A new Approach to Human-Computer Interaction – Synchronous Modelling in Real and Virtual Spaces

Kai Schäfer,
Volker Brauer, Willi Bruns

Forschungszentrum Arbeit und Technik (artec)
University of Bremen
Bibliothekstr. (MZH), 28359 Bremen, Germany
Tel.: +49/421/218-4833, -4206
{schaefer | bruns | vob}@artec.uni-bremen.de

ABSTRACT

Three-dimensional computer-aided modelling of dynamic processes supported by virtual reality techniques like 3D-stereo vision does not reach the usability (ease, concreteness, intuitiveness, directness) we experience in modelling with real physical objects. We propose an interface that aims at coupling two previously separated model worlds – the the real space of physical objects and the virtual space of signs and images. The basic issues of this Real Reality concept are discussed and some applications are presented.

Keywords
real reality, data glove, graspable user interfaces, grasp recognition, modelling and simulation, programming by demonstration

INTRODUCTION

Several projects with industrial partners as well as our own practice in the design of simulation models indicated that physical models play an important role for cognition and communication. They are used as prototypes for new products, design studies, and for the illustration of complex tasks and processes by making use of their medial qualities (Bruns, 1997; Brauer, 1996). Especially in heterogeneously qualified teams physical models allow to work in a very problem oriented way, without the need to concentrate on user interfaces and software functionality as it is the case with purely computer based tools. Nevertheless the advantages of abstract systems are their capabilities of quantitative analysis, modification, and automatic variation of symbolically represented virtual models. Therefore the idea arose to combine both previously separated model worlds the physical and the virtual one, thus preserving all their advantages.

Our concept of human-computer interaction aims at overcoming some often encountered disadvantages of computerbased working: isolation, sensomotorical deprivation and reality loss. Vividness has been an old aim in computer simulation. Since the early days of Computerised Numerical Control (CNC) for turning, milling and robotics the simulation and visualization of these processes have been impressively improved.

In industrial simulation studies, however, where we had to model complex production facilities and logistic systems, we appreciated the possibility to discuss and manipulate the planned system on the basis of mock-ups, physical models made of paper or plastic (Bruns, Heimbucher & Busekros 1995). Fig. 1 illustrates the use of physical models during a planning session for a factory layout.

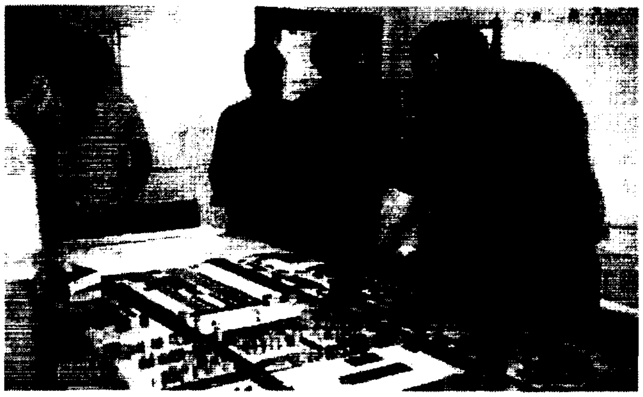

Fig. 1: Factory Layout using Physical Models (Scheel, 1994)

After discussing several variants the results are usually transferred into a virtual model, for example a simulation model, on the computer. This step is the source of several errors: important details may be forgotten, the person who defines the abstract model may use his individual interpretation of the task or the modelling tool addresses not the specific requirements. Furthermore the abstract representation may not be transparent to all participants. From this time on these team members are excluded from the design process.

DIS '97 Amsterdam, The Netherlands
© 1997 ACM 0-89791-863-0/97/0008...$3.50

Obviously both kinds of models, the abstract as well as the physical one, have their own advantages. The question is: Why not coupling them tightly instead of using them sequentially? From this question arose the idea to develop a new interface technology for acting synchronously on a physical and virtual model (Bruns 1993).

THE CONCEPT OF A GRASPABLE REAL REALITY USER INTERFACE

In 1993 Bruns et al laid the foundation for a new class of user interface in shop floor and handicraft working (Bruns 1993, Bruns et al 1993). The main characteristic of the "Real Reality User Interface (RR)", as they called it, is the use of the user's hand as a manipulator of physical objects in a real environment. Appropriate interface devices like data gloves and tracking systems are used to capture the user's hand movements and finger flexions. With the help of gesture recognition algorithms (Brauer, 1994) the raw interface data is analysed and gestures, grasps, or user commands are recognised by the computer in real time. Working with physical objects while being linked to a computer has a certain analogy to the well known Drag & Drop principle of GUIs. When the grasp of an object happens all following data of the Drag-Phase is recorded. This phase terminates when the user places the object at another location and releases it (Drop). Now, the physical object has a new position and due to this the virtual computer internal model of the physical environment is immediately updated. By giving the user an acoustic feedback in the moment of grasping and releasing, the graphical output on a display becomes obsolete. So the user can work distinct from the encumbering aura of the monitor, the keyboard and the mouse.

The term *Real Reality* emphasises the difference to *Virtual Reality*, where the user immerses in and is surrounded by the interface. Real Reality means remaining and experiencing the real world, where all human senses are attracted, and communication within groups is encouraged. The interface becomes a passive observer and is ideally not noticed by its users. This is achieved by linking physical objects to their virtual representation. Because of maintaining objects having both a physical and a virtual representation these objects are called „twin-objects". This makes the description of actions effecting two model representations at the same time easier.

Preparing Real Reality Modelling Sessions

The twin-objects are one of the basic elements of the RR concept. For both kinds of object representations a number of instances must be available. This means to create a virtual representation consisting of the object's geometry and attributes describing the dynamic behaviour. The geometric description contains the object's size (length, width, height) and its surface shape. On the other hand, the physical objects may be constructed using technical construction kits, wood or other materials. The object representations may vary in shape, size and level of detail.

After mounting the data glove, its bending sensors must be calibrated by opening and closing the hand. Then the user places the hand on the tabletop which serves as the model ground, and the zero position is determined by the RR system software. Now the user's hand position is defined relative to the origin of the model's frame of reference.

In the initial state, a model is empty. The objects are located in an object box which has a predefined position on the tabletop (fig. 2 and fig. 3). Thus, for each object in the box the position can be computed.

Using the Real Reality Modelling System

After preparing the modelling elements and the data glove, the RR software is initialised and ready for use. A model is created stepwise by taking twin-objects out of the object box and putting them on the model ground. Because two models are managed synchronously, the RR systems provides two views on the same model. With the help of 3D visualisation software, the virtual representation is displayed on a monitor screen. Although the visual feedback is not necessary for those who model with the physical model, it is used for replaying the actions recorded during a modelling session. Furthermore people working in remote locations can observe a modelling process via a network connection to the Real Reality Model Server.

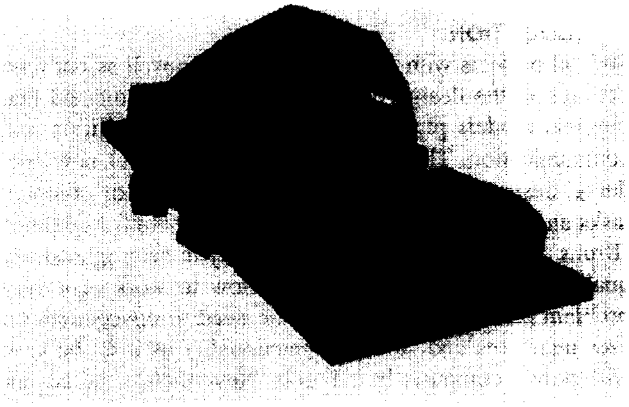

Fig. 2: The Virtual Hand grasping a Twin-Object

Fig. 2 shows the virtual representation of a hand reaching for a twin-object contained in the object box, in this case a conveyor. It is taken out of the box and is placed to a location near another conveyor that was inserted into the model in a preceding step (fig. 3).

A grasp is recognised if a valid grasp-pattern occurs within the close vicinity of an object. If this happens a *grasp event* is generated and the subsequent coordinates measured with the tracking system are recorded. By opening the hand the object is released, a *release event* is generated and the recording of the coordinates is terminated. By this procedure all manipulations, done with the objects are captured and ordered chronologically.

Fig. 3: Building a Model with Twin-Objects

By pointing at a specific object the user can access information about it, while creating a model. The information is displayed on a monitor. For the future, voice output by the computer and projections on the tabletop are planned.

When the model is complete, it is saved in a file. For this purpose a special file format, the Simulation Modelling Language (SML), was developed. It contains the model's geometric description (position and orientation of each object) as well as the recorded actions (paths) performed with them. The SML representation is used subsequently for analysing the model, and also serves as a documentation of the modelling process which can be later recalled and graphically animated.

GENERAL SYSTEM ARCHITECTURE

Working synchronously with two models requires sophisticated communication structures between several software modules in which each of it is responsible for a specific task. These tasks are:

- maintaining a virtual model means to keeping track of the actions performed with the twin-objects,
- recognition of grasp and gesture events,
- data acquisition for recording object movements,
- providing a visualisation of the modelling process and
- persistent storage of data in SML files.

According to this allocation of tasks, a general software architecture was created (fig. 4). A central component of this architecture is the Real Object Modeller (ROM). This module maintains an object database which contains geometric object descriptions, visualises a 3D graphical model, handles SML files, and provides an interface for the dynamic data exchange (DDX) with external processes. Via DDX and appropriate communication protocols, a connection of external processes, running on different machines, may be established. This is the case for the GestureServer task which handles the data glove and sends gesture event messages to the ROM. Another example is the SimulationConverter which acts as a mediator between the ROM and standard simulation software products. Fig. 4 shows bi-directional connections between the DDX interface and simulators. This indicates that a SML model is downloaded from the ROM, translated with a converter to a simulator-specific data format, and is then simulated. The dynamic simulation yields to changes in the model, for example a container is transported with a conveyor to a new location. These changes are immediately transferred via DDX to the ROM where the virtual model is updated. With this link a dynamic visualisation of the experiments done with the simulators is achieved. We also developed visualisation clients which can be connected to the ROM via DDX. By using standard TCP-IP protocols for data exchange even a connection via the Internet to the ROM is possible. This architecture has several advantages: computational power of various computers becomes available, different hardware and operating system platforms may be used, and remote access and visualisation of dynamic changes to the model is provided.

For application software development the ROM may be used in two different modes. Firstly, it can act as a server which manages the basic RR features. In this case an application process which implements the application-specific functionality is connected via DDX to the ROM. The ROM passes all messages to the application where they are handled depending on the application's logic. Secondly, the ROM software may be used as an application framework. Programmers may use the implemented software classes as a basis for their own developments and extend it with additional functions.

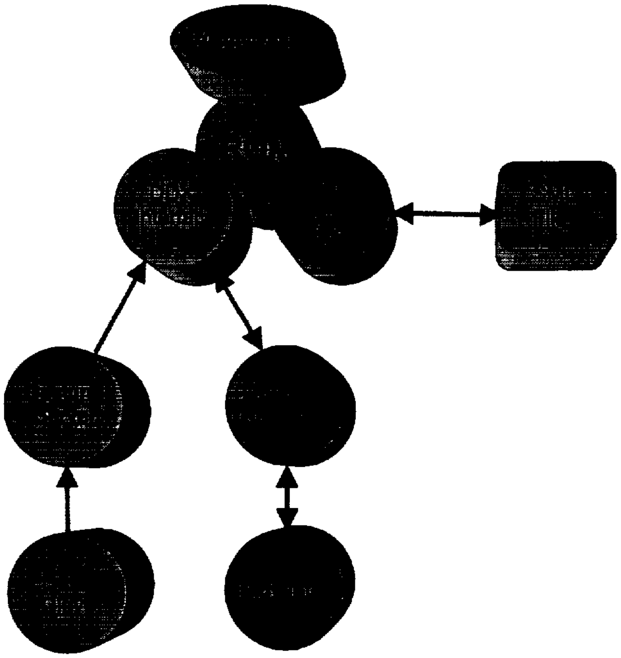

Fig. 4: System Architecture of the Real Reality Modeller

Grasp and Gesture Recognition

In a first approach, the grasp recognition algorithm simply added the bending values of the fingers measured with the data glove. If the sum was greater than a threshold and the hand was located within a close vicinity of a twin-object a grasp was indicated. This solution was quite uncomfortable and inflexible: for releasing an object the fingers had to be splayed out and no hand posture recognition for gesture input was provided.

The actual version is based on statistical multi-variate discriminance analysis (Brauer, 1994). Different grasp- and gesture-patterns can be defined using a teaching software. The user teaches the algorithm with some examples and for each example a set of characteristic features (a feature vector) is computed. While acting on the twin-objects, the gesture server continually tries to match an actual feature vector with one of the taught patterns in the n-dimensional feature space (fig. 5).

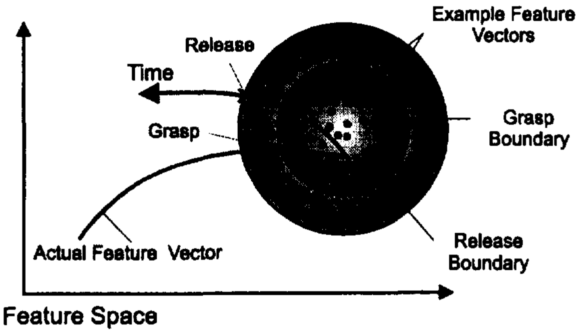

Fig. 5.: Classification of a Feature Vectore

The recognition function returns a value that indicates the distance of a given feature vector from the mean feature vector calculated from the set of examples of a specific gesture. If the distance is smaller than a threshold, a grasp or a gesture is recognised. If this happens, the gesture server subsequently tries to classify only this gesture. If the bending values of the fingers change (the hand opens), the recognition will fail and a release event is indicated. For a stable grasp, the releasing distance threshold may be greater than for recognition.

The statistical recognition algorithm performs well and runs in real time on a dedicated 4/86 CPU.

Problems with Grasp Recognition

The reliability of the grasp recognition depends on the abstract hand model and the measured hand data. The glove we use with the RR Modeller, measures five bending values and thus offers only a simple hand model with five degrees of freedom (DOF) plus six coming from the tracking device. Non-linear glove values yield variances between hand postures that subjectively look equal. An accurate calibration of the data glove helps avoiding problems but a more sophisticated input device would be valuable. Furthermore, the tracking system is extremely irritated by metallic objects. This constrains the choice of materials for models and requires a metal-free environment.

APPLICATION OF REAL REALITY FOR SIMULATION

We expect the Real Reality concept to bring the ease of use and intuitiveness of playing with construction kits to the design and programming of abstract models like models for computer simulation. In our approach, this aim is achieved with a stepwise refinement strategy. By placing twin-objects on the model ground, the model's basic structure is specified. Each type of twin-object (by default) has some attributes characterising its dynamic behaviour. The SML description provided by the Real Reality Modeller can be translated to a first version of a simulation model which is ready for analysis. With the capabilities of the DDX interface and the visualisation features, the model's dynamic behaviour becomes visible through an animation of the objects' movements (fig. 6). This immediate feedback helps the participants of a modelling session to evaluate their decisions. In subsequent steps the model can be refined by specifying rules and by modifying attribute values assigned to the objects.

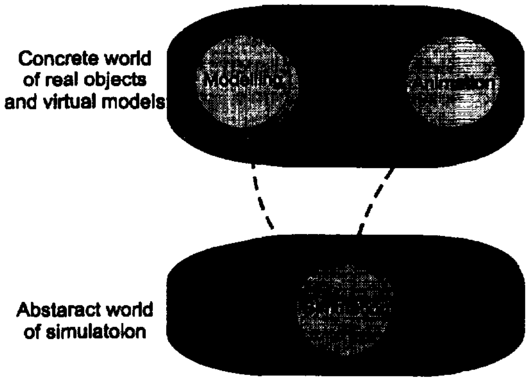
Concrete world of real objects and virtual models
Modelling
Animation
Abstaract world of simulatolon

Fig. 6: Relationships between Modelling, Simulation and Animation

In general, there are two possibilities to make refinements of the simulation model. One of these is to use the functionality provided with the simulation software tool. In this case the user leaves the physical modelling environment and works with the computer in the conventional manner. The other one is to keep on using the RR Modeller and to make use of its features for teaching and specifying additional model behaviour in reality.

Programming by Demonstration

To make the computer redo or even derive programs from what was previously demonstrated by the user, is an old issue in human-computer interaction. This research is currently focussed on the conventional WIMP (Windows Icons, Menus and Pointers) interface styles (Cypher, 1994). The 3D interface provided by the RR concept offers new opportunities for specifying dynamic model behaviour by demonstration. The investigation of this potential is one of our main goals of research. In this section two approaches to this issue are discussed.

Programming Robots and Distribution Rules

In a first approach, a scenario out of the domain of production and logistics was constructed. The conveyor in the foreground delivers containers of different types (here represented by differently coloured blocks) in the plant. A robot has to transport these containers to one of the three conveyors in the background which convey them to further places of processing. The assignment of a specific container to a target conveyor depends on its type (colour). For a concrete situation these assignments have to be specified by the user. Additionally, a control program for the robot which picks up the containers and places them on the target conveyor must be created. In the following it is described how these tasks can be performed with the RR interface.

Fig. 7: Modelling of a Conveyor System

The initial scenario described above was created with the RR Modeller (fig. 7). The containers are located in the object box which in this case is simply drawn on the tabletop. Now, the user takes the containers and moves them through the system on individual paths. While putting a container from one conveyor to another the user plays the role of the robot that picks, transports, and releases containers. Furthermore, the RR application recognises the assignment of a specific type of containers to a target conveyor. According to the RR philosophy the movement paths are recorded, can be saved and animated. A path which bridges a gap between different conveyors (fig 8) may be refined with a path editor, and a basic version of a robot control program is generated.

In addition to control programs, rules for the distribution of the typed containers within the system are generated, for example: *„put green containers always on conveyor A"*. The rules and control programs also can be simulated.

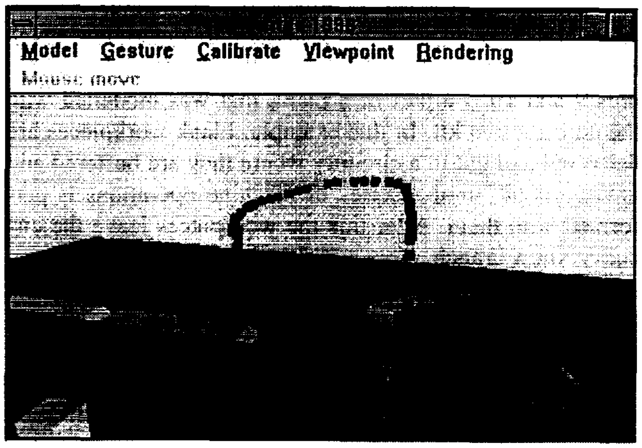

Fig. 8: Visualised Path of an Object Transport

Firstly, randomly created containers are moved through the virtual conveyor system according to the taught set of ramification rules and paths. This way, experimental changes of the material flow through the system can be easily and intuitively analysed.

Secondly, by scaling the model and the paths to the size of a real plant the control programs for the robot can be simulated. For this purpose the robot simulator COSIMIR is employed. It offers various types of robots contained in a library. This simulation tool provides the functions to make visible unreachable coordinates and to optimise transport curves.

Application for Event Based Simulation

Currently we are working on applying the Real Reality concept for event based simulation systems for material flow control and plant layout. A typical scenario in this area of application is, a conveyor system supplying several machines in a factory with raw materials or halffinished parts for further processing. It is the task of a control system to ensure an optimal utilisation of the production capacities. Especially the order of machine utilisation is an important factor for the productivity of such systems. In praxis, this logistical problem is hard to solve, even for small plants. Therefore simulation technology, especially event based simulators are widely used as a planning tools.

For supporting this kind of simulation task a construction kit consisting of conveyors, machines and workpieces was built (fig. 9). The rectangular solids represent the conveyors and the quadric solids are the machines, where specific operations are performed. The arrow on each element indicates the direction of the material flow. If a single element has two exits, what means that there are two possible directions available to continue, this constellation is called a branch. On the other hand, if a single element has two inputs, the material may flow together again. Such elements may cause conflicts if material is delivered from more than one direction at the same time.

At a branch a decision is necessary to determine which direction has to be followed. The branch illustrated in figures 9 and 10 is a part of a plant, that was modelled with the construction kit. In this example, blank workpieces are delivered and put in a circuitry where they are buffered and move around, until a series of machine operations is performed with them. After this the workpieces leave the circuitry via a branch. This simple system allows to investigate important aspects of the flow of workpieces through the plant. The main issue discussed here, is the question how to derive rules for branching decisions from the input recorded with the RR Modeller. Furthermore, these rules must be represented in a formal notation for the utilisation in subsequent simulation experiments as well as for the transformation in control programs.

As described above, due to its default attribute settings, the first version of the simulation model already has a basic dynamic behaviour. For branching situations this is a random decision rule. Of course, a more sophisticated control mechanism is needed, and the animated visualisation will instantly make this obvious.

As discussed previously, one possibility of a conflict solving at a branch is to decide in dependence of the objects' attributes. If, for instance, the user first moves a light-coloured workpiece in one direction and then a dark one to the other (fig. 9) this can be interpreted as: *„move all light containers along this direction and the dark ones along the other direction"*. Again, this rule is transferred to the simulator and the participants can evaluate their model immediately.

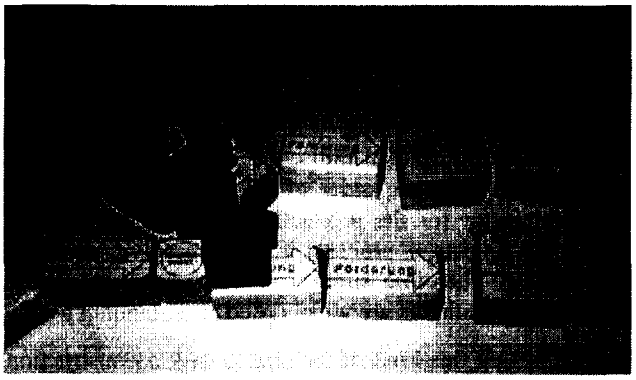

Fig. 9: Demonstrating a Branching Decision depending on Object Properties

In a different situation, the user may want the system to make a decision depending on the current state of the plant. Each resource (machines and conveyors) of the plant is either free or occupied. These two states determine whether a resource is available for processing or not. In the model this is simply indicated by placing workpieces, represented by differently coloured cubes, on the resources. In a branching decision just a few resources are involved. The context of a decision rule must be specified by placing tags on the relevant resources (fig. 10). The situation illustrated in figure 10 shows that the state of the two machines determines the decision of branching. This is indicated by their tags (see the small discs). One of the machines is occupied whereas the other is free for processing. The user moves the light-coloured workpiece towards this machine. From this action the following rule can be derived: *„if machine A is occupied and machine B is free then occupy B"*. From now on, the simulator applies this rule each time the specified condition occurs.

The taught set of rules is represented by a Petri-Net. The resources of the plant are mapped to vertices which are connected via edges, according to the plant's topological structure. The workpieces (or other goods) are mapped to the tokens which are switched from vertex to vertex. By teaching rules edges are inserted in the net what yields a modification of its dynamic behaviour. Petri-Nets can be analysed, to find conflicts or deadlocks. It is easy to convert them to control programs or simulation models.

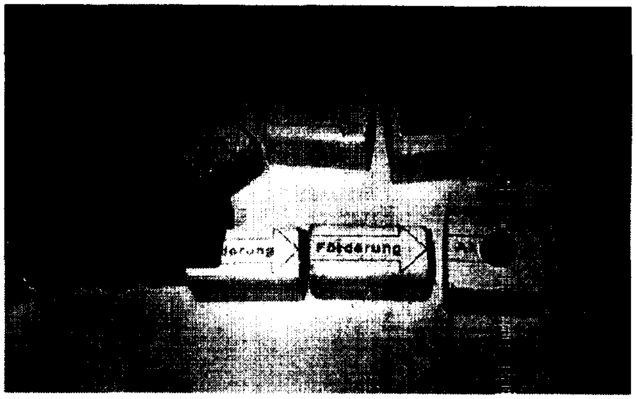

Fig. 10: A Branching Decision depending on the Plant's State

With our approach of programming by demonstration, abstract rules are generalised from manual input, performed with physical objects. The concept is currently under development and will be improved and extended by the integration of gesture commands and two-handed input.

FUTURE RESEARCH AND APPLICATIONS

The application of the Real Reality concept is manifold. An additional example out of the domain of control systems illustrates this.

The construction and operation of control systems often requires the understanding of complex wiring. This is the case, for example, for pneumatic circuits. In reality, physical elements, like tubes and pistons, are linked together to complex systems, while the whole process is supported by the computer. A variety of errors, for example mixed up input and output, are recognised by a control system which compares the actual physical state on the switchboard (fig. 11) with computer internal validity rules.

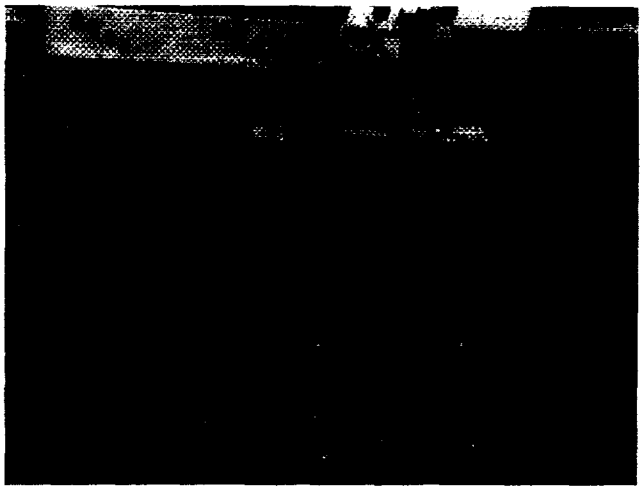

Fig. 11: Switchboard for Modelling Pneumatic Circuits

This application is currently investigated and implemented for utilisation in vocational schools. Using direct manipulations of real objects as a computer interface and integrating this working method into the learning process, as we do, is a new approach. It supports the process of concrete modelling and constitutes a basis for cognitive abstractions, thinking and formation of concepts[1]. This concept raises questions of cognition and system theory. How do we grasp tools and parts? In which way are mental models effected by the acts of grasping and concrete manipulation?

MacKenzie & Iberall (1994) analyze the task-oriented grasping of objects with the hand, and follow the question of how our brain controls our hands. Their research aims at a deeper understanding of the relation between functions of the central nervous system and the grasping hand. They identify different phases of prehension: 1. planning an opposition space (perceive task specific object properties, select a grasp strategy, plan a location and orientation of the hand), 2. setting up an opposition space (preshape fingers, orient palm, drive fingers guarded), 3. using an opposition space (capture, lift, manipulate, replace), 4. releasing an opposition space (release stable grasp, open hand into rest position or open posture, transport hand away from object). They discuss a variety of constraints which become effective on different levels in these phases: social/cultural, motivational, functional, physical, neural, perceptual, anatomical/physiological, evolutionary/developmental. Their results are helpful for a systematic modeling of natural and artificial hands and prehension. The complementary question, however, is: *„How do our hands influence our concepts?"*

Gentner & Stevens (1983) turn to a question that is relevant for our research: which kind of formal representation of physical phenomena is useful for a stepwise differentiable system of conceptual and mental models usable from novice to expert stage of learning? They emphasise requirements like: manipulation of uncompleted, unstable, fuzzy, 'unscientific' models. This motivates us to aim at the integration of concepts of 'Naive Physics' and Qualitative Process Theory (Kuipers 1994, Iwasaki et al 1994) into our work.

Furthermore, there is a whole catalogue of problems to be investigated from the perspective of Real Reality in the future:

- CAD freeform modelling with deformable shapes,
- robot and CNC programming by demonstration,
- traffic simulation and control,
- planning processes for abstract resources like time, tasks, energy, money etc.

Apart from theoretical questions and the extension of the RR concept to new areas of application, the concept itself can be improved in various aspects:

- The problem of pattern recognition for grasping and gesturing. It seems to be promising not to use general pattern recognition methods but to adopt specialties of the grasping hand as they have been investigated by MacKenzie & Iberall (1994).
- The problem of object manipulation and transportation. We need improved abstract models of the hand for the extrapolation of the hand data measured with sensors to effects in the physical world. More sophisticated input devices such as touch sensors or video observation are required.
- The problem of abstraction from recorded activities and events to generalised rules and behaviours.
- The interactive refinement of the model. The interface discussed here, mainly provides input via the twin-objects and the data glove. The output of the RR Modeller still uses conventional display devices. This interrupts the team-oriented design process where the participants stand around the modelling table. This problem can be solved by making use of projectors or large display devices.
- Validation and Verification of the virtual model. The physical model elements such as conveyors, robots and other devices can be equipped with motors and sensors thus offering a fully functional representation of a real system. By employing such a model for testing the taught rules and control programs, a validation and verification of the virtual model is possible. Building functional models will be one of our next tasks.

RELATED WORK

Several efforts are being undertaken to improve the concreteness of modelling. One main stream is Virtual Reality

[1] in german this is expressed by the terms greifen (grasping) and begreifen (grasping / understanding)

(VR) where the aim is to get a better view and feeling of virtual objects by using interface devices like data gloves and head mounted displays (Barfield & Furness, 1995). Improving the appearance of the virtual is not our aim. Another approach frequently referenced to is Augmented Reality (AR) where the user's view of his physical environment is merged with computer-generated images (Feiner et al., 1993; Milgram et al., 1995). In contrast to the approach proposed in this paper the main emphasis of AR lies on enhancing reality with information stored in the computer but not on creating new models.

Although the idea of combining physical entities and virtual objects is investigated by several researchers, their aims as well as their areas of application differ from ours crucially.

Murakami & Nakajima (1994) propose a new interface for direct and intuitive 3D geometric shape-manipulation. They developed a sensored physical elastic object which can be deformed with bare hands. The object's deformations are applied to a virtual 3D shape model displayed on a computer screen.

Fitzmaurice et al. (1995) propose a Graspable User Interface. They use tracking sensors as physical handles (bricks) for controlling virtual objects. The bricks are located on a flat screen and are logically linked to their visible virtual counterparts, thus moving a brick yields to a translation of the attached object. They propose a new kind of drawing program with the option of using several bricks simultaneously. At MIT's Media Lab this approach will be improved in the Tangible Media project (Ishii, 1997). Instead of flat 2D models they additionally use stereo vision and 3D geometrical models, displayed on a desk-like device. The movements of physical handles such as cubes or pyramids are mapped to their graphical counterparts. By doing so, graphical user interfaces are enhanced with physical embodiments of their elements.

Farther reaching is the concept of ubiquitous computing (Weiser 1991). Computational functionality is embedded in many physical artefacts and spread throughout our environment. A behaviour construction kit (Resnick 1993) allows building models, consisting of computerised LEGO pieces with electronic sensors. These can be programmed using LEGO/Logo and are then spread throughout the environment, where they can interact with users or other physical objects.

An open tool for situative learning is proposed by Suzuki and Kato (1995). They developed a graspable programming language, called AlgoBlock. The language consists of several commands having a physical representation. The physical blocks are equipped with an electronic interface. This way the computer can keep track of which blocks are connected. By plugging block by block a sequence of commands can be created and the result is graphically displayed on a monitor. This arrangement supports learning of abstract issues in groups by manipulating concrete artefacts.

All these approaches have a sensorisation of physical objects in common. The implementation must provide a model of how to react on the changes, sensed by the objects. This is different from our approach where the user's hands are sensored, and a model of how the hand changes the environment is implemented.

Kang & Ickeuchi (1994) propose a concept of programming robots by concrete teaching. From the recording of hand postures and gestures they generate a program for automated assembly. The technological basis of this concept is similar to ours, but it provides no perspective for supporting the modelling process in general, nor does it define a new user interface for human-computer interaction.

IMPLEMENTATION

The RR Modelling system described here, is based on PC hardware and the graphical user front-end makes use of Microsoft Windows NT 4.0 functionality. For programming of the RR modules C++ language is used. The rendering and visualisation of 3D graphic models is implemented with Sense8's World Tool Kit R6 programming library. An additional library we found valuable for numeric calculations, analysis and data management is the Library of Efficient Datatypes and Algorithms (LEDA) provided by the Max-Planck Institute for Computer Science, Saarbrücken. For the event oriented simulation tasks we use SIMPLE++ from AESOP which has been extended for data communication via the DDX interface. Robot simulation is done with COSIMIR from Festo Didactic. The table lists the hardware equipment.

Hardware	
Computer	Pentium 200-PC
Graphics Board	Diamond FireGL 3000 (OpenGL)
Data Glove	5th Glove, 5th Dimension
Tracking System	Polhemus IsoTrack II

With this configuration we score frame rates between 10 and 32/sec, depending on model size, complexity and rendering quality. Running the simulation on the same computer reduces the frame rate down to 3/sec, whereas distributing the tasks on different computers always provides a satisfying performance.

CONCLUSIONS

In this paper we introduced a new concept for human—computer interaction. In Real Reality Modelling physical models and virtual models are tightly coupled through sensored user hands. The experience with prototypical applications shows some major advantages of this concept:

- The similarity between real and virtual objects supports the spatial and dynamic orientation in complex systems. Physical laws are carefully respected (spatial extension of bodies, steadiness of motion, friction, acceleration, synchronisation).
- The physical model can be viewed from different perspectives, without additional technical means like head mounted devices. The context as a whole is always preserved.
- The user senses the hardness and heaviness of the twin-objects and uses them intuitively.

The power of this concept compared with conventional Graphical User Interfaces lies in its orientation towards all human senses during the modelling process, especially to the haptic. Instead of sensoring each object, the concept of utilising the hand, yields a universality, because we can use all familiar objects of our surrounding as the user interface. The independence from conventional user interface devices supports a cooperative and communicative work in teams.

ACKNOWLEDGMENTS

We would like to thank our colleagues for their important contributions: Achim Heimbucher for the constructive view of an art designer, Dieter Müller for the first prototype developments and his reflections on model theory, and Martin Faust for the implementation of the prototypes presented above.

Our research is supported by the Deutsche Forschungsgemeinschaft (DFG) (G-Nr. Br 1556/2-2).

REFERENCES

Barfield, W. & Furness, T. (1995): Virtual Environments and Advanced Interface Design. Oxford University Press, New York

Brauer, V. (1994): Feature-basierte Erkennung dynamischer Gesten mit einem Datenhandschuh. Diplomarbeit, Universität Bremen

Brauer, V., Bruns, W. (1996): Bridging the Gap between Real and Virtual Modeling - A new Approach to Human-Computer-Interaction -. Proc. of IFIP 5.10 Workshop on Virtual Prototyping, May 4-6 1996, Arlington Tx, USA.

Brauer, V. (1996): Simulation Model Design in Physical Environments. ACM SIGGRAPH, Computer Graphics, Vol. 30, No. 4, November 1996, pp. 55-56.

Bruns, F. W. (1993): Zur Rückgewinnung von Sinnlichkeit - Eine neue Form des Umgangs mit Rechnern. Technische Rundschau Heft 29/30, pp. 14-18

Bruns, F. W., Busekros, L., Heimbucher, A. (1995): Joint Project: Simulating Work and Technology - Development, Implementation and Application of Computer-Based Systems for Planing, Simulation and Animation in Manufacturing. In F. Rauner (Ed.): Qualification for Computer-Integrated Manufacturing. Berlin: Springer, pp. 188-193

Bruns, F. W., Heimbucher, A., Müller, D. (1993): Ansätze einer erfahrungsorientierten Gestaltung von Rechnersystemen für die Produktion. artec-Paper 21, Bremen

Cypher, E. (Eds.) (1994): Watch What I Do - Programming by Demonstration. MIT Press, Cambridge, Massachusetts

Gentner, D. & Stevens, A. (1983): Mental Models. Lawrence Erlbaum Ass., Hillsdale, New Jersey

Feiner, MacIntyre, Haupt & Solomon (1993): Windows on the World: 2D Windows for 3D Augmented Reality. Proc. ACM Symp. on User Interface Software and Technology (UIST), Atlanta GA, Oct. 1993, pp. 145-155

Fitzmaurice, G. W., Ishii, H., Buxton, W. (1995): Bricks: Laying the Foundations for Graspable User Interfaces. CHI'95 Mosaic of Creativity, pp. 442-449

Gorbet, M. G., Orth, M. (1997): Triangles: A Physical/Digital Construction Kit. Tangible Media Group, MIT Media Laboratory

Ishii, H, Ullmer, B. (1997): Tangible Bits: Toward Seamless Interfaces between People, Bits and Atoms. CHI'97, Atlanta, Georgia

Kang, S. B., Ikeuchi, K. (1994): Grasp Recognition and Manipulative Motion Characterization from Human Hand Motion Sequences. Proc. of IEEE Int. Conf. on Robotics and Automation, San Diego, Cal., Vol 2, pp. 1759-1764

MacKenzie, C., Iberall, T. (1994): The Grasping Hand. Elsevier Science North Holland, Amsterdam

Milgram, Drascic, Grodski, Restogi, Zhai & Zhou (1995): Merging Real and Virtual Worlds. Proc. of IMAGINA '95, Monte Carlo, Feb. 1995

Murakami, T., Nakajima, N. (1994): Direct and Intuitive Input Device for 3-D Shape Deformation. Human Factors in Computer Systems. CHI '94, Boston, pp. 465-470

Piaget, J. (1959): La formation du symbole chez l'enfant. Imitation, jeu et rêve - Image et representation. Neuchâtel

Resnick, M. (1993): Behavior Construction Kits. Communications of the ACM. 36 (7), pp. 64-71

Scheel, Hacker & Henning (1994): Fabrikorganisation neu beGreifen. Verlag TÜV Rheinland, Köln

Suzuki, H. & Kato H. (1995): Interaction-Level Support for Collaborative Learning: AlgoBlock - An Open Programming Language. Proc. of the Computer Supported Collaborative Learning (CSCL) Conf., University of Indiana, 1995

Wellner, P., Mackay, W., Gold, R. (1993): Computer-Augmented Environments: Back to the Real World. Communications of the ACM, 36 (7), pp. 24ff