization. Another direction of research concerns the formation and usage of user groups. Given that a new user aims to solve a certain design tasks, s/he could be categorized into an established user group based on the initial interaction chain. The knowledge stored about this user group could then be used to support his/her behavior subsequently. In such a way, users can jointly contribute to and exploit the systems design knowledge.

Acknowledgements

The author acknowledges Indiana University's VRVE Lab for the resources and people that have influenced this work. I would like to thank Matthew Jadud, Sven Bertel, Bob Najlis (right to left) and Mitja Hmeljak (front) for their contributions to the development of *VegoWelt*.

This research was supported by the German Ministry for Research and Technology (BMBF) within the joint project FABEL under contract no. 413-4001-01IW104. Research on intelligent HCI was partially supported by the German Academic Exchange Center (DAAD).
This article was partially processed using LaTeX2HTML Version 98.1p1 originally by Nikos Drakos, Computer Based Learning Unit, University of Leeds.

Bibliography

Aamodt, A. and Plaza, E. (1994).
Case-based reasoning: Foundational issues, methodological variations, and system approaches, *AICOM* **7**: 39-59.

Benford, S., Snowdon, D., Greenhalgh, C., Ingram, R. and Knox, I. (1997).
VR-VIBE: A Virtual Environment for Co-operative Information Retrieval, *Computer Graphics Forum (Proceedings of Eurographics'95)* **14**(3): 349-360.
Börner, K. (1997).
Konzeptbildende Analogie: Integration von Conceptual Clustering und Analogem Schließen zur effizienten Unterstützung von Entwurfsaufgaben, DISKI 177, Universität Kaiserslautern.
Börner, K. (1998).
CBR for design, *in Case-Based Reasoning Technology: From Foundations to Application*, B. Bartsch-Spörl, S. Wess, H.-D. Burkhard, and M. Lenz (eds), chapter 8, Springer Verlag, 1998.
Cruz-Neira, C., Sandin, D. J. and DeFanti, T. A. (1993).
Sourround-screen projection-based virtual reality: The design and implementation of the CAVE, *in* J. T. Kajiya (ed.), *Computer Graphics (Proceedings of SIGGRAPH 93)*, Vol. 27, Springer Verlag, pp. 135-142.
Hennessy, D. and Hinkle, D. (1992).
Applying case-based reasoning to autoclave loading, *IEEE Expert***7**(5): 21-26.
Kolodner, J. L. (1993).
Case-based reasoning, Morgan Kaufmann Publishers, San Mateo, CA.
Krüger, W. and Fröhlich, B. (1994).
The Responsive Workbench, *IEEE Computer Graphics and Applications***14**(3): 12-15.
Kurmann, D. (1995).
Sculptor - A tool for intuitive architectural design, *in* M. Tan and R. Teh (eds), *Proceedings of the 6th International Conference on CAAD Futures'95*, Singapore, pp. 323-330.
See also http://caad.arch.ethz.ch/~kurmann/sculptor/.
Langley, P. (1997).
Machine learning for adaptive interfaces, *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, AAAI Press, pp. 763-769.
van Dam, A. (1997).
Post-WIMP user interfaces, *Communications of the ACM* **40**(2): 63-67.
Zeleznik, R. C., Herndon, K. P. and Hughes, J. F. (1996).
Sketch: An interface for sketching 3D scenes, *Computer Graphics (Proceedings of Siggraph'96)*.