Coordinating representations through programming activities: an example using Logo

Ana Isabel Sacristán Rock
Cinvestav, México
asacrist@mail.cinvestav.mx

This paper appeared in the book:
Hitt, F. (ed) (2002) Representations and Mathematics Visualization.
Working Group: Representations and Mathematics Visualization (1998-2002) - North American Chapter of the International Group for the Psychology of Mathematics Education / Cinvestav-IPN. ISBN 968-5226-10-5. p127-142

Abstract: *The research literature has pointed to difficulties encountered by students in interpreting and establishing links between different types of representational registers. Here I present results from a study where a Logo-based computational microworld for the exploration of infinity and infinite processes, was meant to facilitate the construction and articulation of diverse types of representations of those infinite processes. I provide a couple of examples from case studies illustrating some of the ways in which students used and coordinated the elements of the exploratory medium to construct meanings for the infinite.*

On the importance of constructing links between different representational registers

We can consider that two main representational registers are those that are often denoted as “visual” and “symbolic” (even though visual representations are also a form of symbolizing). That is, some representations are of visual form (e.g. the graph of a function); others are purely symbolic or algebraic, lacking a graphical aspect. Visual and symbolic representations are complementary, each representation holding a different form of interpreting the information. The visual (graphical) representation of a mathematical situation gives a global view (as explained for

instance by Larkin & Simon, 1987), while, on the other hand, the symbolic representation involves more local analysis. An integration of both types of representations appears to be essential for constructing a richer meaning of the mathematical object.

The research literature, however, has pointed to the difficulties encountered by students in interpreting and establishing links between different types of representational registers. It has been found that pupils tend to prefer symbolic manipulation to visual interpretation, perhaps because the latter requires moving towards a higher cognitive level — for 'decoding' the visual information (see Dreyfus & Eisenberg, 1990; Eisenberg & Dreyfus, 1986; 1991). Many students have difficulties in reading diagrams, and one of the things that has been observed is that students do not easily make links between visual representations and analytical thought (see for instance Artigue, 1990; Presmeg, 1986; Hillel & Kieran, 1987; Dreyfus et al., 1990).

Findings such as these have led many researchers (e.g. Cuoco and Goldenberg, 1992) to advocate incorporating more representations and types of thinking, particularly visual ones, into school mathematics. The need for including more of the visual aspect in mathematics education is evident, particularly in contexts that link it to the numerical and symbolic aspects of mathematics, although it is clear that the mere presence of multiple representations does not guarantee that the learner will construct cognitive links between them. As Noss et al. (1995, p.191) have pointed out, "we should not take for granted that building links *between* representations is straightforward, or that the more representations which are available, the better it is for learning". On the other hand, these same authors (Noss & Hoyles, 1996) have explained that computational environments offer a setting where the objects and relationships can become *meaningful* through actions within the microworld where students can generate and articulate mathematical relationships that are general to the computational situation in which they are working.

We consider that it is the construction of cognitive links between representations and pieces of knowledge that facilitates the learning of a concept, and that this construction is in turn facilitated when there are more opportunities of *engaging — constructing* and interacting — with multiple modes of external representations of the concept. In fact, Wilensky (1991) uses this idea of building connections in his discussion of what makes knowledge abstract or concrete. He points out that the "formal is often abstract because we haven't yet constructed the connections that

will concretize it" (ibid., p.202). Thus, an abstract concept can become concrete by relating to it in as many ways as possible. As he puts it:

> "The more connections we make between an object and other objects, the more concrete it becomes for us. The richer the set of representations of the object, the more ways we have of interacting with it, the more concrete it is for us. ... "This view will lead us to allow objects not mediated by the senses, objects which are usually considered abstract — such as mathematical objects — to be concrete; provided that we have *multiple modes of engagement with them and a sufficiently rich collection of models to represent them."* (ibid., p.198-99; emphasis added)

Our theoretical approach thus follows the 'constructionist' paradigm (see Harel & Papert, 1991). We adopt the position that the construction of meanings involves the use of representations; that representations are tools for understanding; and that the learning of a concept is facilitated when there are more opportunities of constructing and interacting with, as diverse as possible, external representations of a concept.

A study centering on the infinite: using the computer to provide an interaction between diverse representations of infinite processes

For many years we have been studying students' conceptions and difficulties in the learning of concepts related to the mathematical infinity (e.g Sacristán, 1993). The concept of infinity is central to calculus: infinite processes form the basis for the concept of limit; it is also present in other important areas of mathematics. This concept, however, is recognized as difficult and has historically been the origin of paradoxes and confusions. Furthermore, the areas of mathematics where infinity occurs are those that have traditionally been presented to students mainly from an algebraic/symbolic perspective, which has tended to make it difficult to link formal and intuitive knowledge.

There is evidence, however, that in a context that combines numerical and geometrical contexts through the use of algebraic language, some of the obstacles observed in single-representation situations in the reasoning related to infinite processes and sets, seem to disappear (Waldegg, 1988). This is an important finding which supports the idea that by building connections between different types of representations (in this case through algebraic language) some of the difficulties

that arise when working in a single context can be diminished. We thus took it upon ourselves to create situations in which the learning of the infinite infinity could be facilitated by incorporating the use of the computer and the representational systems it provides, even though we are aware that attempts to use the computer for the learning of the concept of limit have pointed to difficulties (e.g. Monaghan, Sun & Tall, 1994).

We postulated that at least some of the infinite processes found in mathematics, could become more accessible if studied in an environment that facilitates the construction and articulation of diverse types of representations, including visual ones. Based on this premise, we built a computational set of open tools (diSessa, 1997) — a microworld — using the Logo[1] programming language, for the study and exploration of infinity and infinite processes. Being aware that the conceptions that students could develop would be conditioned by this environment, the main focus of the research (see Sacristán, 1997) was to investigate, through detailed case-studies of 8 subjects, these students' developing conceptions of infinity and infinite processes as *mediated* by the Logo-based microworld. Part of this research was to investigate how students articulated and coordinated the different representations of the microworld in order to construct meanings for the infinite, which is the focus of this paper.

Description of the microworld

The microworld (see Sacristán, 1999) was a programming[2] environment where infinite processes —infinite sequences and series— were explored through the construction (using Turtle Geometry) of different visual models for representing these processes: such as "unfolding[3]" spirals and fractal figures, with a

[1] The rationale behind the use of Logo is discussed elsewhere (see Sacristán, 1997), but one of the reasons for choosing this medium was the built-in visual interface (through Turtle Geometry) that Logo has, which can be very helpful for the requirements of incorporating the visual representations and creating the interaction with the symbolic code. Also, because figures have to be created linearly in Turtle Geometry this is helpful in the observation of the process as it unfolds (it adds movement and dynamism).

[2] In the programming environment students wrote, created, modified and explored procedures that represented, in different ways, the infinite processes under study.

[3] The visual models under study could be seen in a dynamic way as they were constructed (as they "unfolded") on the screen: this allowed for the observation of the evolution in time and behaviour of the underlying processes, eliminating the limitation of only observing the final state (the result of the process).

complementary numerical analysis (e.g. students constructed tables of values of, for instance, the distances forming the figures at different levels of the construction process). Thus, the infinite processes were explored through three types of representations:

- the symbolic code,
- different types of visual models (such as unfolding spirals; and fractals), and
- numeric representations, to complement and validate the visual observations (taking advantage of the computing capabilities of the computer for reaching high terms in numerical (or other types of) sequences).

The activities were divided into two sections:

a) explorations of classical infinite sequences and their corresponding series, through geometric models such as spirals, bar graphs, and staircases.
b) fractal explorations.

For the explorations of infinite sequences and their corresponding series, we began with geometric models such as spirals, since they seemed to be a straightforward way of translating arithmetic series into geometric form. For instance, in the 'spiral' type of representation each term of the sequence is translated into a length, visually separated by a turn, so that the total length of the spiral corresponds to that of the sum of the terms (the corresponding series). Thus, for instance for the sequence $\{1/2^n\}$, the visual process and added lengths of the spiral would represent the series: $\frac{1}{2}+\frac{1}{4}+\frac{1}{8}+\frac{1}{16}+\cdots$, a notation which is descriptive of the process involved — the ellipsis points indicating an indefinite continuity of the process (a *potentially* infinite process). On the other hand, in the symbolic computer code, the same series can also be represented by a notation corresponding to $\sum_{n=1}^{\infty}\frac{1}{2^n}$, which is an object in itself (an *actual* infinite object[4]). This could serve to illustrate how the same (mathematical) object can be represented both as a (complete, integral) *object*

[4] This is independent of the convergence or divergence of the series, although when there is convergence it is easier to think of the series as a "complete" object, e.g. when $\sum_{n=1}^{\infty}\frac{1}{2^n}=1$.

as well as in terms of a *process*.

The activities included programming by the students of visual models such as spirals[5], bar graphs, staircases, and straight lines of sequences such as $\{1/2^n\}$, $\{1/3^n\}$, $\{(2/3)^n\}$, $\{2^n\}$ etc., and then $\{1/n\}$, $\{1/n^{1.1}\}$,... , $\{1/n^2\}$. The Bar graph, Steps and Line models are illustrated in figures 1, 2, and 3, respectively.

Through the observation of the visual (and numeric) behavior of the models, students were able to explore the type of convergence, or divergence, of a sequence and that of its corresponding series, and predict the behavior at infinity. The different geometric models for the same sequence provided different perspectives of the same process. But an aspect that was considered important for this, was that the students carried out the transformations of the models themselves by changing the computer code. It was intended that this involvement would help build links between the symbolic representation (in the code) and the different models.

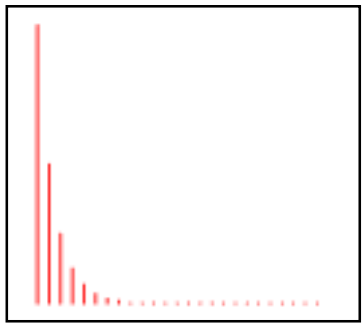

Figure 1. Bar graph corresponding to the sequence $\{1/2^n\}$.

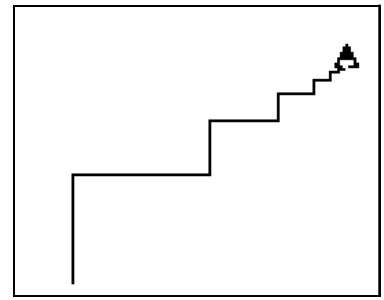

Figure 2. Steps model corresponding to the sequence $\{1/2^n\}$.

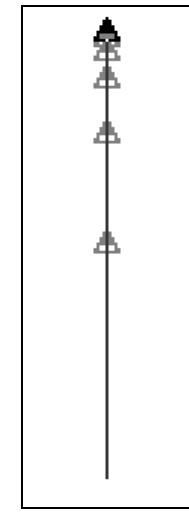

Figure 3. Line model corresponding to the sequence $\{1/2^n\}$.

Because the production of the graphical and numerical representations was carried out through the construction of the Logo symbolic programming code, the

[5] I would like to acknowledge the books on infinity by Mason (1988), in particular, and Hemmings & Tahta (1984), which served as inspiration for some of the geometric models used in the study presented here.

different types of representations were explicitly linked one with the other through the first: the procedural code. In fact, the code can act as an *isomorphism* between the different (visual) models, as well as acting as the link between all the representations, and the subject.

In the second part of the microworld, the exploration of fractal figures centered mostly on the study of the recursive structures of the Koch curve and snowflake (Figure 4) and the Sierpinski triangle (Figure 5), and involved the encounter of some apparent paradoxes such as that of a finite area bounded by an infinite perimeter. Through these sequences and fractals activities we intended to confront students with the idea of "what happens in the infinite" by allowing them to visualize an infinite process through the computer-based approximations.

An essential aspect of the functioning of the microworld was the programming activity on the part of the students. All of the students therefore wrote their own procedures using the Logo programming language, although the activities were suggested by the researchers (who also served as guide). Programming the computer to draw on a computer screen can be thought of as a process of interaction between contexts, from the symbolic code to the visual and conversely. The symbolic code of the computer language can serve, within the computer context, to "explain", model, or represent the process and it encapsulates the structure behind the process. Thus, the programming activities emphasized the interaction between different types of (re-)constructed representations of the infinite processes.

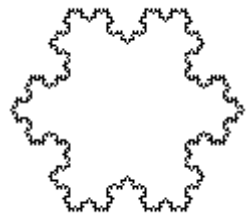

Figure 4: The Koch snowflake

Figure 5: The Sierpinski Triangle

Examples of the ways in which students coordinated the different representations within the microworld: linking the symbolic code and the visual output

Here, I present a few examples from the activities with the microworld to illustrate some of the ways in which students used and coordinated the elements of the exploratory medium *to construct meanings for the infinite*. They illustrate how the

microworld gave the students means to *make sense of what they saw on the screen via the programming code*: the interactions between the code and its outputs. Emphasis is also placed on the role of the *structure* of the procedures, particularly the *iterative or recursive* structure (which was present in all the procedures for constructing the visual models of infinite sequences), and its relationship to the visual structure. There are two facets to the phenomenon:

(a)The link of the endlessness of the process represented on the screen, with the iterative structure of the code; and

(b) The use of the symbolic recursive structure of the code to *visualize the self-similar visual behavior*.

a. Endless movement and the link with the recursive (iterative) structure of the code. An example of the interaction from visual to symbolic, with a return from the symbolic to the visual.

In the first microworld activity, the subjects were given the following procedure, and asked to predict the output and its behavior:

```
TO DRAWING :L
PU
FD :L
RT 90
WAIT 10
DRAWING :L/2
END
```

This procedure makes Logo's turtle walk (without leaving a trace ¾ the Pen is up) through a spiral with arms each having half the length of the previous one (see Figure 6). It should be noted that as there is no stop condition, the procedure continues indefinitely[6]. It was designed to induce students to reflect on the behavior of the turtle and the procedure itself.

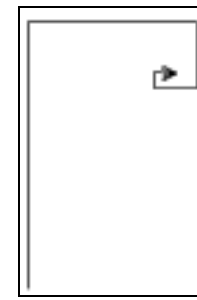

Figure 6. Spiral output of the initial DRAWING procedure (representing the sequence $\{1/2^n\}$).

This initial activity produced interesting results: in particular, the students had to visualize the actual pattern without relying on the computer drawing; it induced students to try to make sense of the relationships between the code and the graphical output: For instance, most students did not expect

to see the turtle endlessly spinning without leaving a trace. In order to explain to themselves this unexpected behavior and make sense of why the turtle was endlessly spinning, the students had to re-examine the procedural code. Victor was one student who immediately remarked that the procedure would never stop because the recursive structure of the code represented an infinite process. He explained it was because the procedure called itself without anything telling it to stop, so it never would; the process of turning and walking half the previous distance would continue repeating itself and would never stop. By analyzing the code Victor was able to connect to it the behavior of the visual output (in this case the *movements* of the turtle) since he correctly predicted the outcome and was able to justify that visual behavior through the code. He *linked the recursive structure of the code with the infinitude of the process.*

A modified procedure (with the Pen down) produced an inward spiral with the turtle then turning endlessly in its center. Victor and his partner, Alejandra, pointed out that although the turtle seemed to be just turning in the same spot, in reality there was "a variation". There were two factors here: a) The turtle kept turning and b) the turtle turned at same spot. The first factor could have served as an indicator that the process continued, but it was the fact that the students seemed to be able to disregard the *visual appearance* of the turtle — spinning in apparently the same spot — that suggests that they understood that the underlying (mathematical) process continued, and that they were able to link the output with the code and the process.

Later in the activities, when the students modified the procedure to give out numeric values, they would confirm the continuation of the process by still getting an output of values, even when the turtle seemed stuck:

Alejandra: Apparently it is stopping on the screen, but it is still walking because we are still getting the values.

It was thus that students were able, via a process of experimenting backwards and forwards from code to figure, to make sense of the behavior of the turtle, which seemed to be spinning on the same spot realizing that the amount that the turtle moved each time was halved. The key point here is that the analysis of the code allowed them to:

[6] Later, when the students became aware of the recursive structure, all of them eventually added a stop condition, which also served as an important investigation tool for making sense of the relationship between code and figure (see Sacristán, 1997).

1) recognize in the recursive structure a potentially infinite process; and

2) to quantify the movement, to explain that although the turtle seemed to be turning without moving forward, in reality there *was* a variation.

Thus by coordinating the visual and symbolic — in the order visual to symbolic to visual — and later complementing it through numerical explorations, their understanding of the process became integrated and potentially misleading visual appearances could be ignored.

This interplay between the code and its outputs, which led students to make sense of what they observed by linking all the elements, is illustrated in Figure 7:

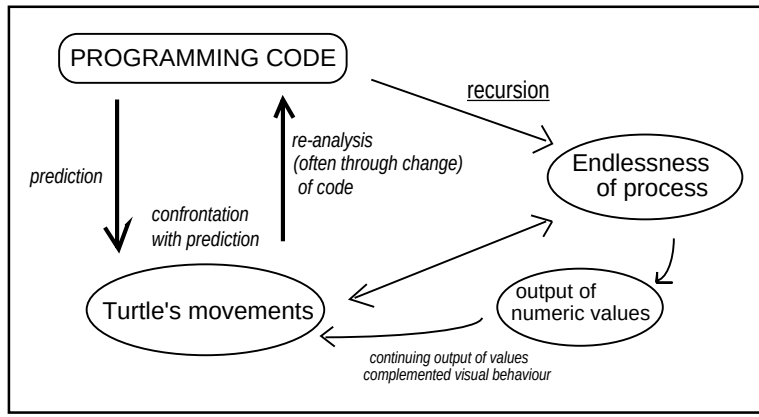

Figure 7. The interplay between the code and its output to make sense of the endless movement: the graphical imagegains meaning from the symbolic representation.

b. Using the *recursive* structure of the code to predict a self-similar visual structure: the code 'encapsulates' the process. *Visualizing the output from the symbolic code: an example of the interaction from symbolic to visual.*

Whereas the recognition of the (tail) recursive structure in the code explained the endlessness of the process, which involved going from the visual to the symbolic, this recursive structure also served to predict and *visualize* the figure produced by the code — a process from symbolic to visual. For instance, when the students were unable to see the deeper levels in the visual representation of the sequence under study (e.g. they noticed that the center of the spiral model looked like a point), some students blamed this on the resolution and were able to compensate for the deficiencies of the screen by using the information provided in the symbolic

structure of the code to visualize those levels. Victor, for example, in the activity described earlier, explained that even though the center of the spiral looked like a point, the figure did *not* become a point, and would always have the same spiral shape, even at its center. Another student, Martin, explained this same point as follows:

> Martin: What happens is that there is a part that our eyes can no longer perceive. Inside [the spiral] it continues the same way, because it is the same process that continues... If we used a magnifying glass and looked at that little square there, we would see like all this part [the full spiral].

This is a key issue: the visual self-similar structure is a reflection of the recursive structure of the code (and of the infinite iterative nature of the process). And the programming code thus serves for "visualizing" beyond the visual image. The programming code can thus be said to embody or "encapsulate" the entire process. An infinite process would, of course, take infinitely long to be generated; but the symbolic code (which generates it) holds the entire process in *latent* form, as would a mathematical formula, and its structure reflects the structure of the process.

This connection between the structure of the code and that of the figure was of course more obvious in the fractal explorations. For example, the construction of the Koch curve procedure was based on the theoretical structure of the visual figure — the structure of the procedure mirroring the way the process would be (visually) constructed. Then as the Koch curve was generated, the students would make sense of the figure (the fact that each part contains the whole) by relating it back to the structure of the code (which calls itself). The same correspondence between code and figure was found for all the fractal figures investigated, and students were able to predict a recursive visual structure from the observation of the structure of the code.

It is interesting to point out that students explicitly recognized the value of having a recursive code that defined and therefore was connected to the process both visually and in general. As they pointed out, the self-similar/recursive structure of the code allowed them to have an idea of what was going to happen subsequently, particularly since the figure can only have so much resolution. This was best explained by one student, Jesús, who pointed out that it was the recursive structure of the procedure, which helped him realize (and reflected the fact) that the figure would repeat itself in a self-similar way, adding that it was the procedure (i.e. the code), which helped to understand what happens at infinity:

Jesús: I would say that our most powerful weapon is recursion, which is what allows us to be aware of the details...
In those shapes that repeat themselves, it is the same part, which is the basis...
...
And because we understand the language, it was the recursion which helped us understand better, and do a better analysis... going from the figure to the procedure, or from the procedure to the figure...
In any case, once I knew that the procedure was recursive, I more or less had an idea of what would happen later, because the drawings by themselves do not have enough resolution...
The Logo procedures give support and help define what happens at infinity. That is, they include the notion of infinity, and that is really helpful. They help convince us, or confirm the ideas we may have....

Summary: A sketch of the microworld

There were three main representational elements involved in the microworld: symbolic (the programming code), visual (geometric figures), and numeric (numerical values). A mathematical process could be represented in each of these complementary forms: symbolically in the code, and visually or numerically by running the code. Thus, the structure of the microworld was such that the process and its different representations were all linked through the computer code (see Figure 7 further below). All of the above elements, including the process, and their characteristics are described in more detail below:

1.- The mathematical process.

All the microworld activities involved a process of construction of an infinite sequence (this includes the fractal activities, since fractals are constructed through *geometric* sequences). In the case of sequences, the process was first described as an iterative action, involving repeated operation on a variable: e.g. the action of halving an element and repeating this for the resulting process. It was later defined as a symbolic function formula — a Logo function in the programming code —, which encapsulated the process.

2.- The programming activity and code.

The programming activity and the code were the means that integrated all the elements and acted as a “control structure”.

3.- The visual/graphical representations.

These representations had several relevant characteristics during the explorations:

(i) First, the visual element was an entry point for the explorations. That is, although the visual figures required the computer procedure, in most cases the realization of what the procedural code was describing and how it operated did not emerge until after the visual figures were generated and an attempt was made to relate the observed phenomena on the screen with the code that produced it.

(ii) These representations served to "visualize" the behavior of the processes in two ways:

On the one hand, they provided a *global* view of the processes: the process is synthesized in the figure.

On the other hand, the computer provided the added benefit that these visual images *gradually* unfolded, so that each element of the figure (e.g. the visual representations of the terms of the sequence) could be specially observed in relation to the previous ones, giving the sense that a *process* was taking place. This allowed for a local analysis, particularly since the computer simultaneously generated the numeric representations which quantified the process.

(iii) The element of *movement* also provided information that is difficult to access. For example, the turtle turning in the same spot conveyed the idea that the process was continuing even when there was no longer any visible change.

(iv) The visual representations had a *self-similar geometric structure*, which resulted from the recursive structure of the code and could be related to the latter, forming an additional link between the two.

(v) Visual models could be *transformed* into others, through the use of the same programming code. Each model had particular characteristics that helped in the understanding of the behavior of the other models. The different models are *complementary*, each adding its own particular perspective on the process: However, since they were all produced by the same code, the connection between them could be made apparent, facilitating the integration of the information. Particular visual appearances could therefore become less dominating.

(vi) Additionally, a valuable exploration tool for the visual representations was the definition of a scale variable in the code. In a sense the scale acted as a

"zoom" function: for instance, it sometimes enabled the students to look "deeper" into the graphical representation (e.g. into the spiral model or the Koch curve). Through this activity they could, for example, appreciate the self-similar characteristic of fractal figures. The use of the scale variable had another value in that its relationship with the figure had to be made explicit through the programming activity. In this sense it could be considered to have an additional function to that of a mere "zoom button".

4.- The numeric values.

The numeric output provided a means for working with the visual and symbolic representations. It was another representation of the mathematical process that added numeric *precision* to both the terms of the sequences and to the limit values. Furthermore, the students could "visualize" *in the numeric values* the limit value of the process or its divergence. The numeric output also provided an additional connection between the visual and the symbolic code acting as a means for quantifying the visual models and confirming the accuracy of the code.

Because the values were generated *simultaneously* with the visual representation as it unfolded, these values had a concrete connection, rather than being abstract numbers whose meanings could be obscure.

Figure 8 is a schematic of a typical paradigm case of the use of the representational tools of the microworld; in it we see the following (refer to numbers in diagram): (1) The mathematical process is symbolically defined in the code. (2) By running the programming code, a visual representation, which models the entire process (3), is produced. Additionally, the visual representation is produced by the *movements* of the turtle, gradually unfolding: this allows the process to be observed sequentially and for its characteristic *behavior* to be highlighted. (4) There are different visual models that can be produced, each providing a different visual perspective on the process. The switch between models is done through the code whose structure, and the process it defines, remain invariant: the models are *isomorphically* constructed. (5) Numeric values can also be produced through the same code; this links them to the visual representations and both (6) simultaneously gradually unfold. (Numeric values are also produced through complementary procedures — also representing the same process). (7) The numeric representations serve to add precision and give confirmation of the observed behavior of the process.

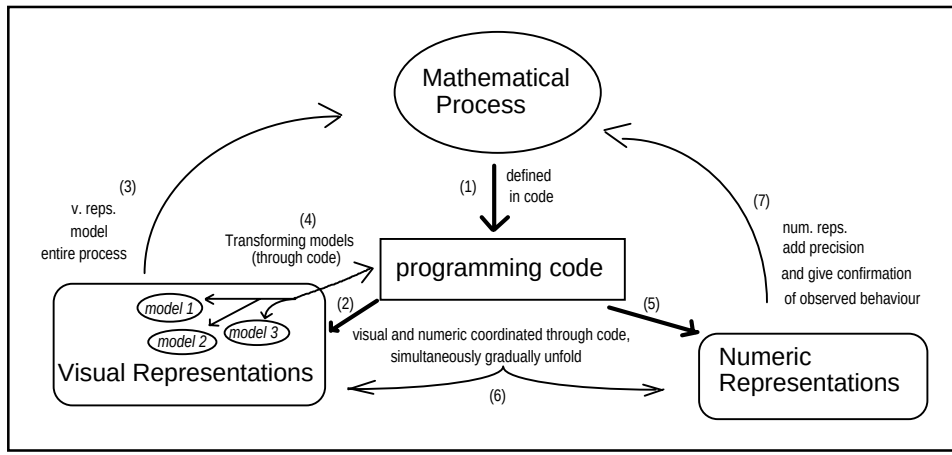

Figure 8. The representational elements of the microworld and their interactions.

I would like to emphasize the way in which the representational tools were used and compare the structure of this microworld with the use of multiple representations illustrated in the work of other researchers (where, although the different representations are linked in the sense that they work simultaneously, the inner workings of the connections between them are not available for the student to manipulate and re-construct). In the study presented here, through interacting with the programming code, the students themselves created and controlled the way in which the multiple representations worked. It is in this sense that — unlike other multiple representation environments (e.g. Kaput, 1995) where the tools are fixed — this microworld is what diSessa (1997) would call an *open tool set:* the students were able to reconstruct or redesign the tools, and the links between them, and express themselves through the programming activity.

References

Artigue, M. (1990), “Difficultés cognitives et didactiques dans la construction de relations entre cadre algébrique et cadre graphique” *PME-14 Proceedings*, México 1990, Vol. 1, p. 11-18.

Cuoco & Goldenberg (1992), “ Mathematical Induction in a Visual Context”, in *Interactive Learning Environments*, Vol. 2, Issues 3 & 4, p. 181-203.

diSessa, A. (1997), “Open Toolsets: New Ends and New Means in Learning Mathematics and Science with Computers”, in *Proceedings of the 21st Conference of the International Group for the Psychology of Mathematics Education*, Erkki Pehkonen (Ed.), p. 47 - 62.

Dreyfus, T. & Eisenberg, T. (1990), "On difficulties with diagrams: Theoretical issues", *PME-14 Proceedings,* México 1990, Vol. 1, p.27-31.

Dreyfus et al. (1990), "Advanced Mathematical Thinking", in ,Nesher & Kilpatrick (eds) *Mathematics and Cognition*, ICMI Study Series, Cambridge Univ. Press, p.113-134.

Eisenberg, T. & Dreyfus, T. (1986), "On Visual versus Analytical Thinking in Mathematics" in *PME-10 Proceedings*, London 1986, Vol.1, p.153-158.

Eisenberg, T. & Dreyfus, T. (1991), "On the reluctance to visualize in mathematics", in Zimmermann, W. & Cunningham, S. (eds): *Visualization in Mathematics*, MAA Notes Series, Vol. 19, p. 25-37.

Harel, I. & Papert, S. (eds.), (1991), *Constructionism*; Ablex Publishing Corporation, Norwood, NJ.

Hemmings, R. & Tahta, D. (1984), *Images of Infinity*, Leapfrogs insight series. Leapfrogs, 1984.

Kaput, J. (1995), "Creating Cybernetic and Psychological Ramps from the Concrete to the Abstract: Examples from Multiplicative Structures", in Perkins, D. et al. (eds), *Software Goes to School: Teaching for Understanding with New Technologies;* p. 130- 154.

Larkin, J.H. & Simon, H.A. (1987) "Why a Diagram is (Sometimes) Worth Ten Thousand Words", *Cognitive Science* 11, p.65-99.

Mason, J. (1988), *Approaching Infinity*. Center for Mathematics Education. The Open University.

Monaghan, Sun & Tall (1994) "Construction of the Limit Concept with a Computer Algebra System" in *Proceedings PME-18*, Lisboa, 1994, p. 279-286.

Noss et al. (1995), "The dark side of the moon", in Sutherland, R. & Mason, J (eds) *Exploiting Mental Imagery with Computers in Mathematics Education*, p. 190-201.

Noss, R. & Hoyles, C. (1996), *Windows on Mathematical Meanings. Learning cultures and computers.* Kluwer Academic Press.

Presmeg, N.C. (1986), "Visualization in High-School Mathematics" in *For the Learning of Mathematics* 6, p.42-46.

Sacristán, A.I. (1993), "Los Obstáculos de la intuición en el aprendizaje de procesos infinitos" *Educación Matemática* Vol. 3 No.1 April 1993, México.

Sacristán, A.I. (1997) *Windows on the Infinite: Constructing Meanings in a Logo-based Microworld.* PhD Dissertation. Institute of Education, University of London.

Sacristán, A.I. (1999) "Creating meanings for the infinite through a Logo-based microworld" en Extending Educational Horizons in the Spirit of Logo: a 20th Century Epilog. *Proceedings (Supplement) of the Seventh European Logo*

Conference EuroLogo '99, Nikolov, R.; Sendova, E.; Nikolova, I. & Derzhanski, I. (eds). Virtech Ltd., Sofia, Bulgaria: 1999 (ISBN: 954-9582-03-5). Pp. 475-484.

Waldegg, G. (1988), *Esquemas de Respuesta ante el Infinito Matemático. Transferencia de la Operatividad de lo Finito a lo Infinito.* Doctoral dissertation. Centro de Investigación y de Estudios Avanzados, México.

Wilensky, U. (1991) "Abstract Meditations on the Concrete, and Concrete Implications for Mathematics Education", in Harel, I. & Papert, S. (eds.) *Constructionism*, p.193-204; Ablex Publishing Corporation, Norwood, NJ.

Logos are everywhere. On the clothes we wear, on the phones we use, and on the food we buyâ€”weâ€™re surrounded. Some logos are incredibly straightforwardâ€”a letterform or a pictorial representationâ€”and some are more complex.Â â€œUSCYBERCOM plans, coordinates, integrates, synchronizes and conducts activities to: direct the operations and defense of specified Department of Defense information networks and; prepare to, and when directed, conduct full spectrum military cyberspace operations in order to enable actions in all domains, ensure US/Allied freedom of action in cyberspace and deny the same to our adversaries.â€Â The white stripes passing through the letterforms give the illusion of equal signs in the lower areas of the letters, which represents equality. Coordinating representations through programming activities: an example using Logo. 209. Ana Isabel SacristÃ¡n Rock Cinvestav, MÃ©xico.Â Here I present results from a study where a Logo-based computational microworld for the exploration of infinity and infinite processes, was meant to facilitate the construction and articulation of diverse types of representations of those infinite processes. I provide a couple of examples from case studies illustrating some of the ways in which students used and coordinated the elements of the exploratory medium to construct meanings for the infinite. On the importance of constructing links between different representational registers. In advertising, logos uses logic to persuade viewers. We'll cover the logos definition with examples so you can nail this technique in your own projects.Â Logos is the persuasive technique that aims to convince an audience by using logic and reason. Also called â€œthe logical appeal,â€ logos examples in advertisment include the citation of statistics, facts, data, charts, and graphs. In Aristotleâ€™s rhetorical triangle, ethos appeals to character, pathos appeals to emotion, and logos appeals to logic and reason. As the â€œheadiestâ€ of the three main rhetorical strategies, logos uses reasoned discourse and logical arguments to convey a point of view and win over the audience. Logos examples in ads: An iPhone commercial that highlights the latest featur Graphical representations, such as those used in the commercial Houdini system [32], represent an animation by a dataflow network (see Figure 4.7). An acyclic graph is used to represent objects, operations, and the relationships among them. Nodes of the graph have inputs and outputs that connect to other nodes, and data are passed from node to node by arcs of the graphics that are interactively specified by the user. A node represents an operation to perform on the data being passed into the node, such as an object description. A transformation node will operate on an object description passed