CAS as Environments for Implementing Mathematical Microworlds

Burkhard Alpers
Fachhochschule Aalen – University of Applied Sciences
Department of Mechanical Engineering
balper@fh-aalen.de

Abstract
In this article we want to investigate whether Computer Algebra Systems (CAS) are suitable environments for implementing so-called mathematical microworlds. For this, we recall first what constitutes such a microworld, and then investigate how CAS can be used for implementation, stating potentials as well as limitations. In order to back our hypotheses, we provide as an example the microworld "Formula 1", implemented in Maple™.

1. INTRODUCTION

The term microworld for a specific learning environment was introduced by Seymour Papert in 1980 and the underlying pedagogical concept has attracted educational researchers ever since. Several Logo based microworlds have been implemented and tested with pupils. Edwards gives an overview of learning environments which carry the label "microworld" and tries to extract common features und pedagogical concepts (Edwards, 1998). We briefly describe the essential aspects of microworlds in an example-free context in the next section, in order to provide the background for the specifics of the paper. This paper focuses on using Computer Algebra Systems (CAS) as an implementation environment for microworlds, rather than discussing the pedagogical potential of such programs at length.

Mathematical microworlds can be completely self-contained or they can carry application meaning and even have an interface to the outer world. The latter have been the subject of recent investigation by (Roschelle/Kaput/Stroup, 2000), and there are close connections to questions concerning the role of representations in mathematical education. This aspect is discussed in more detail in section 3 since in our later example we try to maintain such a link to reality.

CAS have often been considered as "expressive medium" for mathematics since they offer mathematical concepts and a language to express them. (Kent, 2000) already stated that CAS can be seen as medium to "talk mathematics" and he used a CAS (Mathematica™) to implement a microworld called ODEworld. Based on his work, in section 4 we investigate more systematically what the potential and the limitations for using CAS for such a purpose are. In particular, we compare different CAS usage scenarios to position microworlds within the CAS "usage space".

In order to illustrate our hypotheses concerning the usability of CAS, we introduce a small microworld called "Formula 1" which was implemented in Maple™ within one week. Section 5 contains an outline of the objects and operations available in this microworld as well as some anticipated learning opportunities. A more comprehensive description of this microworld including practical experience with grade 12 pupils using it for modelling a toy-course and motion functions for this course will be given in (Alpers/Gerny, 2002).

2. ASPECTS OF MATHEMATICAL MICROWORLDS

The notion of a "microworld" and its usage for different learning environments has been thoroughly investigated by (Edwards, 1998). She distinguishes between two kinds of definitions, a "structural" and a "functional" one, which capture different aspects of this

terminus. The structural definition is concerned with design features of learning environment termed "microworlds" and contains the following aspects (Edwards 1998, p.66):

- a set of "computational objects … created to reflect the structure of mathematical or scientific entities within some subdomain of mathematics or science …".
- a linkage between "more than one representation of the underlying mathematical or scientific entities or objects" (symbolic, visual, sound, motion, …).
- a possibility to construct new objects from existing ones (i.e. the potential for synthetic, constructive activities)
- a set of activities, "in which the user is challenged to use the entities and operations to reach a goal, solve a problem, duplicate a situation or pattern, and so forth."

The functional part of the definition deals with the question how learners are expected to interact with microworlds (Edwards 1998, p.67). The anticipated forms of interaction comprise:

- Manipulation of objects and application of operations "with the purpose of inducing or discovering their properties, and constructing an understanding of the system as a whole. Experimentation, hypothesis generation and testing, and open-ended exploration are encouraged."
- Interpretation of feedback in order to check hypotheses and self-correct wrong assumptions or understandings. Feedback is usually provided in the different representations of mathematical entities (e.g. symbolical, graphical or motion feedback).
- "Use of objects and operations … to create new entities or to solve specific problems or challenges", i.e. perform the activities described in the last item of the feature list above which have been anticipated by the designer of the microworld.

There are several aspects in this definition which induced academic discussion and research. We just outline those ones which are also important when creating a microworld based on computer algebra or in particular for creating the "Formula 1" microworld described below.

Representations

The computational objects contained in a microworld and their relationships represent in some form a mathematical concept or set of concepts. We do not discuss here epistemological questions as to whether the representations "embody" these concepts in an objectivistic sense or are socially-constructed and agreed upon in the community of mathematicians (see (Edwards, 1998) for a treatment of this topic).

The representations available in the microworld are "external" representations in the sense that they are viewable and manipulable computational objects. There usually are multiple representations for one concept in different forms (symbolic, graphical, motion, …). The goal of the learning process is to have the learners construct "internal" representations in their brains which enable them to work with the mathematical concepts represented in the microworld (cf. (Goldin/Shteingold, 2001)). The didactical value of the external representations offered in the microworld, if designed accordingly, can consist of the following aspects:

- Representations in a microworld can bridge the gap between the experience and existing knowledge structures of a learner and a formal symbolic representation of a mathematical concept which is hard to capture directly. (Noss/Hoyles, 1996, p.68) state this as follows: "That is why computational objects are an important intermediary in microworlds, precisely because interaction with them stands a chance of connecting with existing knowledge and simultaneously points beyond it."
- Representations and their manipulation in a microworld can support the learner in performing those manipulations the human brain is not particularly able to do: "…the

aim should be to provide *adequate representations* of objects which the brain is good at manipulating, and *manipulable representations* of objects which the brain is by itself bad at manipulating." (Kent, 1999, p.68) A good example for this are geometric transformations which can be easily done with the computer but are much harder to imagine in the mind.

- Offering "linked multiple representations" (Edwards, 1998, p.61) improves the chance to construct several representations and switch between them according to which is the most appropriate one for solving a problem. (Goldin/Shteingold, 2001, p.4) emphasize the potential of new technology to offer such links dynamically such that changes can be viewed simultaneuosly and hence meanings within one representation can be interpreted within the other.
- Representations can act as "prototypes" (Dörfler, 1993) when reasoning about a certain concept: "A general argument, a conclusion or inference … is carried out by means of the prototypical exemplar which is either imagined or materially present. The prototype guides the cognitive activity by its concretely available properties and relations" (p.178). For an object to be prototypical, the learner must have "the ability to consider the object not just as the given object but as a prototypical, paradigmatic representative of the respective class or concept" (p.179). So, the value depends on whether the learner is able to recognize from the application context the prototypical properties since "… what is specific and what should be disregarded or overlooked is not determined by the object itself" (p.180).

From the above it is clear that when designing objects for a microworld not only the mathematical domain has to be taken into account but also the experience of the group of learners for which the microworld is made. Learner groups with different characteristics (previous knowledge, existing representations and experience) need different representations. We take up this point below when discussing adaptability of a microworld. As (Edwards, 1998, p.73) puts it: "…there is no guarantee that the user will see what the designer intended or what the designer sees…". "External representational systems are useful, or limited in usefulness, according to how individuals understand them." (Goldin/Shteingold, 2001, p.4).

Articulation / Expressiveness / Interface

In order to foster the process of building internal representations a microworld must offer learners a way to work within the microworld, that is to articulate or express what they want to do in the microworld for performing a certain task. The available means for doing this are the objects, operations and feedbacks offered in the microworld. There are several points to be taken into account when designing these offerings concerning content and form:

- (Mellar/Bliss, 1993, p.91) emphasise that the offerings should enable the learner to express his/her own thoughts and therefore must be – in form and content - close to the experience and expressive capabilities of the learner: "The expressive mode of learning describes the opportunity given to students through a tool such as a microworld to express their own ideas about reality through attempting to model it." We already stated a similar point above when discussing representations.
- (Noss/Hoyles, 1996, p.54) point out that - as opposed to a "mere" simulation – a microworld should lead the learner to articulate his modelling and working rather than just play with a model: "…it is in the process of articulation that a learner can create mathematics and simultaneously reveal this act of creation to an observer." The latter aspect is also important for making the learning process more transparent for investigation.
- The means of expressiveness can range from a formal language to a pure menu- and mouse-driven click-and-see interface. There is some discussion on the value of programming in microworlds. (Eisenberg, 1995, p.181) advocates an integration of

"direct manipulation interfaces and interactive programming environments". For the latter he suggests "domain-specific embedded languages" instead of general purpose programming languages. (Mellar/Bliss, 1993, p.109) demand "that a consistent and relatively intuitive meaning can be given to the command, a meaning that will be acceptable to the student". It is certainly a difficult design decision to find a way between the openness and richness of a programming environment on the one hand and complexity and learner distance on the other hand or as (Noss/Hoyles, 1996, p.69) put it: develop "'semi-formalisms', which are general enough to express ideas, but which retain control and familiarity for the learner."

- The discussion on abstraction in (Kent, 1999) and (Tall, 1994) is somewhat similar to the proposition of domain-specific languages which provide a certain view on objects and hide other aspects. Kent recalls the high value which is given to abstraction principles in computer science because abstraction separates concerns and reduces complexity. (Tall,1994, p.194) terms this the "*principle of selective construction*" which enables learners to concentrate on one aspect delegating others to the program. These other aspects then can be successively added enriching the microworld. We will take up this point when discussion extensibility below.

So far, we mainly dealt with the question of expressive means for use by the learner. Vice versa, it is also interesting how the program reacts to learner input and in which way this helps the learner. (Edwards, 1998, p.57/58) considers it as a main feature for a microworld program that it "provides *interpretable feedback* employed by the students in revising their current understanding" or that it offers "information that can be used to discover the intended mathematical relationships." Such feedback can take the form of graphical representations or animations or symbolic output.

Activity design / guidance / exploration

When designing the computational objects and operations in a microworld one has also to anticipate potential activities of the learner within the microworld. According to (Edwards, 1998, p.62), "a curriculum of activities must be developed that provides the students with problems, challenges, or interesting areas to explore." She also requests that the game character of a microworld provides "a meaningful, motivating context for problem solving" (ibid. p.65). (Noss/Hoyles, 1996, pp.67/68) advocate the conctruction of "situations which are sufficiently concrete – as well as interesting and soluble – in the hope that students will generate tasks and sub-tasks for themselves". The underlying idea seems to be that the microworld with its problems and situations acts as a catalyst, starting point and frame for learners' involvement. For this, a realistic background seems to be helpful which we will outline in the next section. Again, there is a fine path between free exploration which might lead to nothing, and guidance which might overly regulate and close off learning paths.

In order to avoid frustration particularly in an environment which includes programming and has a higher degree of complexity, (Eisenberg, 1995, p.182) suggests the inclusion of "scaffolding elements" like "software critics" (warners), "browsable catalogs of exemplary or illustrative work" or tutorial material.

Concerning the kind of activity which should be offered, (Noss/Hoyles, 1996, pp.68) proclaim the "idea of reconstructability": "if we design structures that allow students to (re)construct new objects and relationships out of old ones, we can increase the likelihood that we will achieve some visible and tangible representation of the state of the student's thinking and be more able to observe mathematical thinking-in-change." Here again, the question of making the learning process observable is emphasised.

Another important topic in this area is the design of situations where learners get into difficulties since the microworld does not give the anticipated feedback because of a misconception of the learner. (Noss/Hoyles, 1996, p.65) call such situations "breakdowns"

and stress the learning potential provided by such situations. These "breakdowns" should address learning difficulties which are known from previous investigations of the group of learners for which the microworld is made (Mellar/Bliss, 1993, p.109).

Adaptability / Extensibility

As described above, the external representations offered in a microworld serve to let the learner create internal representations and they should enable the learner to express his/her ideas and not just follow the "logic of the topic" (Mellar/Bliss, 1993, p.109). This calls for a program which is adaptable to the expressive qualifications of a specific learner group if it should not be usable for only one type of learner. In another context, (Noss/Hoyles, 1996, p.68) use the word "macroworld" within which different microworlds or microworld fragments can be created.

Taking not just the starting conditions of a learner group into account but also the development over time, extensibility is an important topic. (Eisenberg, 1995, p.181) points out that "the complexity of such an application should grow with use, as students develop larger and more advanced programs within the application…". This should also allow for differentiation between more capable and less capable students where the former have "the opportunity to develop flexible concept imagery showing the necessity for more subtle symbolic representation of mathematics" (Tall, 1994, p.195). Different levels of understanding should be identified and supported. (Eisenberg, 1995, p.181) has even the vision that with growing complexity the clear cut borderline between educational and professional activity might vanish in favor of a "seamless" transition.

Curricular Embedding

In the description of existing microworlds, the curricular embedding is often not very clear. The central question here is: What are the learners supposed to know to be able to work with the objects and operations presented? Where in the concept development and deepening process is the place of the microworld? (Noss/Hoyles, 1996) state that there was no previous treatment of the topics ratio and proportion before working with "Ratioworld". But it is not clear how in their pre-test pupils are questioned for their understanding of proportional figures when there was no treatment. (Edwards, 1995) states for the TGEO microworld that there was no previous instruction in transformation geometry but two initial classroom sessions on the topic. She also emphasizes in (Edwards, 1998, p.56) that "the learner is not told about a mathematical concept or scientific principle; instead, he or she must discover the concept or principle while solving a problem or playing a game." Working in the microworld serves to "challenge students' current understanding". This indicates that some form of understanding should already be present when working with a microworld but the work deepens, clarifies and enhances the understanding and makes it explicit. (Edwards, 1995, p.84) describes this as "… *use* and thereby learn ...."

3. MATHEMATICAL MICROWORLDS AND REALITY

There are mathematical microworlds which purely work in the realm of mathematics like "Ratioworld" (Noss/Hoyles, 1996), "ODEworld" (Kent, 1999), "TGEO" (Edwards, 1995) or "GreenGlobs" (Dugdale/Kibbey, 1990). Other microworlds like the Boxer microworlds "Number-speed" (diSessa, 1995) or the "MathWorlds" created in the SimCalc project (Roschelle/Kaput/Stroup, 2000) incorporate some connection to reality (application scenarios) as experienced by students. Having a real scenario as a context is also the core of the "Realistic Mathematics Education" approach to didactic of mathematics (Gravemeijer/ Doorman, 1999, p.111): "… context problems can function as anchoring points for reinvention of mathematics by the students themselves." As was pointed out above,

representations serve to bring mathematics closer to the experience of the learner. Therefore, it is quite plausible to place a mathematical microworld within a realistic context which is meaningful for the learners such that the microworld can build the "basis for progressive mathematization" (Gravemeijer/Doorman, 1999, p.117). This must be the ultimate goal since otherwise the microworld would just be a simulation or a learning environment for science but not for mathematics.

Connections to realistic situations can reach from pure interpretation context to interfacing with reality:

- A realistic interpretation of the objects and operations provided in a microworld can simply provide meaning for the learner and thus motivation and a background for posing interesting questions.
- There might also be an animation or simulation of a (naturally rather restricted) real scenario included in the microworld, as is the case in "Graphs and Tracks" (McDermott, 1992). Such animation can provide valuable feedback for testing the correctness or quality of a mathematical object like functions constructed by learners.
- There might also be a possibility to input real data (by some sort of file or online interface or by using the keyboard). This can enhance motivation since the simulated environment is not simply a "sandpit" or a game.
- Finally, there might also be the possibility to control a real scenario with mathematical objects created in the microworld as is the case in "Line Becomes Motion" (Schnepp/Nemirovsky, 2001). The idea is not just to model phenomena but to generate them (Nemirovsky/Kaput/Roschelle, 1998). This provides yet another representation, so there are linked mathematical, simulated and real representations.

The respective didactical values of simulations and interaction with the real world have been investigated in the SimCalc project resulting in the conjecture that "cybernetic and kinesthetic explorations have complementary pedagogical value: cybernetics allow replay and re-examination of more controlled experiments, whereas kinesthetic explorations directly involve bodily understanding and connect directly to familiar experience" (Roschelle/Kaput/Stroup, 2000, p.17).

The realistic scenario which is most often chosen by microworld designers is that of motion. Position, velocity and acceleration versus time scenarios provide a rich context for investigating functions, differentiation and integration and a comprehensive experiential frame for students. (Roschelle/Kaput/Stroup, 2000) term the underlying mathematical concepts the "Mathematics of Change and Variation" (MCV). Motion functions can be considered as prototypes for thinking about functions and their properties (in the sense of Dörfler as stated above), or as (Speiser/Walker 1996, p.370) put it: „Motion … is not just an illustration. It is a paradigm for most of calculus, the part of mathematics that examines change. … The classroom implication is to build on *personally enacted* physical experience."

According to (Schnepp/Nemirovsky, 2001, p.95), using their "Line Becomes Motion" microworld enabled "individual students to take increased advantage of their experiential resources involving motion and thereby to gain a deeper understanding of calculus."

In the motion microworlds mentioned above there is a close link between graphical representations of functions and animations or real motions. This made it possible to teach the MCV concepts to younger children without delving into algebraic notations. Another approach in this direction was adopted in *Boxer* microworlds for motion like "Number-speed" (diSessa, 1995) with its turtle racing activity (used with grade 6 students) where lists of values represent a function. But it still seems to be an open question "… what kind of experiences provided to students in different software environments can provide the best foundations for making connections between symbolic expressions and real world phenomena, and thus to promote students' understanding of symbols as an expressive language?" (Yerushalmy, 2001, p.5).

4. USING CAS FOR BUILDING MATHEMATICAL MICROWORLDS

Creating a mathematical microworld from scratch is a time-consuming activity requiring considerable knowledge in both software design and didactical design areas. Therefore, an implementation environment providing support particularly for mathematical objects and operations facilitates the creation process. The most prominent and often used environments are *LOGO*, *Boxer* (based on LISP, see (diSessa 1986)) and – more recently – *Mathworlds* (based on Java, see (Roschelle/Kaput/Stroup, 2000)). (Noss/Hoyles 1996) call such environments "macroworlds" within which several microworlds can be built.

Although Computer Algebra Systems (CAS) have been used widely in mathematical education and their general didactical potential has been a frequent research topic (e.g. in (Hillel, 1993) and the articles of this journal), their usefulness as implementation environments for microworlds was only recently investigated by Kent (1999, 2000) who also constructed as an example the "Trajectories" microworld based on Mathematica™. In this section, we take up the research done by Kent and look further into the potential as well as the limitations of CAS for creating mathematical microworlds. In doing this, we take into account the advantages of the environments mentioned above in order to check whether CAS can equally well serve the purpose. The statements will be backed in the next section by giving an example for the creation of a microworld in Maple™. Although in the sequel of this section we write about "CAS" in general, it is clear that there are differences between CAS and there is a permanent addition of features. This holds particularly with respect to interface issues. So, the reader should have in mind that the author's CAS background is mainly Maple 6™.

Going through the list of design features of microworlds given by Edwards (see section 2), it is obvious that CAS provide computational objects for many mathematical concepts, including functions, expressions, equations and so on. These objects make mathematical concepts and thoughts explicit since they have to be used for working in the CAS environment. This "work" consists of using operations which are available for the objects offered. (Kent, 1999, p.13) calls this the "object language" and claims that already by making abstract ideas visible and manipulable as computational objects they become more concrete in the sense that one can do something with them (cf. the above quotation of Edwards: "... use and thereby learn ..."). Still, he sees it as an open research question, "… in what ways object languages could offer learners access to formal ways of thinking" (ibid., p.15).

Going further on in Edwards' feature list, it has often been stated that CAS provide multiple linked mathematical representations: symbolic expressions, numeric value tables, static graphics (curves, more general: geometric objects) as well as animations, even sound (e.g., by production of *.avi files). This has been investigated particularly for functions, and we will treat this special case more thoroughly in the next section. There is one limitation compared with other environments worth mentioning. In *Boxer*, for example, there is the possibility of dynamic linking of representations, i.e. changes in one representation have an immediately visible effect in other representations of the same mathematical object (as is also well known from spreadsheet programs). In most of the CAS known to the author this is – at least currently – not possible; re-execution of the respective creation statements for the linked representation is required, e.g. a plot command has to be re-executed in order to update a function graph. Moreover, the operations on graphical objects are usually restricted. Whereas in *Mathworlds* graphs can be constructed by drawing methods (making microworlds built in this environment particularly suitable for younger pupils), paving a road *from* graphical *to* symbolic representations, the operations in CAS are essentially restricted to graph manipulation (scaling, rotation, view of axes etc.). One must add, though, that the environments which provide these additional features are much more restricted with respect to the manipulable objects (e.g. in *Mathworlds* essentially piecewise-linear functions or in *Boxer* tables as function representations). This again resembles the fact that the worlds produced within these environments mainly address the education of younger children.

The third main feature is concerned with constructability of new mathematical objects, or – in a wider sense advocated by (diSessa/Abelson, 1986) – of a new "medium". As to the creation of new mathematical objects from old ones, there are usually many constructive operations already available like the construction of new lists from old ones, the constructive operations on functions (addition, composition etc.), particularly the definition of piecewise-defined functions, the construction of graphical objects from parts (display of separately defined plots) and so on. Again, the construction has to be performed by the learner using the object language instead of using menus, clicks or drag and drop operations with the mouse.

The wider issue of constructability is concerned with allowing the learner to construct his or her own medium giving him/her a sense of ownership and appropriation as advocated by diSessa and Abelson and realised in *Boxer* which they term a "reconstructable computational medium". Since modern CAS are not just "mathematical processing engines" but contain many features of text processing programs and even internet connectability, there is a high potential for personal ownership. Any user can produce his own worksheet (Maple™) or notebook (Mathematica™) with personal descriptions of the activities and goals, constructing nested sections to make the document more readable, adding links to other related sources of information and so on. Therefore, it is quite justified to call a CAS a "reconstructable computational medium".

The essence of the statements made so far can be summarised as follows: CAS provide "for free" a rich set of mathematical objects, operations and construction methods and can be used for exploration and creation of personal documents. But – as we stated in section 2 – the main value of a microworld consists of helping the learner to build internal representations in a certain knowledge domain and for this the microworld should offer him/her adequate means of articulation or expression. So, the question is whether the expressive power of CAS is already adequate or what features CAS provide to make it adequate. This question consist of three sub-questions:

- Is the object language "speakable" by the learner from a syntactic point of view?
- Are the objects suitable for letting the learner articulate his/her thoughts?
- Is the interface adequate?

(Kent, 1999, 2000) made the experience that using a procedure for solving differential equations in Mathematica™ was complicated and rather cryptic to the inexperienced user. Also, the behaviour of some operations like simplification or error messages can be quite arcane. Many CAS are predominantly production tools and not educational tools, so it is no surprise that there are obstacles to the direct usage of CAS for educational purposes. We will examine this point, i.e. the transition from a production to a pedagogical tool, in more detail below.

The second and even more important question is whether CAS as such provide adequate means of expressiveness for a variety of learners. (Crawford, 1995, p.444) already pointed out that the "trade off between generic software and more specialised software … appears centred around the complexity and reconstructability of generic media and the immediate instrumental utility and economy of effort in learning to use well-designed specialised software." CAS provide a lot of opportunities but are as such not adapted to the expressive capabilities of less advanced learners.

For creating adequate representations, CAS provide a programming language which again makes them suitable as an *environment for creating* mathematical microworlds. As (Kent, 1999, p.93) already observed, for adaptation to the learners, an "intermediate language level" for the knowledge domain under consideration (or "domain-specific embedded language" as Eisenberg puts it) must be implemented as is done in computer science when separating concerns of different implementation teams (cf. for example the ISO OSI layer definition for communication in open systems). In his "Trajectories" microworld for dynamics, he programmed Mathematica™ functions where "computationally, details have been suppressed

beneath an abstraction barrier …, but also mathematically, the details of the Trajectory's construction … have been suppressed in favour of a 'qualitative' study of trajectories and phase portraits" (ibid. p.99). Having the symbolic programming facilities available, such functions or procedures can be implemented by a programmer in a relatively short period of time. The "Formula 1" microworld described in the next section was programmed within one week. So, what (diSessa, 1995, p.344) claims for the *Boxer* environment, also holds for CAS environments. This also allows for quick prototyping and testing. We stated above that the objects and operations in the microworld should enable the learner to express his/her ideas and it is often hard to predict and varying between learner groups what these objects might be. Quick prototyping allows early tests and easy adaptation later on. One has to admit, though, that programming with CAS is not a trivial task, so a teacher using a CAS mainly with the available objects and operations will have difficulties to do the adaptations on his/her own. But probably the same holds for changing LISP code in *Boxer*.

When programming the operations, the creator of a microworld should also take into account how to provide "interpretable feedback". This can be done within a CAS from a software point of view by returning expressions, graphs or animations which show whether or not the assumptions of the learner are correct. For example, in the "Formula 1" microworld described below there is a routine that produces a picture of the constructed course such that students can see whether their assumptions on the structure (and closure) of the course are true.

As to the interface issues stated in section 2, there are clear restrictions in many CAS. The requirement of Eisenberg that microworlds should combine "direct manipulation interfaces and interactive programming environments" is often hard to fulfil in a CAS environment since there is no graphical user interface construction toolkit as known e.g. from VisualBasic™ where one can easily build an interface with buttons and boxes. So, CAS are definitely more oriented towards the interactive programming side of the combination and are hence probably more suitable for implementing microworlds for more advanced high school pupils or undergraduates. Another aspect of interfaces is the connection with reality. It is quite easy to import real data into a CAS via a file interface and the same holds for export in order to "generate phenomena". This has been frequently used by the author in CAS projects where geometry data of automobile parts were imported and spline data were exported to a CAD system for further construction and later production (cf. (Alpers, 2001)). It is certainly a disadvantage that such a file interface is not an online interface.

Finally, we want to investigate the aspect of extensibility including questions on the transition from an educational to a professional tool or from *using* tools to *making* tools (Kent, 2000). CAS allow for two ways of extending the work with a microworld: One can inspect more thoroughly the existing objects and operations to see "what is behind the scene", or one can program new ones. With respect to the first way (comparable with inspection of boxes in *Boxer*), it is possible to have a deeper look into the data objects and the procedures. Internal specifications of data objects can be hidden first and made available later when a basic understanding is present. In the "Formula 1" microworld, for example, one can start with course pieces like "short_line_segment" and "curvy_arc" and only later investigate by what data such objects can be determined (see the next section). This is often more problematic with respect to procedures. The "normal" user of a CAS microworld will hardly be capable to make sense of the code implementing a procedure. This seems to be somewhat opposed to what (diSessa, 1995, p.342) claims concerning *Boxer* microworlds which grant the learner "access to inner workings" but those internal structures seem to be much simpler than CAS programming code or they are equally hard to understand (like LISP code for the non-specialist). As (Kent, 2000) also observed, there is quite a step to get from using tools to making (i.e. programming) tools. So, this extension work is rather left to the programmer and only gradually can one expect a more advanced user to get into tool making and thus into the "productive" (as opposed to the educational) usage of a CAS. In order to make this point

clearer it is useful to have a look at different types of using a CAS and possible ways to get from one type to another. Figure 1 depicts a part of the CAS "usage space" as far as using CAS in application scenarios is concerned. The figure contains two axes: the horizontal axis named "CAS model reduction" indicates that going to the right not all basic CAS objects and operations are available but the offering is reduced to only a subset of these and additionally programmed "higher-level" ones. The vertical axis named "Application reduction" indicates that going down means reducing the modelling depth and precision for a real scenario. Within the space spanned by these axes we identified several types of usage. These are traditionally split into industrial or educational types but there is certainly not a clear separation line inbetween (hence the boxes are rather to be interpreted as overlapping clouds).

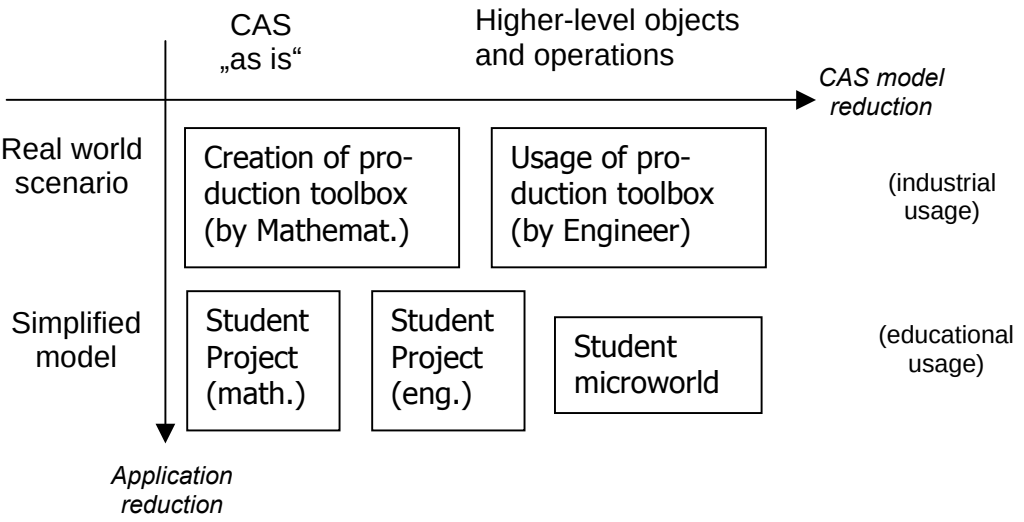

Figure 1: Positioning microworlds in the "CAS usage space"

Starting in the upper left region, using CAS for implementing a production toolbox is mostly the job of a mathematician or a mathematically very talented engineer (see, e.g., the toolboxes or application packages available on the internet pages of CAS vendors, or – as an example – the production of toolbox for cam design according to engineering guidelines in (Alpers, 1998)). The engineer who concentrates on engineering aspects then rather "enters the scene" as a user of such production toolboxes. In the educational realm, there is a similar constellation but the models of reality are often much simpler. Using the CAS "as is" in mathematical application projects is possible in university education as was more detailed described in (Alpers, 2001). For engineering students, we offer additionally some higher-level procedures (e.g. for spline creation or fourier analysis) such that the students can concentrate on other aspects and the projects are not overly time-consuming. Going more to the right, we locate the microworld users who get even more higher-level objects and operations and mostly work within the restricted set given in the world. The advantage with respect to later extension is that microworld learners already work within a production environment which is specifically interesting for very practically oriented engineering students. Transition from the microworld "corner" in this usage space can go in both directions, i.e. going up and working with an extended microworld providing a more realistic and rich application model as well as going left and inspect what is behind the objects and operations provided in the microworld. But still, as the experience of (Kent, 2000) who worked with mathematics students (!)

showed, transitions are not easy. We will provide examples for these transitions in the "Formula 1" microworld described below.

5. EXAMPLE: "FORMULA 1"

In order to back our hypotheses on the usability of CAS for constructing mathematical microworlds, we present as an example the "Formula 1" microworld. As the name suggests, this microworld has a realistic background, namely car racing, be it on a real Formula 1 course or on a toy course which we used with pupils. Even if students are not familiar with car racing, they certainly have experience with cars being driven on a road. So, as was already successfully done in the respective SimCalc (Roschelle/Kaput/Stroup, 2000) and *Boxer* (diSessa, 1995) microworlds, as well as in "Graphs and Tracks" (McDermott, 1992), we can make use of a rich experiential background for interpretation and question-raising with respect to motion. Particularly the SimCalc world served as a model for our microworld in many respects although the kind of students addressed in our world is different.

In order to set up such a car racing situation, two basic constructive activities have to be performed: course construction and motion construction. The real situation is far too complex to be dealt with by students, therefore simplifications along the axes of our "CAS usage space" depicted in Figure 1 are necessary. The reductions we made on the vertical axis include the following:

- The course has no breadth but is simply modelled as a line which can consist of circle segments and line segments (this kind of description can also be found in web pages describing real courses, see for example: http://www.hockenheimring.de/). For toy courses, this is not such a simplification although the slipping of cars is not modelled.
- The motion is only considered as one-dimensional, so we do not consider lateral acceleration. Moreover, one can reduce the complexity by considering simple acceleration schemes (full positive, full negative, 0), by not modelling gears or friction , and so on.

Given these application reductions, one could simply let students use the mathematical objects available in a CAS environment to construct courses and motion functions as was done in a project for mechanical engineering students in the third semester (cf. Alpers, 2001). In order to enable younger students to work in such an environment or to let engineering students get further into the motion modelling phase (not spending most of the time for course construction), it is useful to offer higher-level objects and operations beside those already available in the CAS which are also to be used for modelling car racing like piecewise defined functions. In this respect, the horizontal axis in Figure 1 could also be labelled as "CAS enrichment" but the reduction label serves to indicate that only a small subset of the basic CAS objects is used whereas other ones are hidden in the higher-level objects which are nearer to the situation to be modelled. The higher-level mathematical objects and operations which – together with basic objects and operations like lists, functions, differentiation, integration and plotting – make up the microworld are listed in Table 1.

Before we describe this microworld in more detail, we elaborate on the mathematical concepts which the learner should better understand by working within the microworld. The original intention was to build a microworld for working with functions (as is done in SimCalc) but course construction turned out to offer a rich didactical environment for understanding the geometry of line and circle segments. Since aspects of the situation can be hidden or made visible depending on the intentions of the teacher, it is possible to use the microworld for both knowledge domains.

We will not describe the microworld with its potential learning paths in full detail here since this article is mainly concerned with using CAS as a suitable implementation environment (cf. (Alpers/Gerny, 2002) on the "Formula 1" microworld itself and the experience when using it with grade 12 pupils). Rather, we will concentrate on how the aspects of microworlds described in section 2 could be realized in a CAS environment.

Course definition and construction	**Commentary / Example**
course_piece:=[length, radius, orientation];	c1:=[3,5,`-`]; circle arc of length 3, radius 5, clockwise orientation; s1:=[3,infinity,`+`]; line segment of length 3 (`+` has no meaning here)
course_data:=[piece1,piece2,...,piecen];	course_data:= [[3,infinity,`+`],[3,5, `-`]]; or: course_data:=[s1,c1];
course:=construct_course(course_data);	constructs a course for later plot
complete_coursedata:=complete_course(course_data);	computes pieces for closing course
length:=compute_length(course_data);	computes course length
plot(course);	plots the course
Restrictions	
restriction:= construct_restriction(course_data, max_value_list);	max_value_list:= [10,20,infinity,10,infinity,20];
plot(restriction,s=a..b);	shows v(s)-diagram for restrictions
Motion functions	
Transformation procedures:	
voft2vofs(v) ;	Input and output are functions
voft2soft(v);	dito.
vofs2soft(v);	dito.
soft2vofs(s) ;	dito.
aofs2soft(a);	dito.
Investigation of functions	
lap_time(course_data, function);	computes lap time for certain motion
Animation	
animate_motion(course, function, number_of_frames);	shows motion of a circle on the course according to motion;
animate_motion_with_restriction(course, function, restriction, number_of_frames);	if velocity restrictions are not fulfilled, circle „jumps“ out of course
compare_motions(course, function1, function2, n..m, number_of_frames);	two circles run simultaneously on the course from piece n to m.

Table 1: Additional objects and operations in the „Formula 1“ microworld

In order to offer the learner objects for course construction which are nearer to his/her experience and hence to the expressive means, we defined and implemented objects and operations which appear when building a toy racing course. There are different pieces like long or short line segments and more or less long and “curvy” circle arcs. Depending on the knowledge background of the learner, the details of the mathematical description of such a piece can be hidden or made visible. In the first case, the learner can simply use a pre-defined piece like “curvy_arc” whereas in the latter case learners have to build the mathematical representation of the piece themselves, i.e. set up a list like (in Maple™ notation):

```
> curvy_arc:=[ 2,  3,  `+`];
```

which gives motivation for a discussion on how to describe a piece and its usage when connecting it to an existing part of the course (left or right curve: `+` or `-`). This shows that as in *Boxer* microworlds the contents of data structures can be hidden first and made available

for inspection later in the learning process. One has to admit, though, that in this example the content of the data structure is still rather near to physical experience.
Course descriptions are simply modelled as lists of pieces. The most important operation then is to construct a course from these data such that it can be depicted. The construction procedure sets up a piecewise-defined curve in parameter representation which is rather tedious. Plotting the course like the one depicted in Figure 2 constitutes the visual "feedback" for the learner. Here, he/she can see whether or not the understanding of pieces was correct and the course is closed. Note that for plotting no additional implementation work is necessary. Possible activities like re-building existing courses or thinking up new ones lead to interesting geometric questions like: How can a partial course be closed with few pieces (for giving feedback concerning this question the operation "complete_course" was implemented)? Do the pieces available in a toy course have special measures to ease the completion problem? What kind of courses can be build with a certain set of pieces?
When constructing a course the learner simply uses the respective commands listed in Table 1. Hence, the CAS just serves as an environment in which the commands are implemented and work. More use of CAS variation capabilities for experimentation is made when constructing motion functions as will be demonstrated in a small example at the end of this section.

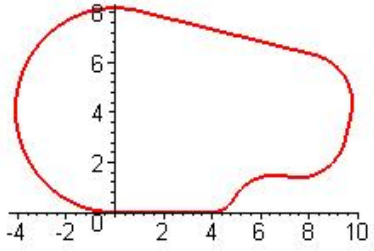

Figure 2: Example for a simple course

The second part of the microworld is concerned with setting up and investigating functions describing the motion of a car on the course. For this purpose, CAS already offer a lot of objects and operations, so additional implementation is only necessary for getting nearer to the expressive means of the learner, for offering operations not yet known to the learner, and for providing feedback. We will describe the additional operations after having outlined the didactical purpose of this part.
There is a vast amount of didactical literature on functions and the use of technology for improving the understanding of functions. We just deal with those aspects that relate to the treatment of and activities with functions in the "Formula 1" microworld:

- The value of using meaningful applications for functions has been emphasised by (Mayes/Schmidt/Swandon, 1996), (O'Callaghan, 1998) and (Roschelle/Kaput/Stroup, 2000), and motion has often been used to provide this application.
- (Vinner/Dreyfus, 1989) have investigated learner's difficulties in understanding the function concept and (Breidenbach et al., 1992) have identified different (more or less advanced) types of understanding, starting with an "action" understanding (sequence of actions for computing the function value for a special input value), proceeding to a "process" image (algorithm for producing output from given input) and arriving at an "object" concept (function as one object with a certain behaviour and which can be operated upon) (cf. also (Kent, 1999) for an outline of the so-called "APOS" theory). Among others, (Hillel et al., 1992), (O'Callaghan, 1998), (Cuoco, 1995), (Breidenbach et al., 1992), (Weigand, 1997), (Gomes-Ferreira, 1997), (Kent, 1999) and (Yerushalmy/ Shternberg, 2001) investigated how higher-levels of understanding can be achieved by using technological environments (in particular CAS in the work of Hillel et al., Cuoco, Weigand, and Kent).

Having available multiple representations of functions (numerical, symbolic, graphical) is considered as potentially beneficial. But the mere availability does not guarantee an actual linking of properties in one representation to properties in another one as (Even, 1993) observed. Linking representations and respective properties is supported when changes in one representation are simultaneously depicted in the other (called "dynamic" linking in (Gomes-Ferreira, 1997) or "term-graph-interactivity" in (Weigand, 1997)). Unfortunately, in most of the CAS currently available, this is not implemented such that an explicit switching by re-executing commands is necessary. Moreover, the "procedure" of linking is mainly uni-directional from term expression to graph. (Roschelle/Kaput/Stroup, 2000) advocate and implement a way to graphically construct a function (mainly piecewise linear functions) which seems to be particularly appropriate for providing learning opportunities for younger children (grade 6 in middle school) and those who have difficulties with symbol manipulation (cf. also (Yerushalmy, 2001) for "paving the road" from graphical to symbolic representations).

- (Roschelle/Kaput/Stroup, 2000, p.10) emphasise the advantages of piecewise-defined functions particularly for modelling motion (distance, velocity or acceleration vs time): "... students spontaneously understand graphical representations of motions (and other phenomena) by performing interval analysis. Piecewise functions draw upon this natural inclination." Another advantage of this type of function observed by (Weigand, 1997) is its support for the "object" concept of a function: Two or more expressions are "bundled" in one object which is named and can be operated upon as just one object. This helps also to address the mis-understanding experienced by (Dreyfus/Vinner, 1989) that students do not recognise a piecewise function as a function.

It is not the intention of the "Formula 1" microworld to introduce the notion of a function or different classes of function. Working with functions in the microworld is meant to deepen the understanding of functions and their properties by constructing functions and exploring their properties motivated by the application background provided by the car racing scenario. As (Weigand, 1997) already stated, experimenting with functions requires a certain amount of function knowledge on the side of the learners since otherwise there is a considerable risk of a mainly "trial-and-error" working style without any gain in understanding. We will state below which aspects and properties of functions are particularly treated in the microworld.

The main activity in the "Formula 1" microworld is the construction of a motion function for a specific course. This would be meaningless if there were no restrictions to be taken into account. The most evident ones present in the experience of any learner are restrictions on the maximum velocity for the car in general and for certain pieces of the course. These restrictions can be visualised by setting up the maximum value list and using the construct_restriction operation which provides a function plot to be used for later restriction checking.

The most important and interesting part is the construction of a motion function. The learner can start, for example, with s(t), v(t) or a(t) (distance, velocity, acceleration vs time). Having the experience of a speedometer in a car, the most obvious approach seems to be to start with a v(t) function (this is also the starting point in other motion microworlds, cf. (Roschelle/Kaput/Stroup, 2000), although starting with a(t) having in mind the acceleration pedal would also be based on experience). Since the restrictions are piecewise, it seems to be quite natural to construct a piecewise function ("speed up", i.e. increase velocity, then slow down before the next curve starts, and so on), and – for a start – piecewise linear functions provide rich modelling possibilities without being overly complex and difficult to construct. Whereas in SimCalc (Roschelle/Kaput/Stroup, 2000) such functions can be constructed graphically, in our CAS environment they have to be set up symbolically. Although the SimCalc approach is definitely easier and adequate for younger pupils (grade 6 in SimCalc), the symbolic approach has also advantages for higher-grade students since they have

explicitly to think about how to produce continuous functions. If one goes further on and works with more sophisticated (continuous) acceleration schemes, the question of differentiability also shows up. Moreover, when other classes of functions are taken into account, investigations on which function of which class provides the "best" increase properties come up. These are just some indications of the kind of activities and questions that show up when working on the construction of functions. It is clear that more advanced constructions with a variety of function classes are only possible when basic knowledge of these classes is available. Then, the microworld serves to deepen the understanding by investigating properties which are meaningful in the realistic application context "Formula 1". Concentrating on properties instead of computing values also strengthens the "object" concept image of a function.

So far, the objects and operations (functions, differentiation etc.) needed for modelling motion are already available in a CAS environment without further work. The operations offered additionally mainly serve to support the learner in investigating whether the constructed function fulfils the restrictions and is "a good" driving function. With respect to restrictions, there is an interesting mismatch (comparable to what other authors call "breakdown" or "impasse") which provides the motivation for a rich discussion on "what depends on or corresponds to what" in functions. The constructed velocity is specified for times (as v(t)) whereas the restrictions are naturally specified for intervals of the course (as v(s)), so in most cases it cannot immediately be deducted from the v(t) function whether restrictions are fulfilled. One can let the students work on this problem if they have the necessary prerequisites, or one can provide operations which help them and let them concentrate on the construction aspects. For the latter purpose, the transformation procedures listed in Table 1 are offered. We just mention one of these, namely "voft2vofs(v)" which transforms a velocity vs time function into a velocity vs distance function (which is possible if you drive forward all the time, another interesting topic for discussion). Having this, learners can create a common plot of the restriction function and the v(s) version of the constructed function and see whether the graph of the latter is below the one of the former. This provides optical feedback both on the fulfilment of velocity restrictions and on intervals with potential for improvement. Another operation which provides animation feedback displays a circle running on the course according to the constructed motion function. The circle "jumps" out of the course when a velocity restriction is violated. This is not as precise as the function plot but has some motivational advantages.

Having constructed an admissible function, there is still the question of quality. We use as a "realistic" criterion the lap time and offer an operation to compute this. Transforming functions and computing properties like lap times emphasises the object character of functions and should therefore also serve to create advanced mental images in the learner.

The following short sample session serves to illustrate the potential usage of the microworld. We assume that the learner first constructs the simple course depicted in Figure 3:

```
> long_line_segment:=[ 200, infinity, `+`]: #define line segment
> half_circle_curve:=[ Pi*30, 30, `+`];     #define circle segment
> course_data:= [long_line_segment, half_circle_curve , long_line_segment,
                 half_circle_curve];        #define course data
> course:=construct_course(course_data):    # construct curve
> plot(course);
```

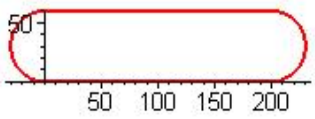

Figure 3

The learner then sets up a list with velocity restrictions per segment and produces a restriction data structure which can be plotted:

```
> max_value_list:=[500, 230, 500, 230]:
> restriction:=construct_restriction(course_data, max_value_list):
```

Next, a velocity vs time function is to be defined. At this stage, the learner can really experiment and use CAS capabilities for flexibly varying functions. We just give an example (assuming that the maximum positive and negative acceleration is 300 resp. –400):

```
> v:=t -> piecewise(t <= 1.1 , 300*t, t <=1.5 , -400*(t-1.1)+330 ,
                    t <=5, -400*(1.5-1.1)+330 ):
> plot(v(t), t=0..5);
```

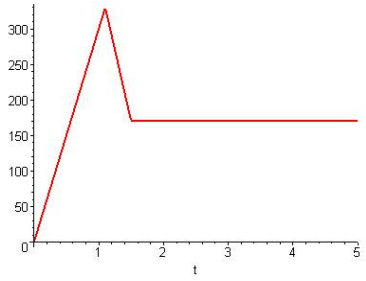

Figure 4

How the CAS capabilities were really used by pupils to construct families of such functions where the time of brake (here: t=1.1) and the end of brake (here: t=1.5) can be varied easily for experimentation purposes will be described in more detail in (Alpers/Gerny, 2002).
In order to check whether the restrictions are fiulfilled, the learner uses the voft2vofs procedure to produce vofs. This can then be plotted together with the restriction function (also depending on s) as is shown in Figure 5.

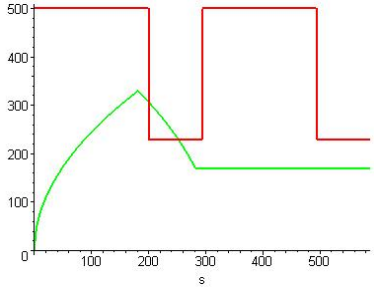

Figure 5

The visual feedback gives ideas on how to change v(t) in order to fulfill the restrictions. If an admissible function has been constructed, the lap time can be computed using the respective command. This serves as quality criterion.

We just mentioned some of the potential activities in the microworld and showed that what is offered can be adapted to the knowledge and capabilities of the learner. One can extend the microworld simply by allowing the usage of other function classes or by modelling other kinds of restrictions (e.g. maximum rate of change of acceleration). The inspection of implemented CAS procedures, though, is certainly not reasonable because of their complexity but the question of "how does it do it?" (e.g. the v(t)->v(s) transformation) can be discussed using simple examples.
As yet, there is no link to a real toy course but this is planned using a file interface (download table of values to a micro-controller which then emphasises the numeric representation of functions!). This should provide more information on the validity of the conjecture made by (Roschelle/Kaput/Stroup, 2000, p.17) on the "complementary pedagogical value" of cybernetic (=animation, B.A.) and kinesthetic explorations quoted in section 3.

6. SUMMARY AND CONCLUSIONS

It was the intention of this article to investigate whether or not CAS are adequate implementation environments for mathematical microworlds. For this, we first recalled the main definition aspects of such learning programs: there are computational objects and operations representing mathematical concepts and there is a set of activities around these objects which serve to understand the concepts. Important aspects of microworlds include the linked representations they offer, the kind of expressive means available to the user comprising also the graphical and programmatic elements of the user interface, the activity design and exploratory potential as well as adaptability and extensibility. Moreover, microworlds need a curricular embedding for enabling meaningful work and not just trial-and-error mouse clicking.
CAS provide already a rich set of mathematical objects and operations and are thus a good candidate for an implementation environment since they offer much without additional programming. On the other hand (although this varies from CAS to CAS), CAS may have a more complicated user interface: Objects can often not be manipulated graphically, but statements in a more or less complicated syntax (at least for a beginner) have to be made. Dynamic linking between different representations (i.e. automated update in other representations when one is changed) is rather an exception in CAS. Some of these problems can at least be alleviated by using the programming facilities in CAS to produce higher-level objects and operations which are easier to handle. This way, it is also possible to provide objects close to modelling a realistic scenario. Simplifying both the application model and the CAS objects means to provide reductions in two dimensions as shown in Figure 1 in order to offer adequate means of expressiveness. The programming facility also allows good adaptation to the expressive needs of the user and also an easy extension. The question of "How is it done" opens the transition path to a deeper understanding but direct inspection of CAS procedures is hardly helpful. We think that CAS are adequate for implementing microworlds that address more advanced learners whereas the microworlds created in the SimCalc projects offered (graphically oriented) opportunities for younger children (grade 6).
As an example, we implemented the "Formula 1" microworld in Maple™ using motion (car racing) as application scenario which has already proven to provide a rich environment in several other microworlds. The implementation was done in one week which backs the hypothesis that easy implementation and hence "rapid prototyping" are possible (although it is sometimes quite surprising what does not work when it comes to piecewise functions). Building upon objects and operations which are already there, we created objects and operations for modelling a racing course and for working with motion functions offering additional feedback such that meaningful work within realistic restrictions is possible. Having a CAS environment, much more work can go into the didactical design of the microworld.

Being already in a professional environment, it is easier to have a smooth transition from educational to professional software which is particularly interesting for engineers in a university environment.

ACKNOWLEDGEMENTS
The author would like to thank the discussion group members (Ph. Kent, M. Yerushalmy, A. Heck, N. Challis, B. Torrence) for an interesting and inspiring discussion during the CAME meeting in Utrecht, July 2001.
The author also thanks the referees for their comments and remarks.

REFERENCES

Alpers, B., Steeb, S. (1998). *Flexible Mechanical Engineering Guidelines – A Maple-based Tool for Cam Design*, MapleTech Vol. 5, No. 2/3, pp. 40-48.

Alpers, B. (2001): *Mathematical Application Projects for Mechanical Engineers – Concept, Guidelines and Examples*, Proc. of the 5th Int. Conference on Technology in Math. Teaching (ICTMT 5), Klagenfurt.

Alpers, B. , Gerny, M. (2002): *Formula 1 – A Mathematical Microworld with CAS – Learning Opportunities and Real Usage*, in preparation.

Breidenbach, D., Dubinsky, E., Hawks, J., Nichols, D. (1992). *Development of the Process Conception of Function*, Educational Studies in Mathematics, vol. 23, pp. 247-285.

Crawford, K. (1995). *Designing, Exploring and Interacting: Novice Activities in the Boxer Computational Medium*, in: diSessa, A., Hoyles, C., Noss, R. (Eds.): Computers and Exploratory Learning, Springer: Berlin Heidelberg, pp. 443-462.

Cuoco, A. (1995). *Computational Media to Support the Learning and Use of Functions*, in: diSessa, A., Hoyles, C., Noss, R. (Eds.): Computers and Exploratory Learning, Springer: Berlin Heidelberg, pp. 79-107.

diSessa, A., Abelson, H. (1986). *Boxer: A Reconstructible Computational Medium*, Communications of the ACM, vol. 29, no. 9, pp. 859-868.

diSessa, A. (1995). *The Many Faces of a Computational Medium: Teaching the Mathematics of Motion*, in: diSessa, A., Hoyles, C., Noss, R. (Eds.): Computers and Exploratory Learning, Springer: Berlin Heidelberg, pp. 337-359.

Dörfler, W. (1993). *Computer Use and Views of the Mind*, in: Keitel, C., Ruthven, K.: Learning from Computers: Mathematics Education and Technology, Springer: Berlin Heidelberg, pp. 159-186.

Dugdale, S., Kibbey, D. (1990). *Beyond the evident content goals: Part I. Tapping the depth and flow of the educational undercurrent*, Journal of Mathematical Behavior, vol. 9, pp. 201-228.

Edwards, L. (1992). *A Comparison of Children's Learning in Two Interactive Computer Environments*, Journal of Mathematical Behavior, vol. 11, pp. 73-81.

Edwards, L. (1995). *The Design and Analysis of a Mathematical Microworld*, Journal of Educational Computing Research, vol. 12, no. 1, pp. 77-94.

Edwards, L. (1998). *Embodying Mathematics and Science: Microworlds as Representations*, Journal of Mathematical Behavior, vol. 17, no. 1, pp. 53-78.

Eisenberg, M. (1995). *Creating Software Applications for Children: Some Thoughts About Design*, in: diSessa, A., Hoyles, C., Noss, R. (Eds.): Computers and Exploratory Learning, Springer: Berlin Heidelberg, pp. 175-196.

Even, R. (1998). *Factors involved in Linking Representations of Functions*, Journal of Mathematical Behavior, vol. 17, no. 1, pp. 105-121.

Goldin, G., Shteingold, N. (2001). *Systems of Representations and the Development of Mathematical Concepts*, in: Cuoco, A., Curcio, F. (Eds.): The Roles of Representation in School Mathematics. 2001 Yearbook of the NCTM, Reston, pp. 1-23.

Gomes Ferreira, V. (1997). Students' perceptions of functions articulated in dynamic microworlds, Proc. of the 3rd Int. Conference on Technology in Math. Teaching (ICTMT 3), Koblenz.

Gravemeijer, K., Doorman, M. (1999). *Context Problems in Realistic Mathematics Education: A Calculus Course as an Example*, Educational Studies in Mathematics vol. 39, no. 1-3, pp. 111-129.

Hillel, J. (1993). *Computer Algebra Systems as Cognitive Technologies: Implications for the Practice of Mathematics Education*, in: Keitel, C., Ruthven, K.: Learning from Computers: Mathematics Education and Technology, Springer: Berlin Heidelberg, pp. 18-47.

Hillel, J., Lee, L., Laborde, C., Linchevski, L. (1992). *Basic Functions Through the Lens of Computer Algebra Systems*, Journal of Mathematical Behavior, vol. 11, pp. 119-158.

Kent, P. (1999). *Trajectories for Learning*, Unpublished Masters dissertation, Institute of Education, University of London: http://pkent.homestead.com/ .

Kent, P. (2000). *Expressiveness and Abstraction with Computer Algebra Software*, Journée d'étude: Environnement informatiques de calcul symbolique et apprentissage des mathématique, Rennes, France.

Mayes, R., Schmidt, L., Swandon, T. (1996). *ACT in algebra: The function concept*, Int. DERIVE® Journal, vol. 3, no. 2, pp. 17-37.

McDermott, L. (1992). *Research as a Guide for the Design of Intelligent Learning Environments*, in: Tiberghien, A., Mandl, H. (Eds.): Intelligent Learning Environments and Knowledge Acquisition in Physics, Springer: Berlin Heidelberg, pp. 245-264.

Mellar, H., Bliss, J. (1993). *Expressing the Student's Concepts versus Exploring the Teacher's: Issus in the Design of Microworlds for Teaching*, Journal of Educational Computing Research, vol. 9, no. 1, pp. 89-113

Nemirovsky, R., Kaput, J., Roschelle, J. (1998). *Enlarging Mathematical Activity from Modeling Phenomena to Generating Phenomena*, Proc. of 22nd Annual Meeting of the Int. Group for the Psychology of Mathematics Education (PME 22), Stellenbosch, Vol. 3, pp. 287-294 (ed. by Olivier, A., Newstead, K.).

Noss, R., Hoyles, C. (1996): *Windows on Mathematical Meaning – Learning Cultures and Computers*, Dordrecht: Kluwer.

O'Callaghan, B. (1998). *Computer-Intensive Algebra and Students' Conceptual Knowledge of Functions*, Journal for Research in Mathematics Education, vol. 29, no. 1, pp. 21-40.

Roschelle, J., Kaput, J., Stroup, W. (2000). *SimCalc: Accelerating students' engagement with the mathematics of change*, in: Jacobson, M., Kozma, R.B. (Eds.): Innovations in Science and Mathematics Education, Mahwah: Lawrence Erlbaum, pp. 47-76.

Schnepp, M., Nemirovsky, R. (2001). *Constructing a Foundation for the Fundamental Theorem of Calculus*, in: Cuoco, A., Curcio, F. (Eds.): The Roles of Representation in School Mathematics. 2001 Yearbook of the NCTM, Reston, pp. 90-102.

Speiser, B., Walter, C. (1996): *Second Catwalk: Narrative, Context, and Embodiment*, Journal of Mathematical Behavior, vol. 15, no. 4, pp. 351-371.

Tall, D. (1994). *Computer Environments for the Learning of Mathematics*, in: Biehler, R. et al. (Eds.): Didactics of Mathematics as a Scientific Discipline, Dordrecht: Kluwer, pp. 189-199.

Vinner, S., Dreyfus, T. (1989). *Images and Definitions for the Concept of Function*, Journal for Research in Mathematics Education, vol. 20, no.4, pp. 356-366.

Weigand, H.-G. (1997). *Understanding the Function Concept while Using a Computer Algebra System – An Exploratory Study*, in: Weigand, H.-G. et al. (Eds.): Selected Papers from the Annual Conference on Didactics of Mathematics, Leipzig.

Yerushalmy, M. (2001). *A Response to Neil Challis: "Expressive and explicit CAS – is it enough?"*, Proc. of the CAME 2001 Symposium, Utrecht
http://ltsn.mathstore.ac.uk/came/events/freudenthal/4-Reaction-Yerushalmy.pdf .

Yerushalmy, M., Shternberg, B. (2001). *Charting a Visual Course to the Concept of Function*, in: Cuoco, A., Curcio, F. (Eds.): The Roles of Representation in School Mathematics. 2001 Yearbook of the NCTM, Reston, pp. 251-268.

BIOGRAPHICAL NOTES

Burkhard Alpers studied mathematics, history and educational sciences and received his PhD in Mathematics from the University of Hamburg. For several years, he worked at the Corporate R&D Division of Siemens AG on topics in network management. He is currently professor of mathematics and computer science in the Department of Mechanical Engineering at the University of Applied Sciences in Aalen, Germany. He is interested in educational and industrial applications of computer algebra, optimization and geometric modelling in CAD.