

## Appendix 1: the six ReMath DDA profiles and the NETLOGO profile

### I. Aplusix epistemological profile

#### 1. Objects

What are the main objects represented and manipulated by/with the DDA?

##### 1.1 Framework

Aplusix uses a general notion of expressions which includes equations, inequations and systems.

The current framework of Aplusix has two major features: real numbers and exact calculations:

- Numbers are considered as elements of  $\mathbb{R}$  (set of real numbers), variables have values in  $\mathbb{R}$ , solutions of equations, inequations and systems are searched in  $\mathbb{R}$ ;
- Aplusix and the user only perform exact calculations.

##### 1.2. Objects: Algebraic expressions with a usual representation

Representation: usual two-dimensional representation

Syntax: syntax is checked by the system.

Creation of objects: an object can be created by the user and the system.

Manipulation: totally WYSIWYG with direct manipulation

Computational:

- User's computations are free and are checked as correct or not
- Computer computations are performed on user's request

Curriculum: totally compatible

##### 1.3 Objects: Algebraic expressions with a tree representation

Representation:

- First form: tree representation with operators in internal nodes and variables and numbers in leaves
- Second form: mix- representation (combination of usual and tree with expand/collapse commands)

Syntax: syntax is checked by the system.

Creation of objects: an object can be created by the user and the system.

Manipulation: totally WYSIWYG with direct manipulation

Computational: User's transformations of a representation into another are checked as correct or not

Curriculum: innovative

##### 1.4 Objects: Graphical representations

Representation:

- Graphical representations of expressions of one variable are represented in a plane
- Graphical representations of equations and inequations of one unknown are represented in two spaces:
  - o In a plane where  $f=g$  is represented with a curve of  $f$  and a curve of  $g$
  - o In a line where the solution set of  $f=g$  is represented

- Graphical representations of systems of linear equations of two unknowns are represented in two spaces:
  - o In a plane where each equation is represented with a line
  - o In the same plane where the intersection of the lines is represented as a set of solution.

Creation of objects:

- a curve can be created only by asking the system to draw a curve of an expression,
- dynamicity: when the expression is modified, the curve representing it is updated.

Manipulation: colour, thickness, position (which curve is behind) and zoom can be chosen by the user.

- Computational: User's transformations of a representation into another are checked as correct or not

Curriculum: totally compatible.

### 1.5 Objects: Numbers, Variables, Polynomials, Functions, Equations, Inequations, Systems

All these objects are considered and manipulated through algebraic expressions and curves

### 1.6 Concept: Denotation, equivalence, correct calculations

Algebraic expressions have mathematical objects as denotations:

- Numerical expressions have functions as denotations,
- Equations, inequations and systems have sets of solution as denotations.

Two algebraic expressions are equivalent if and only if they have the same denotation.

The main reasoning process is "reasoning by equivalence" where correct calculations lead to equivalent expressions.

Manipulation:

- The equivalence between the two expressions of a user's calculation is shown by the system for each calculation step,
- The interpretation of the feedback signs related to checking equivalence between two expressions is left to the user (and his/her teacher).

Computational: The computation of the equivalence is done by the system.

Curriculum:

- Correct calculations is an element of any curriculum,
- Equivalence is explicit or not, depending of the curriculum,
- The notion of denotation is probably implicit in any curriculum.

### 1.7 Concept: algebraic expressions are intermediate objects between strong mathematical objects they represent (their denotations) and their concrete representations on media:

- Algebraic expressions can be represented in their usual form,
- Algebraic expressions can be represented in a tree form.

Manipulation: The fact that two representations represent the same expression is indicated by the system at the end of exercises of the type "Write a tree/usual form of this usual/tree expression".

Curriculum: The notion of multiple representations of an algebraic expression is probably absent of all curriculums.

## 2. Connections

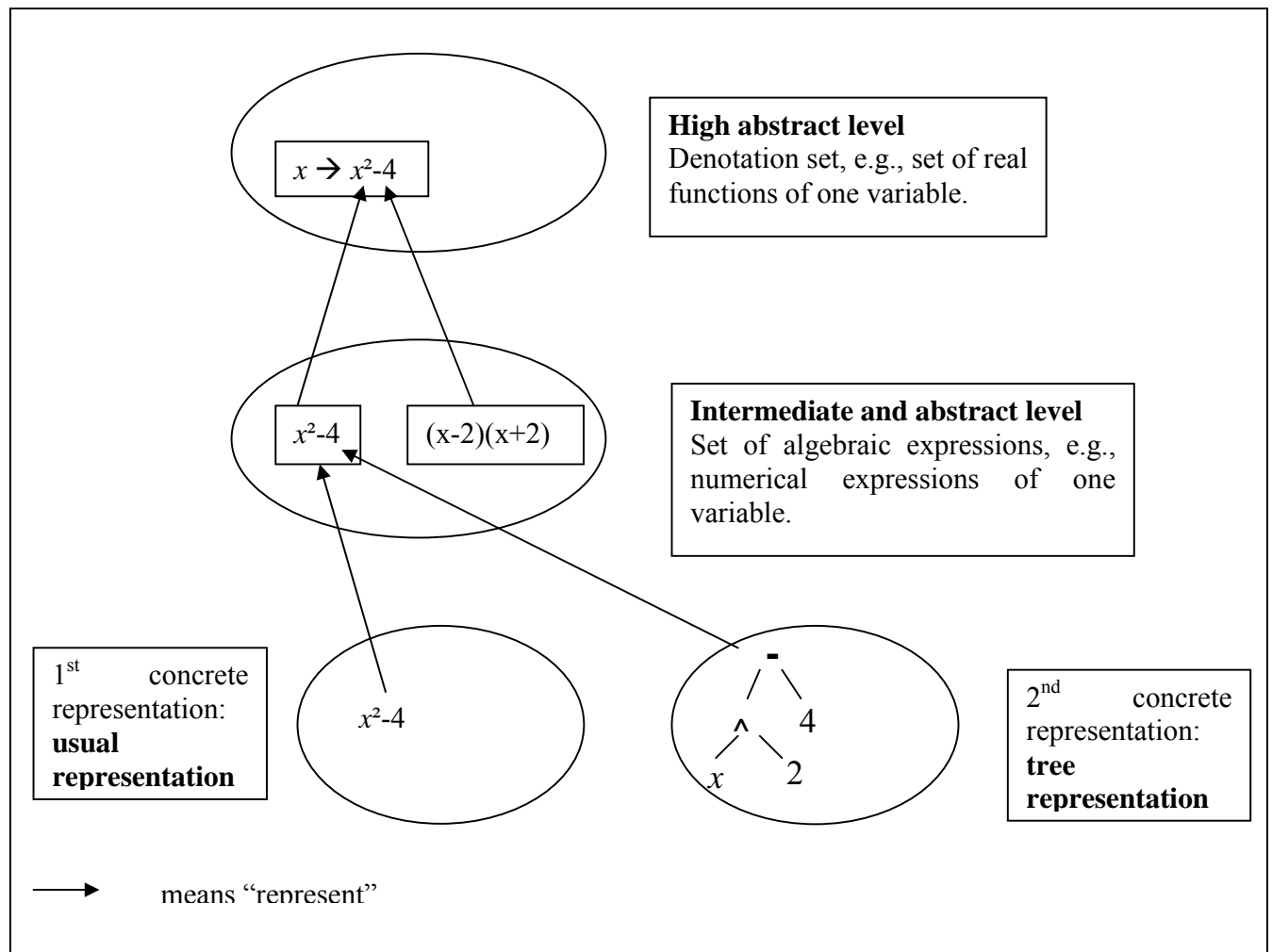


Figure 1. Link between different sets of objects and representations spaces. Graphical representations, not in the figure, are concrete high level objects. Arrows mean "represent".

## 3. Activities

Aplusix DDA offers at least 4 kinds of activities:

- (1) Formal exercises of numerical or algebraic calculation (ex. calculate a numerical expression, solve an equation, develop, simplify or factor an algebraic expression...). These are, in Duval's terms, treatment tasks in the usual (or tree) register of representation.
- (2) Word problems, which can be considered as modelling tasks.
- (3) Exercises of conversion of given expressions between usual and tree registers.
- (4) Exercises where a task is to explore relationships between two expressions without equivalence feedback by using the graphical representation of the expressions (exploration task).

*How the objects and connections favour this kind of activities?*

- (1) Formal exercises are defined in the DDA, which enables the system to recognize the types of exercises and check the appropriateness of the solving process.

- (2) Specific exercise editor allows creating word problems whose solving is based on producing and transforming algebraic expressions. The author of the exercise defines the nature of the expected answer and the way the student's solution is compared to it, which enables the system to check for the correctness of the student's solving process.
- (3) Conversion tasks of the type "Write a tree/usual form of this usual/tree expression" have been defined with the implementation of the tree representation in the system (cf. 1.7 above).
- (4) Exploration tasks are favoured by the possibility to represent given expressions in usual and graphical representations at the same time and by real-time updating of graphical representations when usual representations are modified.

*Are these activities compatible with existing curricula and practice, or are they innovative?*

- (1) Formal exercises are traditional school algebra exercises, completely compatible with curricula and practices.
- (2) Modelling tasks are also traditional exercises completely compatible with curricula and practices.
- (3) Tree representation of expressions is not yet a curricular object in France, although recent mathematics curricula for junior high school mention a tree representation and its articulation with usual and natural language registers as a means to help students distinguish between procedural and structural aspects of expressions. Nevertheless, this type of task is not yet compatible with school practices.
- (4) In school algebra, graphical representations of expressions are mainly used in solving equations, inequations and systems, but never to check equivalence between expressions. In this sense, the exploration tasks articulating usual and graphical registers in Aplusix are compatible neither with the curricula nor the practices.

#### 4. Pedagogical/intervention agenda

##### *1. Innovation vs acceptance - where does the DDA design stand?*

Aplusix brings innovative aspects to the teaching and learning algebra by enabling interactions between different registers of representations of algebraic expressions, which responds to the general requirements of French mathematics curricula to articulate different representations of mathematical objects. The software can be considered as highly acceptable since most tasks it offers are compatible with the curricula.

##### *2. Distance to traditional curriculum - where does the DDA design stand?*

For the above mentioned reasons, Aplusix is rather close to traditional curriculum although it proposes some types of tasks (exploration tasks) that are not compatible with it.

##### *3. Status of representations traditional, innovative, to be handled, related to others etc*

Representations of algebraic expressions in Aplusix are traditional. The innovation lies in the possibility of their direct manipulation by the user and in the possibility to have simultaneously two different representations of a same algebraic expression at one's disposal.

##### *4. Scope for mathematization or only about mathematics?*

The scope of Aplusix is both for mathematization through modelling tasks and about mathematics through the other types of activities.

## II. Casyopée epistemological profile

### 1. Objects

Casyopée's two main objects are

1. “algebraic functions”
2. “geometrical dependencies”.

A “function” is defined by a formula and a domain. The formula involves a function variable and possibly parameters. Casyopée provides means for creating sets of ordered real numbers, possibly including parameters, in order to define domains.

The parameters can be treated both formally and numerically by way of animation. Constraints can be set on parameters in order to adapt to all situations: for instance if the parameter is intended to model a measure, it can be defined as positive.

The main capabilities relative to “algebraic functions” are:

In the “symbolic window”

- calculations (e.g. expanding or factoring formulas, integrating or differentiating functions, solving equations...);
- graphic representations (with different functionalities such as zooming, changing axis scales...);
- numerical or formal values (such as particular values or limits);
- proof capabilities (theorem are available inside Casyopée that the user can apply to functions in order to prove signs or extrema or variations or zeros);

In the Dynamic Geometry window

- geometrical representations (curves)

“Geometrical dependencies” exist in figures of the Dynamic Geometry window. They can exist as covariation between geometrical objects, as covariation between measures and as functional dependencies between measures. The main capabilities to create figures and study geometrical dependencies are:

- geometrical constructions including free points; Casyopée's algebraic objects –functions, expressions and parameters- can be used.
- creation of geometrical calculations (well-formed formulas involving measures and symbolic objects);
- numerical explorations of geometrical calculations;
- choice of a measure as an independent variable for studying functional dependencies
- numerical explorations of functional dependencies;
- “exportation” of these functional dependencies into the symbolic window: Casyopée computes a domain and a formula and creates the corresponding “algebraic function”.

The central concept in Casyopée is “mathematical function of one variable”. This concept is seen through complementary aspects:

- dependencies in a physical system. In, 2D geometry is the physical system that has been chosen. Dependencies can concern geometrical objects and measures (lengths, areas...). Dependencies between measures can be explored as co variations (simultaneous variations against time) or as functional dependencies: Casyopée offers means to choose an independent variable for the latter case,.
- real function of a real variable with
  - o standard mathematical representations

- Algebraic formula and domain
- Graph and curve
- tables
  - equivalence (same domain, and equivalent formulas)
  - properties (sign, growth) that constitute opportunities for problem solving and proof.

This view of functions is encouraged by the curriculum, but impossible to address properly by most students with paper-pencil or existing (dynamic geometry or CAS) software. In this sense, the representations are curriculum compatible *and* innovative.

## 2. Connections and Activities

The grid of table 1 explains how the objects and representations are connected in Casyopée and what are the activities aimed at. The rows refer to the above aspects of the notion of function, distinguishing dependencies between geometrical objects and between measures.

The columns refer to different representations of concepts in calculus. They separate representations of dependencies between two elements that can be thought of enactively or from images or approximations (enactive-iconic), and those that imply an explicit exact expression and thus an algebraic language. “Explorative” activities with ‘enactive-iconic’ representations involve experience of dependencies in a dynamic figure, as well as work on numerical values of measures, and ‘explorations’ on graphs and tables of approximate values. These activities involve complex semiotic systems with rich connections. With regard to dependencies between geometrical objects, dragging points and naming particular elements in the figure (for instance segments representing either the domain, on which the independent variable varies, or the range, on which the dependent variable varies) offer a first semiotic system. Then the dependencies between magnitudes offer an opportunity to speak of properties of dependencies (increasing, decreasing, reaching an extremum) in a more numerical way. Finally observing the graph of a corresponding mathematical function, the dependency is explored and spoken of in the more standard language of domain, values, shape (parabola, straight line)...

Activities in the three other columns involve the algebraic language. Student activities in algebra have been classified by Kieran (2004) into three categories: generational, transformational, and global / meta-level that correspond these columns: “The generational activities of algebra involve the forming of the expressions and equations that are the objects of algebra (...). The transformational (rule-based) activities include, for instance, collecting like terms, factoring, expanding, substituting (...) The global / meta-level mathematical activities include problem solving, modelling, noting structure, studying change, justifying, proving, and predicting.”

	Representations Types of activities	Enactive-Iconic	Algebraic		
		Explorative	Generational	Transformational	GlobalMeta
Objects represented	Dependencies between geometrical objects	Exploration: various shapes in the figure			Using parameters to create generic figures
	Dependencies between magnitudes	Exploration of numerical dependencies	Introducing 'geometrical calculations' Choosing a variable		Working on 'generic' dependencies
	Mathematical Function	Local trace of graph of the function, global recognition of the graph	Getting an algebraic formula and a domain as a model	Operating transformations, finding an algebraic proof.	Working on 'families' of functions

Table 1

Casyopée provides specific means to connect generational and enactive-iconic activities at the level of magnitudes: after independent and dependent variables have been built and chosen using a formalisation specific to magnitudes, Casyopée can “export” a dependency into the symbolic window, creating the corresponding “algebraic function”. Meaning can develop from this connection: considering an algebraic expression (domain and formula) for a function is motivated and takes sense when the function is conceived as a model of an enactive phenomenon.

Transformational activity is a domain where Casyopée helps the student thanks to the underlying symbolic kernel.

Rich connections exist between transformational and enactive-iconic activities. For instance students can connect the notion of equivalence, central in the transformational activities with the coincidences of graphs.

Global-Meta activities include using algebraic means to express generality. Casyopée offers parameters for that. It helps to connect global-meta and enactive-iconic activities by treating parameters both formally and numerically by way of animation. General objects involved in a ‘generic’ geometrical dependency are expressed by using parameters: for instance arbitrary points in a geometric figure, different from free points, are expressed as coordinate points involving parameters; a general circle is defined with a radius depending on a parameter. The enactive-iconic exploration involves then two different gestures: dragging a free point helps to explore a particular dependency while using sliders to animate the parameters contributes to the exploration of a family of dependency. The algebraic model of the dependency computed by Casyopée is a family of functions, that is to say a function whose domain and formula depends on formal parameters. With regard to semiotics, students identify two different literals (parameters and function variable) as corresponding to two different gestures of exploration. Using the model to solve a problem in the physical system brings together algebraic formal treatment of parameters and the enactive-iconic interpretation of their values: for instance students compute the maximum of the function modelling a dependency in a figure and get a value depending on parameters, then they interpret this value as a ‘generic’ position of a free point.

*Are these activities compatible with existing curricula and practice, or are they innovative?*

The links between the algebraic and enactive-iconic representations allowed by Casyopée are a foundation for activities connecting the semiotic systems elaborated for expressing dependencies and the algebraic language. All these activities are recommended by existing curricula, but not realistic without technology.

### 3. Pedagogical/intervention agenda

#### *1. Innovation vs acceptance - where does the DDA design stand?*

Casyopée's dynamic interface has been designed for easy use by the students: no specific syntax is needed. However, teachers can be surprised if they are used to CAS command driven interface. Also, because of the link with the symbolic window, dynamic geometry window's constraints can be surprising for teachers used to numerical dynamic geometry environments.

#### *2. Distance to traditional curriculum - where does the DDA design stand?*

As explained above, Casyopée is compatible with the curriculum. It aims to make the curriculum feasible rather than to change it.

#### *3. Status of representations traditional, innovative, to be handled, related to others etc*

Computational representations in Casyopée try as much as possible to be consistent with 'standard' mathematics.

#### *4. Scope for mathematization or only about mathematics?*

Modelling (a typical mathematization process) is central in the Casyopée extension. Casyopée helps also doing mathematics.

Kieran, C. (2004). The Core of Algebra: Reflections on its Main Activities, in Stacey et al. (eds.), *The Future of the Teaching and Learning of Algebra: the 12th ICMI Study*. Springer 2004



### III. ALNUSET epistemological profile

#### 1. Objects

The mathematical objects of reference for Alnuset are :

- Variables, unknowns and parameters
- Algebraic expressions and polynomials
- Algebraic propositions
- Numerical sets
- Functions

#### Representations

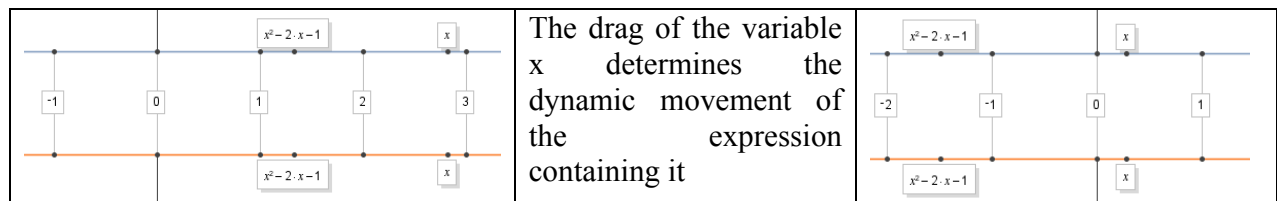
Here we describe how the mathematical objects of reference are represented in the three environments of Alnuset and how their representations can be manipulated by the user.

##### *Algebraic Line*

Exploiting the digital technology, the number line has been enriched with the possibility to directly manipulate mobile points associated to letters dragging them along the line. According to the way the mobile point is acted with respect to the task at hand, it can assume respectively the role of variable, unknown and parameter in a quite transparent way.

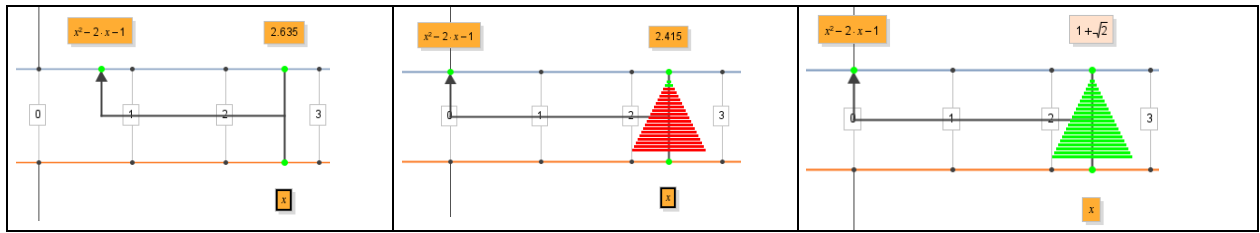
*Algebraic expressions and polynomials are associated to points on the line whose positions depend both on the position (on the line) of the letters that are contained in their form and on the structure of operations that characterize their forms.*

When the user drags the mobile point of a letter, the computer refreshes the positions of the points associated to the expressions or to the polynomial containing such a letter, in an automatic and dynamic manner.

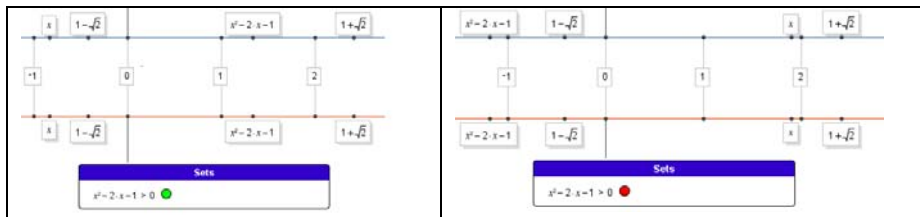


We observe that *a mobile point associated to a letter assumes the role of a variable when it is dragged along the line while it assumes the role of parameter when it is not dragged.*

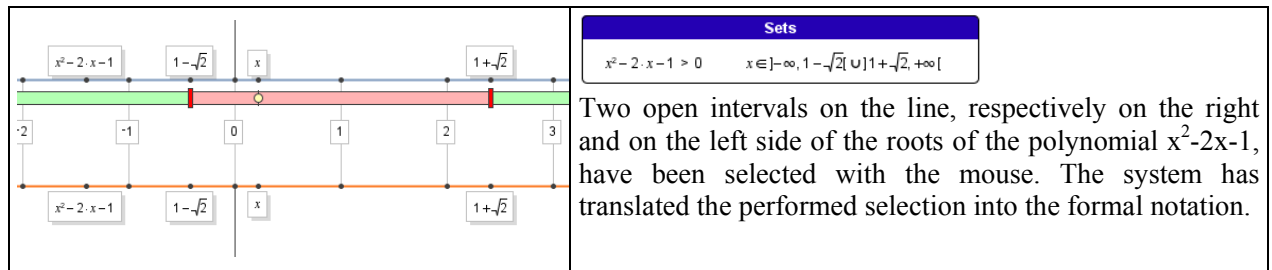
Through a specific command of this environment it is possible to find the roots of polynomials with integer coefficients. *The root of a polynomial can be found by dragging the variable on the algebraic line in order to approximate the value of the polynomial to 0.* When this happens, the exact root of the polynomial is determined by a specific algorithm of the program and it is represented as a new point on the line.



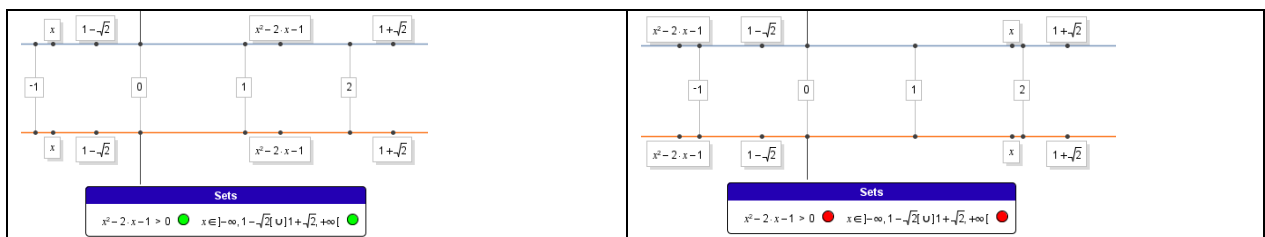
Algebraic propositions are represented in a specific window of this environment named “Sets” as associated to coloured markers that are under control of the system. *The green/(red) colour of the marker of the proposition at hand means that the proposition is true/(false) for the actual value of the letter on the line.*



Through a graphical editor, it is possible to edit the truth set of a proposition.



Once the truth set of a proposition has been edited, it can be validated by using a specific feedback of the system. At this regard we note that the numerical sets are associated to coloured (green/red) markers that are under control of the system. *The green/(red) colour for the numerical set means that the actual variable value on the line is/(is not) an element of the set. Through the drag of the variable on the line, colour accordance between proposition marker and set marker allows the user to validate the defined numerical set as a truth set of the proposition (see figure below).*



We note that the mobile point associated to a letter assumes both the role of unknown (whose value makes the proposition true or false) and the role of element of a numerical set that can belong or not to a previously defined set

### Algebraic Manipulator

In this environment algebraic expressions, algebraic propositions and numerical sets can be manipulated formally through a set of basic rules that correspond to the basic properties of addition, multiplication and power operations, to the equality and inequality properties between algebraic expressions, to basic logic operations among propositions and among sets. Every rule produces the simple result of transformation that is reported on the icon of its corresponding command on the interface, and this makes easy the control of the rule and of its result.

Moreover, this manipulator provides not expert users with cognitive supports in the development of specific manipulative skills. A first support is the possibility to explore, through the mouse, the hierarchical structure that characterises the expression or the proposition to be manipulated. By dragging the mouse pointer over the elements of the expression or proposition at hand (operators, number, letters, brackets...), as feedback the system dynamically displays the meaningful part of the expression or proposition determined by such pointer. In this way it is possible to explore all meaningful parts of an expression in the different levels of its hierarchical structure. Another feedback occurs when a part of expression has been selected. Through a pattern matching technique, the system, as feedback, activates only the rule of the interface that can be applied on the selected part of expression. This is a cognitive support that can be used to explore the connection among the transformational rules of the interface, the form on which it can be applied, and the effects provided by their applications.

User Rules		Show	Import...	Export...	Clear
<b>Addition</b>	<b>Multiplication</b>				
$A+B \Leftrightarrow B+A$	$A \cdot B \Leftrightarrow B \cdot A$				
$A+(B+C) \Leftrightarrow (A+B)+C$	$A \cdot (B \cdot C) \Leftrightarrow (A \cdot B) \cdot C$				
$A \Leftrightarrow A+0$	$A \Leftrightarrow A \cdot 1$				
$A+(-A) \Leftrightarrow 0$	$A \cdot 0 \Leftrightarrow 0$				
$A-B \Leftrightarrow A+(-B)$	$-A \Leftrightarrow -1 \cdot A$				
$a_1+a_2+\dots \Leftrightarrow x$	$1 \Leftrightarrow -1 \cdot -1$				
$n \Leftrightarrow a+b$	$A \cdot \frac{1}{A} \Leftrightarrow 1$				
<b>Powers</b>	$\frac{A}{B} \Leftrightarrow A \cdot \frac{1}{B}$				
$A^n \Leftrightarrow A \cdot A \dots$	$\frac{1}{A_1 \cdot A_2 \dots} \Leftrightarrow \frac{1}{A_1} \cdot \frac{1}{A_2} \dots$				
$A^{n_1+n_2+\dots} \Leftrightarrow A^{n_1} \cdot A^{n_2} \dots$	$a_1 \cdot a_2 \dots \Leftrightarrow x$				
$(A_1 \cdot A_2 \dots)^n \Leftrightarrow A_1^n \cdot A_2^n \dots$	$n \Leftrightarrow p_1 \cdot p_2 \dots$				
$(A^n)^m \Leftrightarrow A^{n \cdot m}$	<b>Distribute / Factor</b>				
$A^{-n} \Leftrightarrow \frac{1}{A^n}$	$A \cdot (B_1+B_2+\dots) \Leftrightarrow A \cdot B_1+A \cdot B_2+\dots$				
$A^{\frac{1}{n}} \Leftrightarrow \sqrt[n]{A}$	<b>Solving</b>				
<b>Computation</b>	$A \leq B \Leftrightarrow B \leq A$				
$A \Rightarrow (A)$	$A \leq B \Leftrightarrow A-B \leq 0$				
Remove extra ()	$A \leq B+T \Leftrightarrow A-T \leq B$				
Simplify numerical expression	$A+T \leq B \Leftrightarrow A \leq B-T$				
Expand	$T \cdot A \leq B \Leftrightarrow A \leq \frac{B}{T}$				
Collect					
Eliminate variable					
<b>Logic and Set</b>	$\frac{p}{q} \leq B \Leftrightarrow A^p \leq B^q$				

$a^2 - b^2$   
 $a^2 + 0 - b^2$   
 $a^2 + a \cdot b + (-a \cdot b) - b^2$   
 $a^2 + a \cdot b + -1 \cdot (a \cdot b) - b^2$   
 $a^2 + a \cdot b + -1 \cdot a \cdot b - b^2$   
 $a^2 + a \cdot b + -1 \cdot a \cdot b + -b^2$   
 $a^2 + a \cdot b + -1 \cdot a \cdot b + -(b \cdot b)$   
 $a^2 + a \cdot b + -1 \cdot a \cdot b + -1 \cdot (b \cdot b)$   
 $a^2 + a \cdot b + -1 \cdot a \cdot b + -1 \cdot b \cdot b$   
 $a \cdot (a+b) + -1 \cdot a \cdot b + -1 \cdot b \cdot b$   
 $a \cdot (a+b) + -1 \cdot (a \cdot b + b \cdot b)$   
 $a \cdot (a+b) + -1 \cdot ((a+b) \cdot b)$   
 $a \cdot (a+b) + -1 \cdot (a+b) \cdot b$   
 $a \cdot (a+b) + -1 \cdot b \cdot (a+b)$   
 $(a+(-1 \cdot b)) \cdot (a+b)$   
 $(a-1 \cdot b) \cdot (a+b)$   
 $(a-b) \cdot (a+b)$

This figure shows a part of the commands available with the interface and an example of algebraic transformation.

The figure shows a characteristic of the interactivity of this manipulator: the selection of a part of an expression determines the activation of the commands of the interface that can be applied on it. This characteristic can help students to explore the systems of rule for the algebraic transformation and the effects they produce

### Function environment

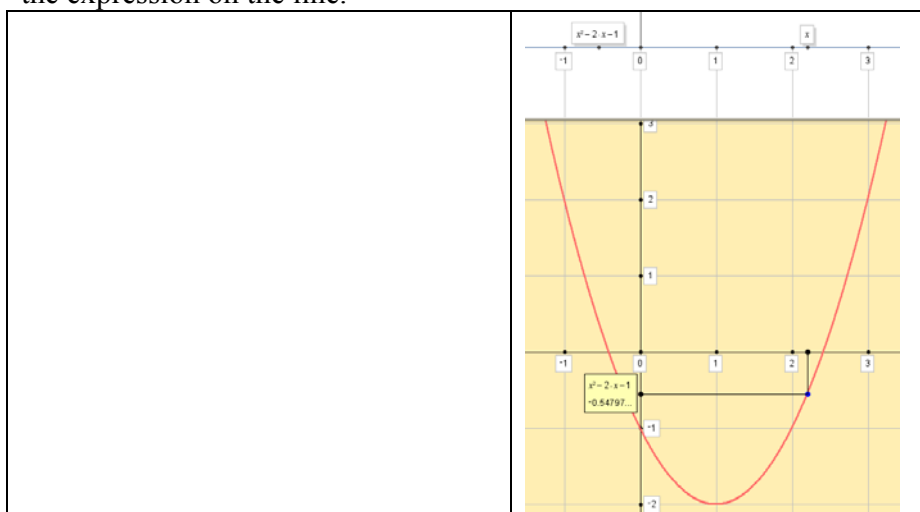
The referential mathematical object of this environment is the function.

This environment offers the possibility to connect the dynamic functional relationship between variable and expression on the algebraic line with the graphical representation of the function associated to the expression in the Cartesian plane. As a consequence, the interface of this component has been equipped with the Algebraic line and a Cartesian plane. This idea makes this component deeply different from other environment for the representation of function in the Cartesian plane. Through a specific command and the successive selection of the independent

variable of the function, an expression represented on the Algebraic line is automatically represented as graphic in the Cartesian plane. Dragging the point corresponding to the variable on the algebraic line, two representative events occur:

- on the algebraic line, the expression containing the variable moves accordingly
- on the Cartesian plane, the point defined by the pair of values of the variable and of the expression moves on the graphic as shown in the following figure.

These representative phenomena support the integrated development of a dynamic idea of function with a static idea of such a notion. The functional relationship between variable and expression is visualized dynamically on the algebraic line through drag of the variable point, and statically in the Cartesian plane through the curve. The movement of the point along the curve during the drag of the variable on the algebraic line supports the integration of these two ideas, showing that the curve reifies the infinite couples of values corresponding to the variable and to the expression on the line.



If the expression on the line contain more than a letter, only one can be defined as variable while the others assume the role of parameters. When the letter associated to a parameter is dragged on the line, the graph change according to the role of parameter assumed by that letter.

### Concepts:

Here we describe some main concepts associated to the mathematical objects and the way the representations of Alnuset deals with them. We realize this description with respect to the three environment of Alnuset.

#### *Algebraic line*

In this environment the object algebraic expression is associated to different concepts such as the concept of equivalence between expressions, the concept of opposite expressions, the concept of reciprocal expressions.

As far as the concept of equivalence between expressions is concerned, a specific representation deals with this concept. Every point represented on the line is associated to a post-it. The post-it of a point contains all the expressions constructed by the user that denote that point. The computer automatically manages the relation among expressions, their associated points and post-it. By dragging a variable on the line, dynamic representative events can occur in a post-it.

They are very important for the construction of the concept related to the equivalence between expressions. As a matter of the fact, the presence of two expressions in a post-it may mean:

- A relationship of conditioned equivalence, if taking place at least for one value of the variable during its drag along the line
- A relationship of equivalence, if taking place for all the values assumed by the variable when it is dragged along the line.
- A relationship of equivalence with restrictions, if taking place for every value of the variable when it is dragged along the line, but for one or more values, for which one of the two expressions disappears from the post-it and from the line.

As far as the concept of opposite expressions is concerned, their position on the algebraic line is “opposite” with respect to the point 0 during the drag of their variables. Moreover, their sum always belongs to the post-it of the point 0 during the drag of their variables. The experience of this invariance is crucial to understand the notion of opposite and to justify a fundamental rule of transformation ( $a + \bar{a} = 0$ ).

As far as the concept of reciprocal expressions is concerned, the position on the algebraic line of their product always belongs to the post-it of the point 1 during the drag of the their variables. The experience of this invariance is crucial to understand the notion of reciprocal and to justify a fundamental rule of transformation associated to this notion ( $a \cdot 1/a = 1$ ).

Working in this environment several concepts emerge as associated to the way propositions are represented and manipulated. The most important are the concepts of truth value of a proposition, of a truth set of a proposition and of equivalent propositions.

The feedback of the system connected to the coloured green/red marker for propositions and sets during the drag of the variable is important to introduce the notions of the truth value and of the truth set of an algebraic proposition and to develop a discourse on their relationships.

Moreover, this feedback is of great importance for the construction of the notion of equivalent propositions as well because it allows the user to understand that equivalent propositions are characterized by the same truth set.

### *Manipulator environment*

The operative and representative characteristics of this algebraic manipulator support the development of an idea of algebraic transformation well framed from a conceptual point of view. The rules of transformation available in the interface are axioms of the theory of the algebraic transformation to be conceptualized. Once the student has realized a meaningful transformation that presents the feature of generality he can create a new rule of transformation to be used in successive algebraic manipulations. The new rule corresponds to a theorem that enriches the theory of transformation of reference for the activity.

### *Function Environment*

The operative and representative characteristics of this environment support the construction of several concepts both of algebraic and functional nature.

The dynamic functional relationship between variable and expression on the algebraic line with the graphical representation of the function in the Cartesian plane contributes to assign an algebraic meaning to the intersection of two curves (for the value of the variable that determines the intersection, the two expressions are contained in the same post-it on the algebraic line) or to the intersection of a curve with the x-axis (in this case the expression is contained in the post-it of 0)

Other examples are related to the construction of meaning for the sign of a function (in terms of the position of the corresponding expression on the line with respect to 0), or to relationships of order among functions (in term of the respective position of the expressions on the algebraic line), or to the construction of a meaning for the notion of parameter.

## 2. Connections

Here we describe the different ways in which mathematical objects and their associate concepts can be connected using the different operative and representative possibilities of the three environment of Alnuset.

a) Equivalence between algebraic expressions.

In the algebraic line two equivalent expressions belong to the same post it during the drag of their variable on the line

In the manipulator environment two equivalent expressions present the same structure through the algebraic transformation.

In the function environment two equivalent expressions are associated to the same graph.

b) Opposite expressions

On the algebraic line two opposite expressions are “opposite” with respect to the point 0 during the drag of their variables and their sum always belongs to the post-it of the point 0 whatever the value of their variable are. .

In the manipulator environment an expression A is opposite of the expression B if it is possible to prove that  $A = -B$  or that their sum is 0

In the function environment the graph of two opposite expressions are symmetrical with respect to the x axis and the graph of their sum coincide with  $x=0$

c) Reciprocal expressions

On the algebraic line the position of their product belongs always to the post-it of the point 1 during the drag of the their variables.

In the manipulator environment an expression A is reciprocal of the expression B if it is possible to prove that  $A = 1/B$  or that their product is 1

In the function environment it is possible to visualize the graphs of the two reciprocal expressions and to verify that the graph of their product coincides with  $x=1$

d) Propositions

On the algebraic line the solution of an equation or inequation can be performed using the operative and representative opportunities of the environment previously described

In the manipulator environment there are more then one possibility to find the solution. Consider for example the solution of the equation  $x^2+2=2 \cdot x+3$  as reported in the following figure A, B, and C

Figure A	Figure B	Figure C
$x^2+2 = 2 \cdot x+3$		
$x^2+2-(2 \cdot x+3) = 0$		
$x^2-2 \cdot x-1 = 0$		
$x \in \{1-\sqrt{2}\} \cup \{1+\sqrt{2}\}$		

	$x^2+2 = 2 \cdot x+3$	$x^2+2 = 2 \cdot x+3$
	$x^2+2-(2 \cdot x+3) = 0$	$x^2+2-(2 \cdot x+3) = 0$
	$x^2-2 \cdot x-1 = 0$	.....
	$(x-(1-\sqrt{2})) \cdot (x-(1+\sqrt{2})) = 0$	$x^2-2 \cdot x-1 = 0$
	$\left[ \begin{array}{l} x-(1-\sqrt{2}) = 0 \\ 2 \geq 0 \end{array} \right] \vee \left[ \begin{array}{l} x-(1+\sqrt{2}) = 0 \\ 2 \geq 0 \end{array} \right] = 0$	$x^2-2 \cdot x-1 = 0$
	$\left[ \begin{array}{l} x-(1-\sqrt{2}) = 0 \\ True \end{array} \right] \vee \left[ \begin{array}{l} x-(1+\sqrt{2}) = 0 \\ 2 \geq 0 \end{array} \right] = 0$	$\left[ \begin{array}{l} (-2)^2-4 \cdot -1 = 0 \\ x = 1 \end{array} \right] \vee \left[ \begin{array}{l} (-2)^2-4 \cdot -1 > 0 \\ x = 1-\sqrt{2} \end{array} \right] \vee \left[ \begin{array}{l} (-2)^2-4 \cdot -1 > 0 \\ x = 1+\sqrt{2} \end{array} \right]$
	$\left[ \begin{array}{l} x-(1-\sqrt{2}) = 0 \\ True \end{array} \right] \vee \left[ \begin{array}{l} x-(1+\sqrt{2}) = 0 \\ True \end{array} \right] = 0$	$\left[ \begin{array}{l} 8 = 0 \\ x = 1 \end{array} \right] \vee \left[ \begin{array}{l} 8 > 0 \\ x = 1-\sqrt{2} \end{array} \right] \vee \left[ \begin{array}{l} 8 > 0 \\ x = 1+\sqrt{2} \end{array} \right]$
	$x-(1-\sqrt{2}) = 0 \vee \left[ \begin{array}{l} x-(1+\sqrt{2}) = 0 \\ True \end{array} \right] = 0$	$\left[ \begin{array}{l} False \\ x = 1 \end{array} \right] \vee \left[ \begin{array}{l} True \\ x = 1-\sqrt{2} \end{array} \right] \vee \left[ \begin{array}{l} True \\ x = 1+\sqrt{2} \end{array} \right]$
	$x-(1-\sqrt{2}) = 0 \vee x-(1+\sqrt{2}) = 0$	.....
	$x = (1-\sqrt{2}) \vee x = (1+\sqrt{2})$	$False \vee x = 1-\sqrt{2} \vee x = 1+\sqrt{2}$
	$x = 1-\sqrt{2} \vee x = 1+\sqrt{2}$	$x = 1-\sqrt{2} \vee x = 1+\sqrt{2}$
	$x \in \{1-\sqrt{2}\} \vee x \in \{1+\sqrt{2}\}$	$x \in \{1-\sqrt{2}\} \vee x \in \{1+\sqrt{2}\}$
	$x \in \{1-\sqrt{2}\} \cup \{1+\sqrt{2}\}$	$x \in \{1-\sqrt{2}\} \cup \{1+\sqrt{2}\}$

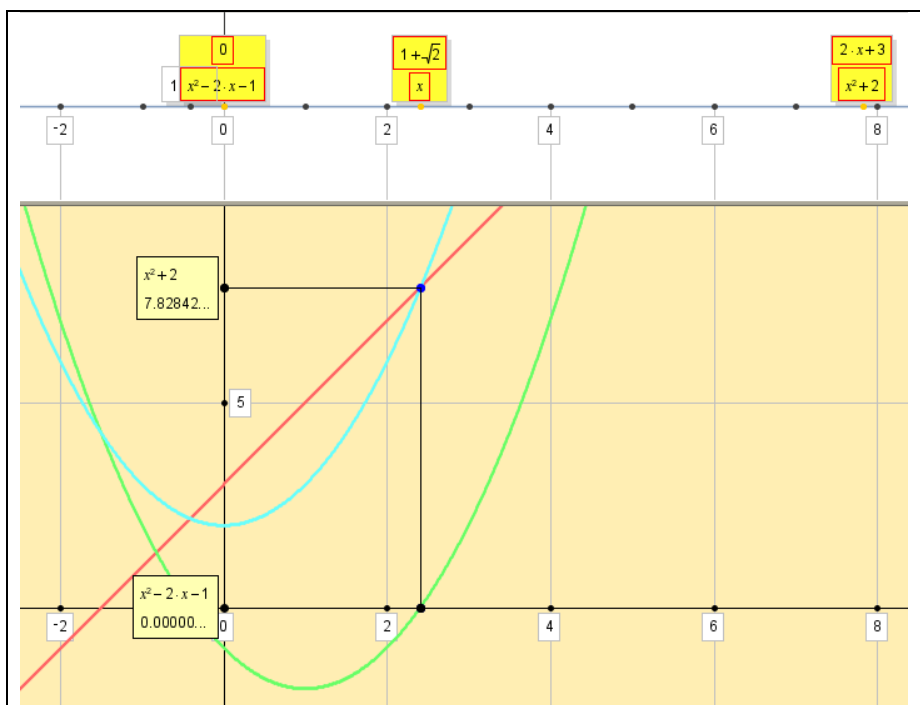
The first solution (figure A) has been realized exploiting a command that imports into the manipulator the truth set of the equation previously defined in the algebraic line environment.

The second solution (figure B) has been performed exploiting a command of the interface that factorize the polynomial on the basis of the its roots previously determined on the algebraic line (see the fourth step of the transformation) The successive steps of the solution have been realized rules of transformation of the interface.

The third solution (figure C) has been performed only through a formal approach exploiting the above reported rule created by the user that he previously had demonstrated by means of the rules of the interface

User Rules
$x^2+b \cdot x+c = 0 \Leftrightarrow \left[ \begin{array}{l} b^2-4 \cdot c = 0 \\ x = \frac{-b}{2} \end{array} \right] \vee \left[ \begin{array}{l} b^2-4 \cdot c > 0 \\ x = \frac{-b \pm \sqrt{b^2-4 \cdot c}}{2} \end{array} \right] \vee \left[ \begin{array}{l} b^2-4 \cdot c > 0 \\ x = \frac{-b \pm \sqrt{b^2-4 \cdot c}}{2} \end{array} \right]$

In the function environment the integration the dynamic functional relationship between variable and expression on the algebraic line with the graphical representation of the function in the Cartesian plane contributes to link the intersection between of the graphs related to  $x^2+2$  and  $3 \cdot x+3$  (the two expressions are containing in the same post it) with the intersection of the graph of  $x^2-2 \cdot x-1$  with the x-axis (in this case the expression is contained in the post-it of 0)



### 3. Activities

In the previous sections we have evidenced that the three environments of Alnuset offer different operative and representative possibilities to deal with the mathematical objects of reference for the system and their associated concepts. This can occur because these operative and representative possibilities can be exploited to design proof and exploration activities of quantitative, symbolic and functional nature able to support the development of the capability to use these mathematical objects and to construct appropriate meaning for them.

The activities of quantitative nature focus on numerical quantities that a variable or a literal expression indicate in an indeterminate way or on the numerical quantities that condition the equality or the inequality of a proposition. They are mainly explorative and they are aimed mainly to construct meaning for algebraic concepts associated to the mathematical objects involved in the activity. The activities of symbolic nature focus on the rules system to symbolically manipulate algebraic expressions and propositions preserving the equivalence through the transformation. They are activities mainly oriented to prove conjectures previously constructed through explorative activities of quantitative nature. The activities of functional nature focus on the link between a variable and an expression containing it and on the different representations of such link to construct mathematical concepts or to model specific situations or phenomena to be interpreted. These different types of activities can be integrated among them within the didactical practice. The integration of these different types of activities can innovate the teaching of algebra, of functions and of numerical sets. For example the operative and representative possibilities of the algebraic line environment can be easily exploited to design explorative activities in the field of algebra or in the domain of the properties of numerical sets. The way in which the manipulator environment has been realized supports the development of proof activities in a natural way.

These explorative and proof activities of different nature can be introduced into the standard curriculum to improve the teaching and learning of mathematics. For example the experimentations performed in these two years have evidenced that explorative activities within



the algebraic line environment can be easily integrated in the standard curriculum to mediate the construction of abilities and of meanings of the algebra curriculum.

Finally, we observe that the Alnuset operative and representative possibilities can be exploited to deeply innovate the curriculum of algebra. As matter of the fact they allow the teacher to change not only the way in which the contents of the current curriculum are taught but also the curriculum itself, namely what can be taught and the temporal sequence of what is taught.

#### 4. Pedagogical/intervention agenda

The analysis of the Alnuset operative and representative possibilities realized both by Italian and French researchers has highlighted that they are rather different from those reported in school manual (think for example to the operative and representative possibilities of the algebraic line). Nevertheless they can be easily learned because they can be controlled on the basis of the perceptive, kinesthetic and spatial experience of the subjects. Moreover, the performed experimentations have demonstrated that the familiarization phase with these operative and representative possibilities is generally very short because they are very transparent with respect to the results that they produce and to the way they produce such results.

Finally, the researchers have evidenced that the discourse to justify and to interpret them not only is compatible with the discourse of the school manual but it is strongly favored by the way in which the operative and representative possibilities of Alnuset can be controlled at the perceptive, spatial and kinesthetic level. In fact many operative and operative events can be metaphorically used to speak of the properties of the mathematical objects, processes and relations involved in the activity. Think for example to the use that can be done of the mobile point on the line to speak metaphorically of variable, unknown and parameter or to the use of the colored markers associated to algebraic propositions and numerical sets to speak metaphorically of truth of a proposition or of a truth set of a proposition.

## IV. MoPiX epistemological profile

### 1. Objects

#### a) Time

We identify time as a significant object within MoPiX as it is a parameter in all MoPiX equations and is essential to the animation of models, which takes place in time. Time is measured in discrete units and in Version 1 may only progress in a forward direction. It is started and stopped by clicking the play/stop button. On clicking the start button, the value of time is always reset to 0. (Version 2.0 also includes buttons that allow time to be progressed stepwise in single units and to be run backwards or set to a specific value. This was not available during the teaching experiments.)

#### b) Cartesian plane

The MoPiX stage may be considered a Cartesian plane with points identified by pairs of  $x$  and  $y$  coordinates. The unit of measurement is a screen pixel. Conventional aspects of the representation of such a plane such as axes, grid, etc, are not present; there are thus no overt signs of its presence. On the other hand, the use of  $x$  and  $y$  variables within equations imply its presence.

#### c) Equations/Functions

Equations are represented in a way that is recognizably similar to but not identical to standard algebraic notation as used within the traditional curriculum. Components of equations include numbers, variables, parameters and functions. The form of variable, parameter and function names is not restricted as it is in conventional mathematical notation and meaningful names may be used as in computer programming paradigms. Primitives and non-primitive variables, parameters and functions appearing in the equations provided in the equation library have names that may be construed as motivated (Kress, 1993) (e.g. the primitive parameter  $t$  represents time; the primitive variable  $x$  represents the horizontal coordinate of a position in the Cartesian plane of the MoPiX stage; the non-primitive function  $V_x$  used in equations provided in the library may be interpreted as velocity in the  $x$  direction). MoPiX equations themselves may be considered to be closer to functions as they act in a computational way.

There is scope within MoPiX Version 2.0 for other forms of representation to be incorporated into the equations, for example, natural language or icons.

Equations may be edited or created in the Equation Editor. This treats algebraic expressions as tree structures depicted using containment (nested boxes) rather than nodes and links (Figure 1). When constructing a new equation, it is necessary to plan the overall structure of the tree in advance. Most terms are entered through the keyboard; operators and specialist terms (Arithmetic or Trigonometric function names, relations, logical connectors) are entered via a menu. Incompletely formed equations are displayed in red, turning black once well-formed.

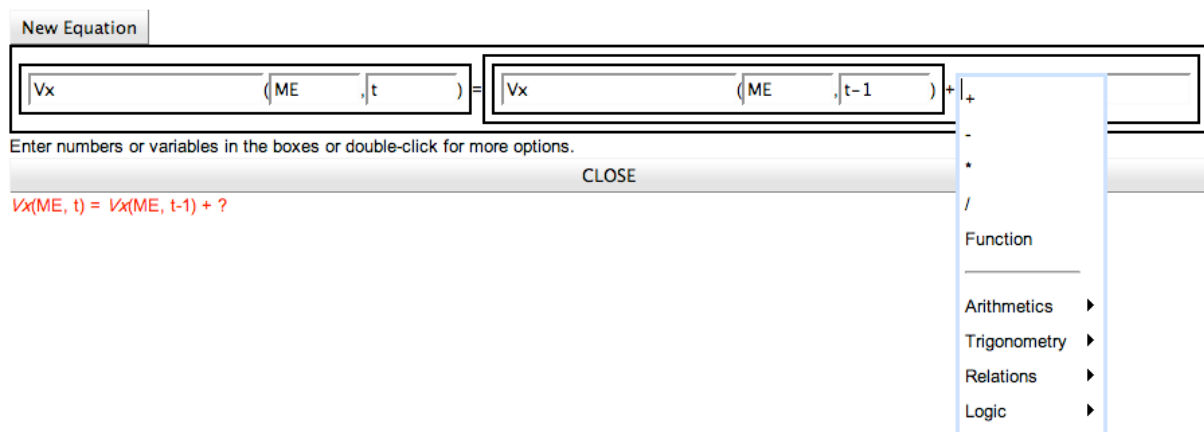


Figure 1: Equation Editor (Version 2.0)

A library of equations is provided. The user directly manipulates these in three ways: (i) by allocating them to physical objects (see below); (ii) by editing equations selected from the library within the editor; (iii) by creating new equations directly within the editor.

The library is structured to group together and label sets of equations that are useful for particular purposes, e.g. horizontal motion; size; colour; multi-object equations. While some of these categories of equations are logical groupings of equations with equivalent functions (e.g. colour equations) others are more pragmatic groupings, providing sets of equations that, when used together, enable users to perform particular kinds of tasks (e.g. horizontal motion equations, graphing equations).





In Version 2.0 the library has been enhanced to provide explanations of the equations. It also has the potential to provide student tasks with embedded equations that can be applied to physical objects and to allow users to extend the library by adding their own equations to it.

The equations provided in the library are directly usable to allocate properties and behaviours to physical objects, but their scope is limited. To take a basic example, the library equation  $V_x(ME, t) = 3$ , interpreted as assigning a value of 3 to the x velocity of a chosen object, is the only one of this form. If a user wishes to assign a different velocity, they need either to edit this equation or to create a new one of a similar form. The equations provided to the user may thus be considered as ‘half-baked’ in the sense that, in order to build any but the simplest of animations, the user must adapt them and/or construct new equations.

#### d) ‘Physical’ objects









      
===== Graphing Equations =====  
 $x_{\text{ME},t} = t$   
 $y_{\text{ME},t} = x_{\text{OTHER},t} + 10 + 400$   
 $y_{\text{ME},t} = y_{\text{OTHER},t} + 10 + 400$





      
===== Graphing Equations =====  
 $x_{\text{ME},t} = t$   
 $y_{\text{ME},t} = x_{\text{OTHER},t} + 10 + 400$   
 $y_{\text{ME},t} = y_{\text{OTHER},t} + 10 + 400$

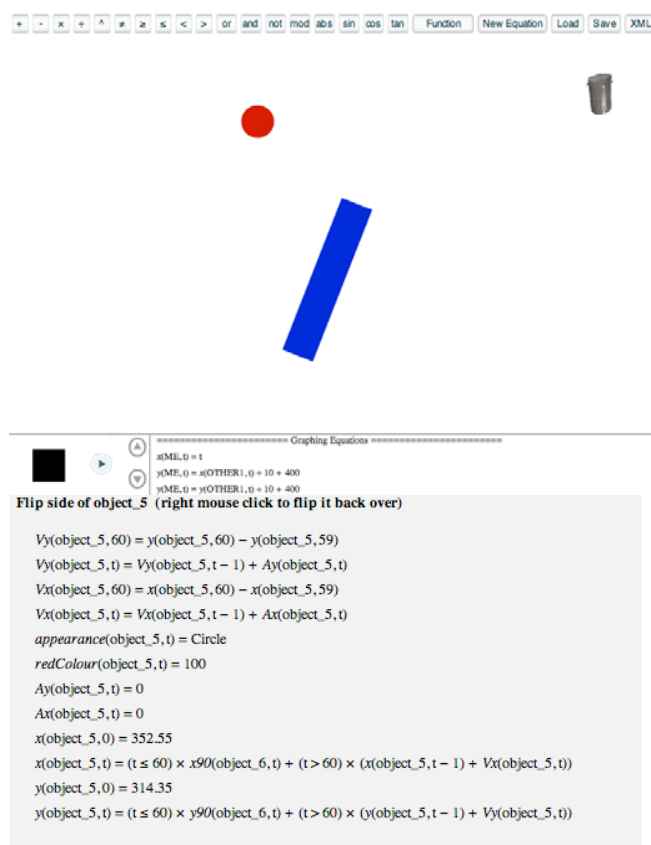


Figure 2: The motion of the red ball is dependent on the blue arm until  $t > 60$

Rectangles and ellipses of various dimensions and colours may be created and manipulated on the MoPiX stage. We refer to these as ‘physical’ objects as they are generally used to represent material components of animated models representing ‘real world’ objects; they are not treated as geometric objects.

When initially created, such physical objects have only an initial default size, shape and colour and a position defined by  $x$  and  $y$  coordinates within the Cartesian plane of the MoPiX stage. They are assigned other properties and behaviours by the equations added by the user. The user may also manipulate them directly by dragging.

Physical objects may be programmed to be dynamic by assigning appropriate sets of equations. Change in position (or change in other properties such as colour) over time is explicit and controllable through the use of equations involving functions of time.

Physical objects may also be programmed in ways that make the behaviour or properties of one object dependent on some property or properties of another object. This enables animations that simulate, for example, the movement of a ball bouncing off a racquet or being spun and then released by a rotating ‘arm’ (Figure 2).

The creation of dynamic models within MoPiX is innovative in relation to the standard curriculum. However, depending on the objectives of the activities offered to students, the concepts encountered can be directly related to curriculum content. Thus, for example, a physical object can be programmed to behave as a projectile, engaging the student in working with concepts of velocity, acceleration and force. (While in some national contexts these concepts are part of the physics curriculum only, in the UK they are also studied within mathematics.)

#### e) Traces of movement of physical objects

By adding an equation that sets the ‘pen’ of a physical object, a trace of the path of the object is created as it moves (Figure 3). Such a trace is not directly manipulable by the user but is the product of the behaviour of the physical object, though its shape can obviously be determined by programming the generating object to move in specific ways.

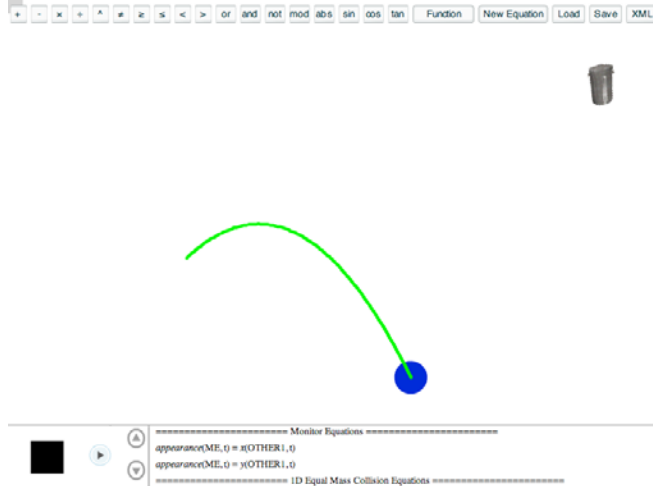


Figure 3: Physical object with a trace of its motion

The trace may be considered as both dynamic and static. It is produced dynamically over time as its generating object moves but it also remains on the screen as a static record of the path after the movement has stopped.

It is possible to conceive of a trace as a graph of the y position of the generating object against its x position. It is also possible to create and programme further physical objects whose movement is related to that of a target physical object, creating traces that represent graphs of other aspects of the target object's movement. For example, the x coordinate value of the graphing object can be set equal to t (time) while the y coordinate is set equal to, say, the vertical velocity of a target object, creating a graph of velocity over time (Figure 4).

In this description we have distinguished ‘graphing objects’ from other physical objects. Within MoPiX terms, there is no distinction between these kinds of objects, although a basic set of equations used to program ‘graphing objects’ with a convenient scale has been provided in the equation library. The conceptual distinction arises only within the context of specific activities in which the trace of one object is defined and perceived as the graph of some aspect of its motion or that of another object. Such graphs are clearly related to the standard curriculum, though the environment does not provide explicit supporting annotations (axes, labels, etc.).

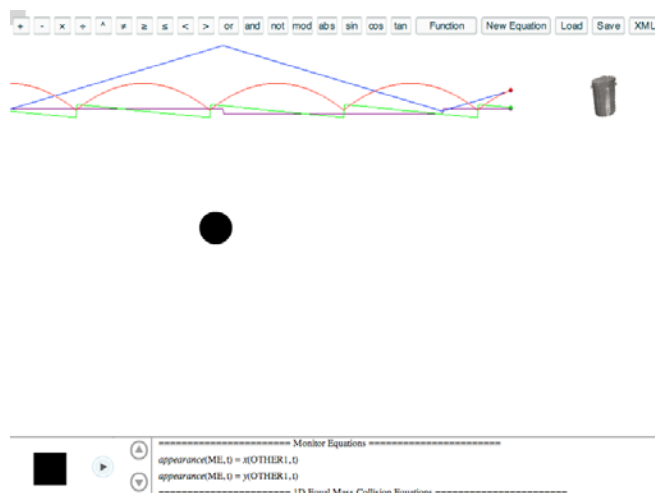


Figure 4: Four graphs traced by 'graphing objects' linked to aspects of the motion of a 'bouncing ball'

## 2. Connections

The values of all equations/functions, and hence all attributes and behaviours of physical objects are dependent on time.

The position and movement of physical objects is defined within the Cartesian plane of the MoPiX stage.

Behaviours and appearance of physical objects are determined by the set of equations applied to the object.

- (i) Equations that set the values of primitive functions may act directly on the physical object. Thus, for example: applying the equation  $\text{length}(\text{ME},t)=200$  to a physical object sets the length of one dimension of the object to 200 units for all values of  $t$ ; applying the equation  $x(\text{ME},t)=x(\text{ME},t-1)+V_x(\text{ME},t)$  sets the value of the  $x$  coordinate of the physical object so that at each successive time point it is augmented by the current value of the function  $V_x$  (a non-primitive function generally taken to represent horizontal velocity).
- (ii) Equations that set the values of non-primitive functions only have an effect on a physical object when combined in a set of equations that serve together to define the value of a primitive function. For example, the equation  $V_x(\text{ME},t)=3$  has no effect if applied by itself. In conjunction with the equation  $x(\text{ME},t)=x(\text{ME},t-1)+V_x(\text{ME},t)$ , however, it augments the  $x$  coordinate of the physical object by 3 at each successive time point.

While the equations all use a consistent semiotic system of formal symbolism, they effect changes in a range of aspects of the physical object, including: colour, size, shape, whether or not it leaves a trace (and the colour and thickness of that trace) as well as its position and the ways in which its position changes.

Equations applied to one physical object can include reference to properties of another so that the behaviour of one object is dependent on the behaviour of others. This allows, for example, moving objects to interact.

Equations may be used to set the 'pen' of a physical object. The sub-set of equations applied to the object to determine its motion will thus also generate a trace or graph of its path. The 'natural' user perspective on this occurrence is to perceive the motion of the physical object as the generator of the trace.

### 3. Activities

MoPiX is conceived as a constructionist toolkit. The constructionist approach to learning (Harel & Papert, 1991; Kafai & Resnick, 1996; Papert, 1980) promotes investigation through the design of microworld environments, i.e. technology-enhanced educational tools and activities, and the observation of learners' actions, developments and communication within these environments. As developed by Strohecker and Slaughter (2000) constructionist toolkits are dynamic visual environments that support building activities in social contexts. Learners build constructs with fundamental elements – in MoPiX these are equations applied to one or more objects - and then activate these constructions as a means of investigating their hypothesis. Working with MoPiX is thus intended to provide students with opportunities to explore and develop their understanding of equations and relationships within a Cartesian plane as well as to investigate the behaviours of the objects they construct. While it is possible to address standard curriculum content through such exploration, the approach is innovative.

The process of connecting equations with the behaviour of physical objects is essentially a modelling activity. Unlike modelling within the usual curriculum, MoPiX modelling includes two levels of model. Like 'usual' modelling, students produce a set of equations that model a real world phenomenon (e.g. a ball bouncing). In MoPiX, however, the animation effected by applying these equations may itself be considered a model of the same phenomenon. By running this animation model and comparing its behaviour to that of the real world phenomenon, the student is provided with more direct feedback about the adequacy of the model than is normally available within the usual curriculum.

### 4. Pedagogical/intervention agenda

1. Innovation vs acceptance - where does the DDA design stand?
2. distance to traditional curriculum - where does the DDA design stand?
3. status of representations traditional, innovative, to be handled, related to others etc
4. scope for mathematization or only about mathematics?

1. The design of MoPiX is innovative in relation to existing curriculum and pedagogy.
2. The distance of MoPiX to the traditional curriculum is dependent on its mode of use and the specific activities offered to students. This is illustrated by the difference between the two teaching experiments carried out in ReMath. In the IOE experiment the pedagogical plan was structured according to topics met in the standard curriculum. Although the nature of the activities with MoPiX were very different from those normally used, the mathematical learning objectives were similar and the students' experience was intended directly to support their study within the standard curriculum. In the ETL experiment, on the other hand, the pedagogical plan was not directly connected to the standard curriculum.

3. The equations, variables, parameters and functions in MoPiX are recognisably connected to representations used within the traditional curriculum, though the use of motivated variable and function names is noticeably different. The way in which users handle these objects in the Equation Editor is very different from within a paper-and-pencil. In particular, the overall structure of an equation has to be created before specific elements are entered into it.

Equations are used to express appearances, behaviours and relationships and, in contrast to traditional use of equations, are not solved within MoPiX (though could be exported to another DDA for solution).

The representation of the Cartesian plane is compatible with standard representations, though lacking visible markers that might help users to recognise its presence.



4. Using MoPiX is primarily about mathematising, e.g. building a mathematical model and animated representation of a physical behaviour such as a bouncing ball.

Harel, G., & Papert, S. (Eds.). (1991). *Constructionism*. Norwood, NJ: Ablex.

Kafai, Y., & Resnick, M. (Eds.). (1996). *Constructionism in Practice: Designing, thinking and learning in a digital world*. Mahwah, NJ: Lawrence Erlbaum Associates.

Kress, G. (1993). Against arbitrariness: the social production of the sign as a foundational issue in critical discourse analysis. *Discourse and Society*, 4(2), 169-191.

Papert, S. (1980). *Mindstorms: Children, Computers, and Powerful Ideas*. New York, NY: Basic Books.

Strohecker, C., & Slaughter, A. (2000). Kits for learning and a kit for kitmaking. *CHI '00 Extended Abstracts on Human Factors in Computing Systems*, 149-150.

## V. MaLT epistemological profile

### 1. Objects

The objects in the MaLT environment are classified in two main categories -Level 1 and Level 2- according to if they constitute primary, intrinsic in the environment, objects or secondary, user-constructed ones. The Level 1 objects are the turtle, the trace it leaves inside the Scene, the three variation tools, the logo commands and procedures, the coordination system and the ready-made, inserted by the user, objects. It was a pedagogical and epistemological decision not to portray the 3D objects (scene, turtle, trace, ready-made objects) as representations of abstract mathematical objects (i.e. the thinnest line possible for trace, an abstract representation of the turtle object). Instead we decided to portray them as simulations of real objects and let the learning situation, social orchestration and tasks provide the context for dissociation from the ‘tangible’ to the abstract notions of entity (turtle), position and segments. Correspondingly, we included a sequence of different scene backgrounds so as to have a choice of cues to making connections with 3d space. Level 2 objects are the Turtle constructions, graphically represented in the 3d Scene in the form of 1d, 2d or 3d geometrical figures. The Level 2 objects are considered to be secondary in the sense that they are the result of the combined use and manipulation of the Level 1 objects. However, this doesn’t mean that, in order to construct a Level 2 object, the user has to employ all of the objects found in the Level 1 category.

#### **The Level 1 Objects**

##### *f) The Turtle*

The Turtle in the MaLT environment is a 3d programmable object resembling in appearance to a real turtle. Through Logo commands and procedures the turtle can be driven around inside the Scene’s 3d space, leaving behind a trace for each one of its logo defined displacements. The state of the turtle (position and heading) can be defined at any time in relation to its difference from the immediately previous state, which directly correlates to the essence of differential geometry. Thus, any change in the turtle’s state is considered to represent a vector in the 3d Scene.

##### *g) The Turtle Trace*

The trace the turtle leaves when moving inside the Scene is represented as a thin 3d cylindrical line. When it is produced by a logo procedure that includes at least one variable (e.g. To rectangle :a :b :c), the turtle trace becomes selectable and transforms from static to dynamic. Clicking on it causes the 1d and the 2d Variation Tools to activate and appear on environment’s interface (the 3d variation tool - VVT - activates through the 1dVT only if the procedure includes at least three variables). Appropriate dragging in each variation tool erases and re-executes the respective procedure with the new value(s) resulting in a DGS-style effect on the constructed figure.

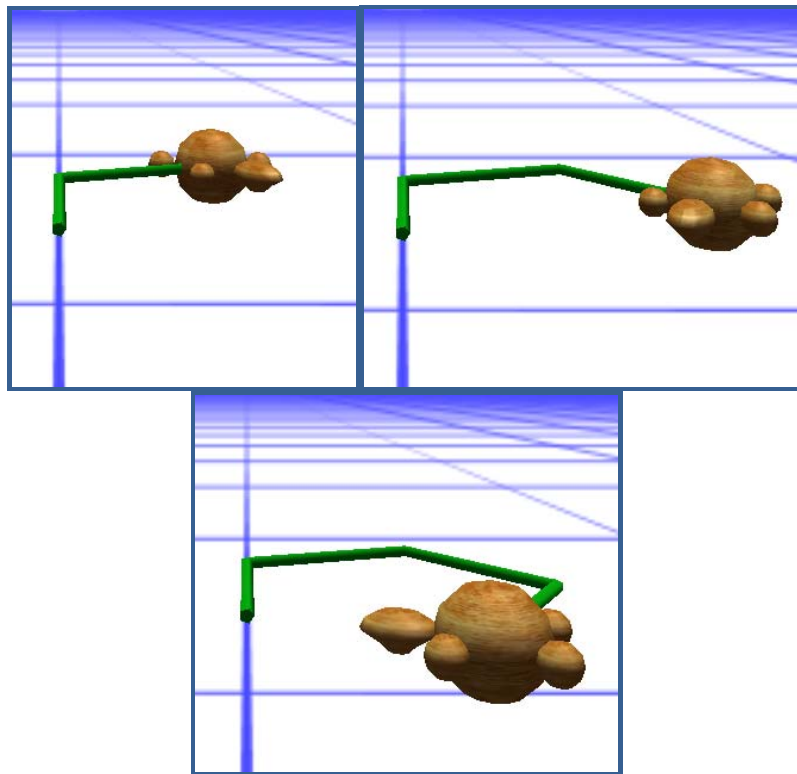


Figure 1: The 3d turtle and the trace it leaves behind in each one of its displacements

#### *h) The Logo Commands and Procedures*

By running Logo procedures and commands at the Logo Editor the user may construct 1d, 2d or 3d figures in the Scene using the Turtle's displacements, create and manipulate stereometric objects (e.g. cylinders, spheres and cones), and control the Scene's camera so as to change the selected viewpoint.

The Logo procedures that control the Turtle's displacement inside the Scene may include variables whose values are determined either at the Editor as the user runs the procedure or dynamically through the manipulation of the Variation Tools. The Logo procedures in MaLT can be used to create new primitives as well as recursive procedures. Apart from Turtle 3d graphics commands, the MaLT Logo also includes commands for Flow Control, Data Structures, Workspace Management and Mathematical Operations.

The syntax of the Logo language is aligned with the syntax of the standard mathematical formalism in the usual way. However, there is a difference to standard Logo syntax (due to the fact that Logo is a compiler so as to gain speed of response to the variation tool manipulations). Inputs to procedures need to be in brackets.

#### *i) The three Variation Tools (1dVT, 2dVT and VVT)*

The MaLT Variation Tools provide users the opportunity to dynamically manipulate:

- I. *the values of the variables in a Logo procedure* (Manipulation level 1).

After typing a variable procedure in the Logo Editor and in order to make the turtle appear and move inside the Scene, the user in MaLT must run the procedure at least once, attributing specific numerical values to each of the procedure's variables. Manipulating the Variation Tools, the user changes the values of the variables, causing the procedure to

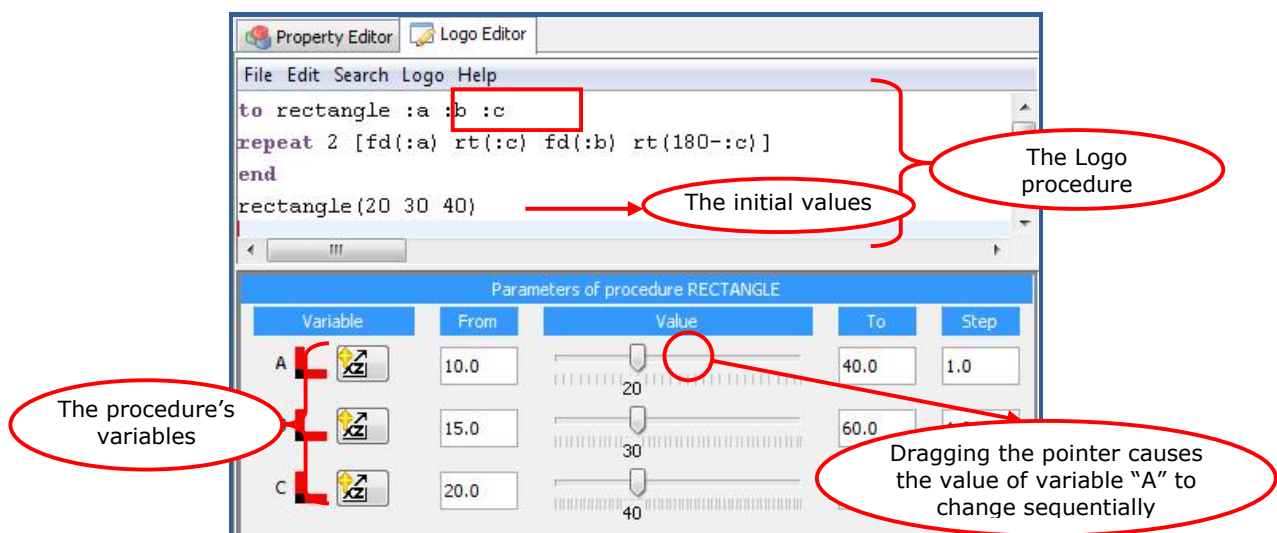
run once again taking into account the new values that appear on the Variation Tool. As the time needed for the MaLT environment to run again the procedure is substantially little, the user may constantly change the values, dynamically manipulating in this way the procedure's variables.

## II. *the figure generated in the screen by the variable procedure* (Manipulation level 2).

A change in the values of the variables through the use of the Variation Tools causes the Logo procedure to be run again taking into account the new values and the geometrical figure in the Scene to regenerate from scratch. The powerful 3d game engine behind MaLT allows the Scene to respond fast and effectively to any dynamic manipulations performed. Thus, the user may manipulate the Variation Tool in a way that the values change in a continuous way, causing the figure to change dynamically.

The three available variation tools are the *Uni-dimensional Variation Tool (1dVT)*, the *Two-dimensional Variation Tool* and the *Vector Variation Tool (VVT)*.

- *The Uni-dimensional Variation Tool (1dVT)*: The 1dVT appears just below the Logo Editor, only after the user runs a Logo procedure that includes at least one variable and clicks on the trace the turtle has left behind after constructing the graphical outcome of the procedure in the 3d Scene. It consists of “number line”-like sliders, each of which corresponds to one of the variables used in the Logo procedure. A pointer on the slider indicates the current value of the corresponding variable. Dragging the pointer the values of the variable change sequentially, causing the figure to change dynamically. Apparently, what is manipulated is not the figure itself, but the value of the Logo procedure's variable. The user may also define the step of the variation as well as the range of the variation.



**Figure 2: The 1d Variation Tool (1dVT)**

- *The Two-dimensional Variation Tool (2dVT)*: The 2dVT allows the co-variation of two variables at the time. Thus, it requires a Logo procedure that includes at least two variables. It is activated through the 1dVT, after selecting two variables and defining which one will be represented in which axis (the X or the Y). The 2dVT is in the form of an orthogonal pad on which the mouse can be freely dragged, leaving behind a trace. Each position on the pad (x,y) corresponds to a specific value for each of the selected variables. The changes in the mouse's position cause respective changes to the 1dVT's sliders' values as well as to the figure in the Scene. The 2dVT is used not just to represent a relationship but mainly to define and implement one.

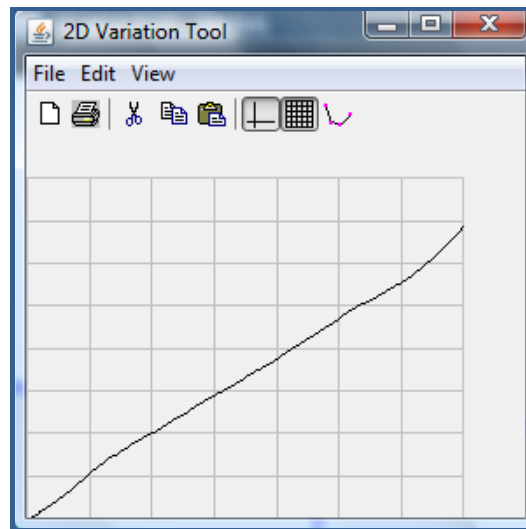


Figure 3: Creating the  $f(x) = x$  functional relationship between the two variables.

- *The Vector Variation Tool (VVT):* The VVT allows the co-variation of three variables by using two 2d representations of a vector defined by these variables according to an  $(r, \varphi, \theta)$  polar semantic in the 3d space. The VVT requires a Logo procedure that includes at least three variables and is activated when clicking on the icon next to the 1dVT. A pop-up menu appears so as for the user to select which variable will correlate to  $r$ ,  $\varphi$  and  $\theta$  ( $r$  stands for length,  $\theta$  for the angle between the vector's projection on the  $xz$  plane and the  $z$ -axis and  $\varphi$  for the angle between the vector and the  $xz$  plane). The three variables' values can be manipulated by dragging and rotating vectors in the VVT window that appears.

In the window appear two vector-like representations (the constituent projections) and a resultant vector representation (1<sup>st</sup> window compartment). Using the first vector-like representation (2<sup>nd</sup> window compartment), the user can dynamically manipulate the vector's length and rotate it to control the value of angle  $\theta$ . Using the second vector-like representation (3<sup>rd</sup> window compartment) the user can manipulate the vector's length and rotate it to control the value of angle  $\varphi$ . The changes performed on the two vector-like representations are reflected on the resultant vector representation, which is not available for direct manipulations.

The VVT displays both polar and Cartesian values. The polar values are represented graphically by the vector itself and numerically in the corresponding text boxes. The Cartesians are represented graphically by the vector's projections on the two axes planes and numerically in the text boxes.

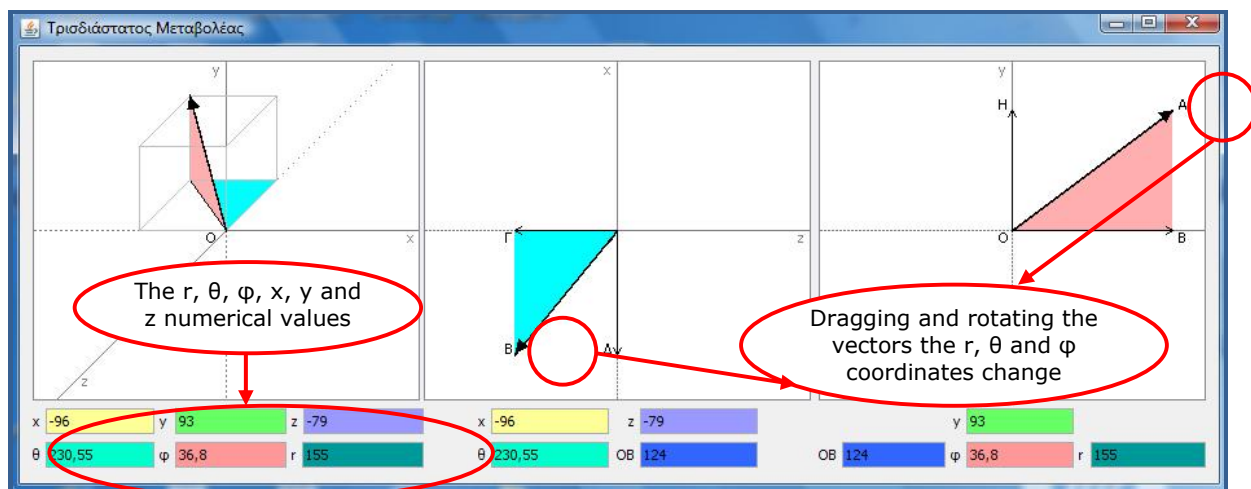


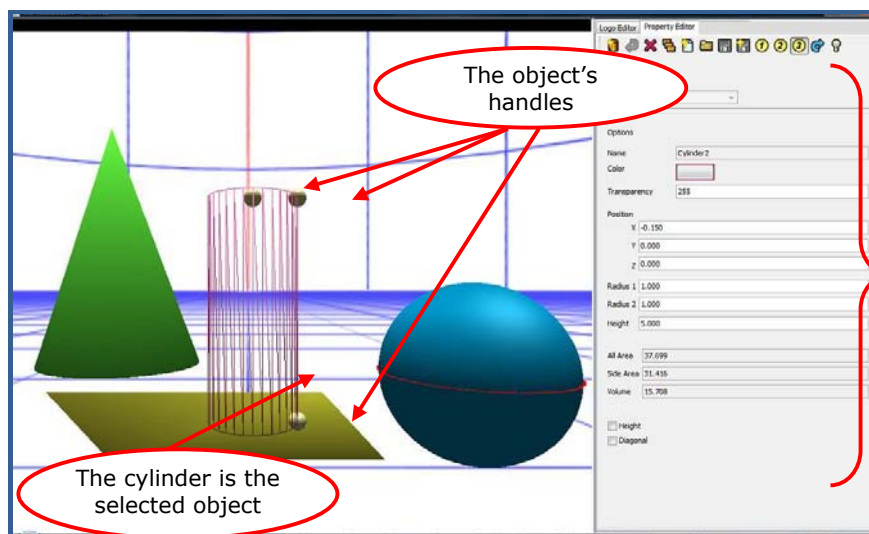
Figure 4: The Vector Variation Tool (VVT)

#### j) The ready-made objects

The Property Editor in MaLT is a component designed to provide the user a library of ready-made objects which she may insert in the Turtle Scene and manipulate them either dynamically or by editing their properties using the Editor's corresponding value fields. The user may select from a variety of ready-made objects, such as: spheres, cylinders, cones, pyramids, cuboids, canonical prismatic, canonical polygons, line segments, circles and planes.

The user before inserting the object may define in the Editor the values of certain of its properties (e.g. the Colour and Transparency). The properties, however, to be defined are not the same for all the types of objects. Other kinds of properties that may appear are the: Radius, Height, Sides, Length, Width, Length of diagonal, Centre, Point and Position (X, Y, Z). Apart from the Property fields whose values are defined by the user, there is also a set of fields whose values are calculated automatically by the environment when the object is inserted in the Turtle Scene. These also vary according to the object's type. The most common of them are the: All Area, Side Area and Volume.

Once inserted in the Scene the ready-made object can be manipulated either dynamically using the handles that appear on it (possibly more than one) or by changing the values in the Property Editor's corresponding fields. The dynamic manipulation causes the values on the Editor's fields to change while the editing of the Editor's values produce an immediate visual result to the object inserted in the Scene.



**Figure 5: Several ready made objects inserted in the Scene**

*k) The XYZ coordination system defining the Scene's 3d space*

The coordination system of MaLT is represented in the 3d Scene through three differently coloured arrows signifying the X, Y and Z axis. Although the arrows themselves are not directly manipulable, the user may observe different aspects of his constructions in the 3d space by dynamically manipulating -using a 3d controller- the position and direction of the 3 available cameras.

<b>1<sup>st</sup> Camera: Floor view (default)</b>	<b>2<sup>nd</sup> Camera: Main view (default)</b>	<b>3<sup>rd</sup> Camera: Side view (default)</b>	<b>A random view using the Floor Camera</b>

**Figure 6: The MaLT coordination System**

The X axis is represented by the red arrow, the Y by the green and the Z by the blue one.

**The Level 2 Objects**

*a) The Turtle constructs*

The Turtle Constructions (e.g. 2d, 3d geometrical figures) visualized in the 3d Scene are the graphical outcomes of Logo commands and/or Logo procedures that the user types and runs at the Logo Editor. The line segments of which the figure consists are the traces the Turtle has left behind in each one of its logo-defined displacements.

The properties of the Turtle constructions (e.g. the height, radius, angle, number of blocks/figures comprising it, position) can be defined in the Logo procedure in the form of varying quantities. For example in the procedure:



```
to parallelogram :a :b :c
repeat 2 [fd(:a) rt(:c) fd(:b) rt(180-:c)]
end
```

the length of the two sides (a and b) as well as the degrees of the four angles (c and 180–c) are represented through the use of variables (:a, :b and :c).

Thus, running at the Logo Editor again and again the same logo procedure attributing each time different numerical values to these variables, the user may generate in the Scene different Turtle Constructions. For example if the user runs the procedure “parallelogram” using the values 5, 5, 90 for the :a, :b and :c variables, a square will appear in the Scene. If she chooses to run the procedure using the values 5, 7, 90, a rectangle will appear and if she chooses to use the values 5, 7, 60 a rhomboid will be generated in the Scene.

Manipulating the Variation Tools, however, the user may dynamically manipulate the Turtle construction appearing in the Scene, attributing values to the variables in a continuous, possibly sequential, way. The Variation Tools allow student to observe the process in which a square turns into a rectangle and then into a parallelogram and not isolated instances of this process.

## 2. Connection

The Logo procedures and commands control the Turtle’s displacements inside a 3d space defined by a XYZ coordination system. Each displacement of the Turtle inside the Scene leaves behind a Trace, generating a 1d, 2d or 3d geometrical figure.

If the Logo procedure includes variables (e.g. To triangle :a :b :c), in order to make the figure appear in the Scene, the user has to run the procedure at least once, using specific numerical values for the variables (e.g. triangle(20,20,60)). Clicking then on the Turtle’s trace (i.e. the figure generated), the Variation Tools appear.

Manipulating either one of the Variation Tools (the 1dVT, the 2dVT or the VVT) the user manipulates the values of the variables, causing each time the procedure to run again taking into account the new values that appear on the corresponding Variation Tool.

In this process, however, apart from manipulating the values of the variables, the user also manipulates the Turtle Construction in the Scene. Each time the procedure is run, the figure is regenerated according to the values appearing on the Variation Tool. Thus, manipulating the Variation Tools the user may manipulate dynamically the Turtle construction.

The geometrical objects visualized in the environment’s Turtle Scene, however, are either *constructed* by the user when running logo procedures and commands or *inserted* by the user after selecting them from a library that offers numerous ready-made objects, such as cylinders, spheres, cones and circles.

These objects, once inserted in the Scene, can be manipulated either dynamically using the handles that appear on it or by changing the values in the Property Editor’s corresponding fields. The dynamic manipulation causes the values on the Editor’s fields to change while any modifications to the Editor’s values produce an immediate visual result to the object inserted in the Scene.

## 3. Activities

MaLT is conceived as a programmable constructionist environment (Papert, 1980, Harel & Papert, 1991) providing multiple linked representations and functionalities to facilitate spatial thinking, 3d visualisation and means for the dynamic manipulation of 3d geometrical objects.

The epistemological validity and the pedagogical design of the software aim to promote an integrated use of both formal mathematical notation inherited from Logo as a programming language and dynamic manipulation of geometrical objects in 3d space. The design of MaLT thus



suggests that 3d geometry is a field where mathematical formalism and graphical representation of objects and relations can be dynamically joined in interesting ways and that joint symbolic and visual control may have important potential for mathematical meaning-making processes.

The main educational activities underlying the design of MaLT concern the development of student's mathematical meanings for 3d geometrical notions when exploring (e.g. constructing, controlling, measuring, transforming) the behaviours of geometrical objects in programmable 3d geometrical constructions.

Moreover, abilities such as spatial orientation and spatial visualisation come into play and are interwoven with the use of various frames of reference (e.g. an egocentric frame of reference related to the Logo commands 'forward/backward, right/left', a coordinate frame of reference, a display frame referring to the orientation and movement in the display screen etc.) and mathematical formalism.

#### 4. Pedagogical/intervention agenda

1. Innovation vs acceptance - where does the DDA design stand?
2. Distance to traditional curriculum - where does the DDA design stand?
3. Status of representations traditional, innovative, to be handled, related to others etc.
4. Scope for mathematization or only about mathematics?

1. The design of Malt is innovative in relation to existing curriculum and pedagogy.
2. The distance of MaLT to the traditional curriculum is dependent on its mode of use and the specific activities offered to students. As an example we will refer to the (familiar) ETL pedagogical plan in ReMath. Although it was referring to a topic met in the standard curriculum (i.e. angle in 3d space), the nature of the activities with MaLT were very different from those normally used in the mathematics textbooks, while the mathematical learning objectives differed substantially. Particularly, the design of tasks aimed at integrating different facets of angle embedded in different physical angle contexts which might challenge real and/or imagined body syntonicity within 3d space in multiple ways (i.e. walking or observe/pilot something flying). Additionally, the notion of angle was not considered a didactic topic in isolation (as it is usually the case in the official discourse of the National Curriculum) but rather as part of a conceptual field involving also the concepts interrelated with it (i.e. the notion of turn), the situation evoked by a given task (e.g. an opening-closing door simulation) and the available representations.

3. The mathematical objects in MaLT are innovative.

- The graphical representation of geometrical objects in TS can be considered as recognisably connected to representations used within the traditional curriculum. However, the study of 3d geometrical notions is not embedded in a systematic curricular activity but appears in a fragmented way in different parts of the mathematics textbooks. So, we consider students' experience with the conventions used to represent geometrical objects in 3d space as rather limited.

- The algebraic-like formalism in the form of programming language is more or less conventional and familiar to students, though the Logo programming environment demands a possibly unfamiliar degree of precision in entering the notation.

- The dynamic manipulation through the variation tool is novel. As an example, dynamic manipulation in 1dVT takes place through dragging on the 'number line'-like sliders. While number lines are commonly used and thus provide a basis of familiarity for making sense of the representation, the dynamic nature of the sliders offers a new way of visualising dynamic variation of variable geometrical objects.

4. Using MaLT is primarily about mathematising. In MaLT, students are able to capitalise upon the power of Logo as a programming language to explore and address mathematical topics such as geometrical properties and relations of 3d geometrical figures. The exploratory nature of the environment is expected to enhance the links between the symbolic representation of geometrical objects (involving the relations between them) and the dynamic manipulation of them using trial and error validated by visual inspection of the outcomes of experimentation. Thus the potential of the visual representations provided by the variation tools is to facilitate new kinds of understanding of the symbolic notation and a learning trajectory from trial and error to gradual mathematisation based on symbolic reasoning.

## VI. Cruislet epistemological profile

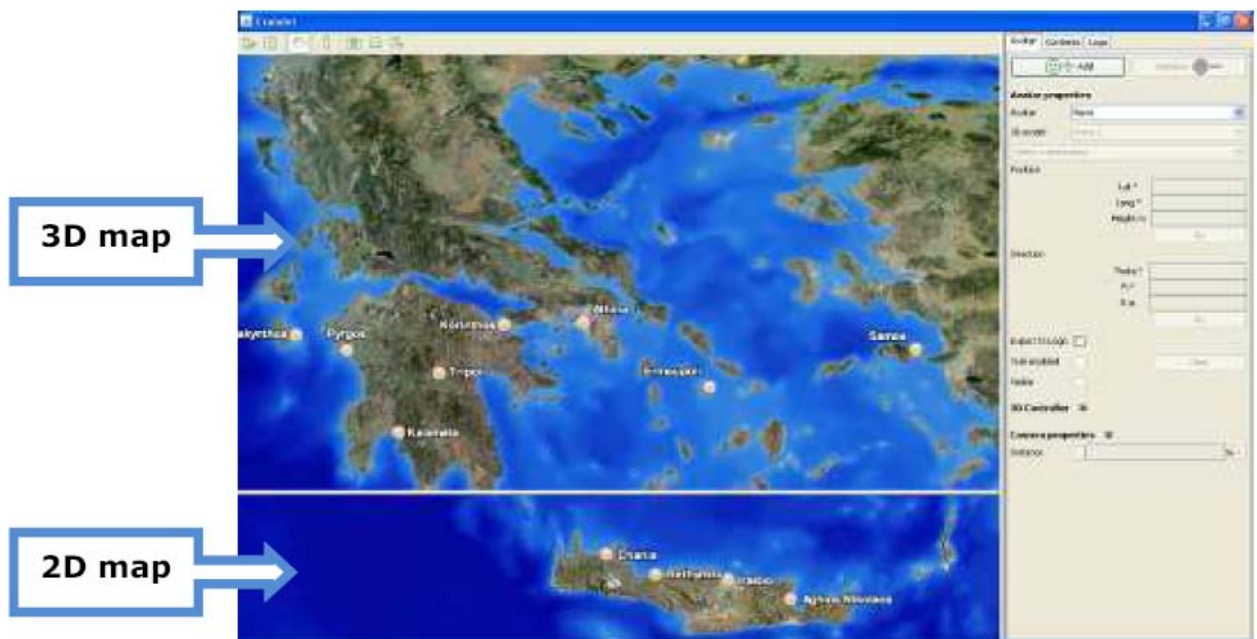
### 1. Objects

The main objects that represented and manipulated with the Cruislet DDA are:

- geographical map,
- avatars
- vectors
- geographical coordinate system and
- spherical coordinate system.

#### The geographical map

The geographical map is a terrain scene where the process of navigation takes place. It contains rich and valid geo-coded information which provides information of the various land features, cities, historical and tourist locations and geographic entities. The terrain scene is split into two map viewers. The one above contains a 3d representation of the geographic map of Greece while the one below contains a 2d map (Figure 1). Before inserting avatars, 3d geographical map can be dynamically manipulated by using the mouse as a navigation tool. In particular, the user moving the mouse across the 3d map can see a moving pointer with the symbol X. This pointer is connected with the box on the bottom left corner of the map. This box displays the description of the location, that is the geographical coordinates (Latitude and Longitude) and the height of the point of the 3d map that user points using the mouse (Figure 2). The user can navigate on the 3d map by clicking with the left mouse button. By clicking the right mouse button, the user affects the actual scale of representation of the area of the terrain and consequently the perspective of the map. Finally, the variation of the scale of the representation of the map is possible by rolling the wheel mouse button. Progressive focusing makes layers of information visible with respect to the proximity to the ground level.



**Figure 1. Cruislet map viewers**



**Figure 2. Pointing a location**

## Avatars

Avatars are central objects of the DDA and their representation is connected with all the represented objects. The user can insert any number of avatars and navigate them upon the 3d geographical map. There is a choice between a number of ways to navigate an avatar, all navigation methods are connected between them and each one to a particular way of doing mathematics. Navigation is possible through the definition of an avatar displacement using either of two geometrical systems:

1. a geographical (lat-long-height) coordinate system by setting the displacement location or
2. a spherical ( $\theta, \phi, r$ ) coordinate system by setting the direction and the length of the vector of displacement.

Defining and executing a displacement is possible either by using a special GUI interface or through Logo programming (Figure 3). The navigation of the avatar can be considered as static, as it is possible though step – by step displacements, though a number of sequential displacements over time could be presented in a rather dynamic way similar to flight simulators. In addition to navigating the avatars, the user can navigate the ‘camera’, i.e. the viewpoint of the map which is always connected to an avatar.

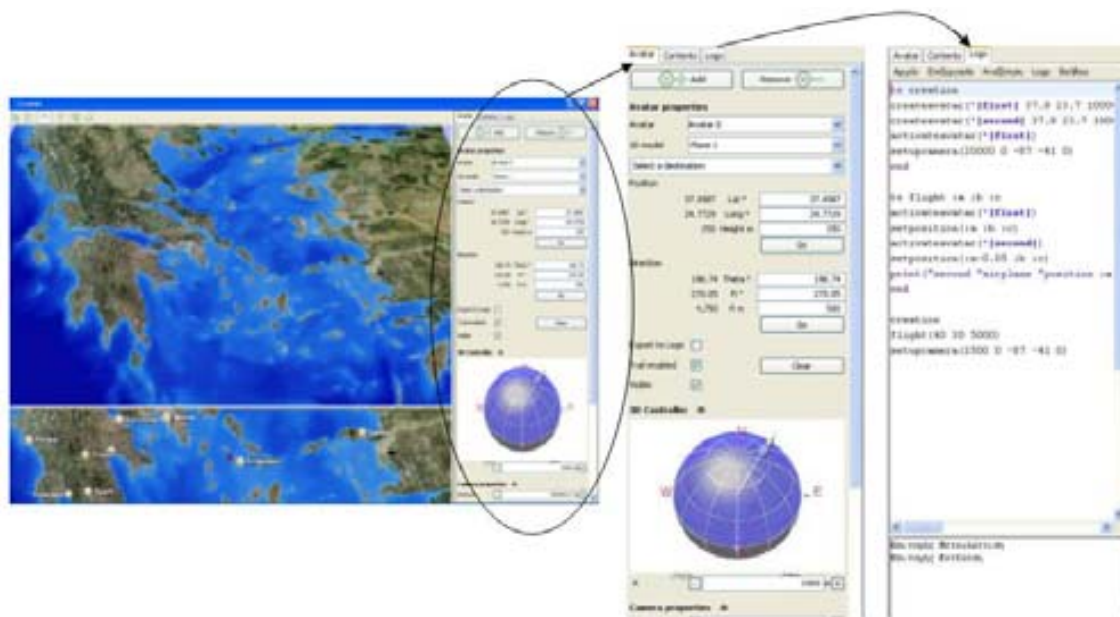


Figure 3: Cruislet environment – Avatar Tab – Logo Tab

## Vectors

The user navigating avatars upon the geographical map actually defines the vector of displacement. The represented vector is connected with the displacement of the avatar acting as a trace. The vectors that are produced by a number of repeated displacements could form geometrical shapes and different geometrical curves. Moreover, the vectors could be added and multiplied by integers in a static way isolated from the behavior of the avatars. In particular, the user viewing the produced vectors as isolated entities could manipulate them by choosing the vectors to be added or by setting the integer with which the selected vector will be multiplied (Figure 4).

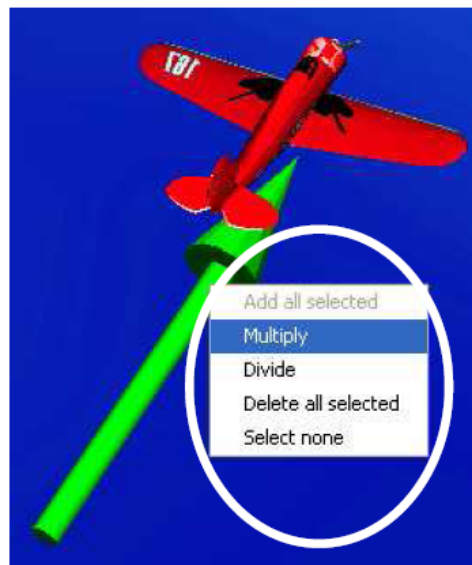


Figure 4: The vector menu

### Geographical coordinates

The implication of the geographical coordinate system is connected to the 3d geographical map and the real geo-coded information and consequently to the navigation process of the avatar. Moreover, the geographical coordinate system is interconnected with the spherical coordinate one as the geographical coordinates of the displacement location are coincided with the end point of the vector. The user can manipulate the geographical coordinates (lat-long-height) of the displacement of the vector either by using the GUI input box or by using Logo programming.

### Spherical coordinates

The spherical coordinate system can be used as an alternative way of defining the vector of displacement of the avatar. The variation of the parameter of the vector of displacement is possible either by using the input box or by dynamically manipulating the 3d controller. The vector representation on the screen is coincided with the defined vector of displacement. The user can also define the parameters of the vector of displacement by using Logo primitives. The spherical coordinate and the geographical coordinate systems are actually two interdependent representational systems as the variation of each one results in the correspondence alteration of the other one (Figure 4).

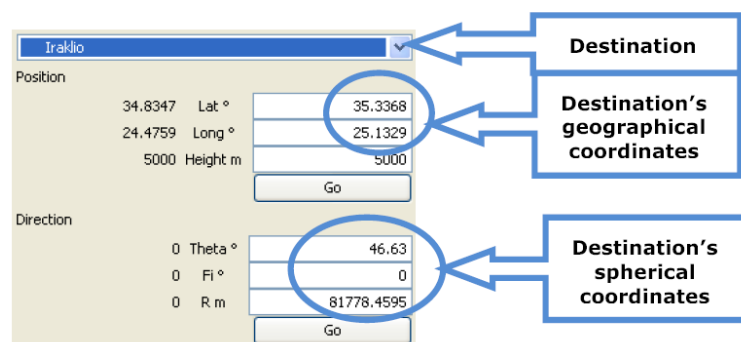


Figure 4: Linkage between the selection of a destination and spherical - geographical coordinates' values



### Logo programming commands

Displacement primitives are special objects within an available Logo language. Commands and parametric procedures can be written and executed containing any sequence and choice of displacements and displacement methods (figure 4). Moreover, through the use of the Logo programming language the software becomes constructionist in the sense that avatar trips can become parametric models of sequences of displacements. Procedures constituting half-baked microworlds can be designed in the sense that students can observe the trips they define and then change the procedures to make changes to the trips. Furthermore, procedures addressing more than one avatar can be defined making it possible to construct functional relationships between sequences of displacements of pairs of avatars. One such half-baked object was the “guess my flight” game where the user tries to guess the functional relationship either between two planes or between the input and the actual displacement of the plane.

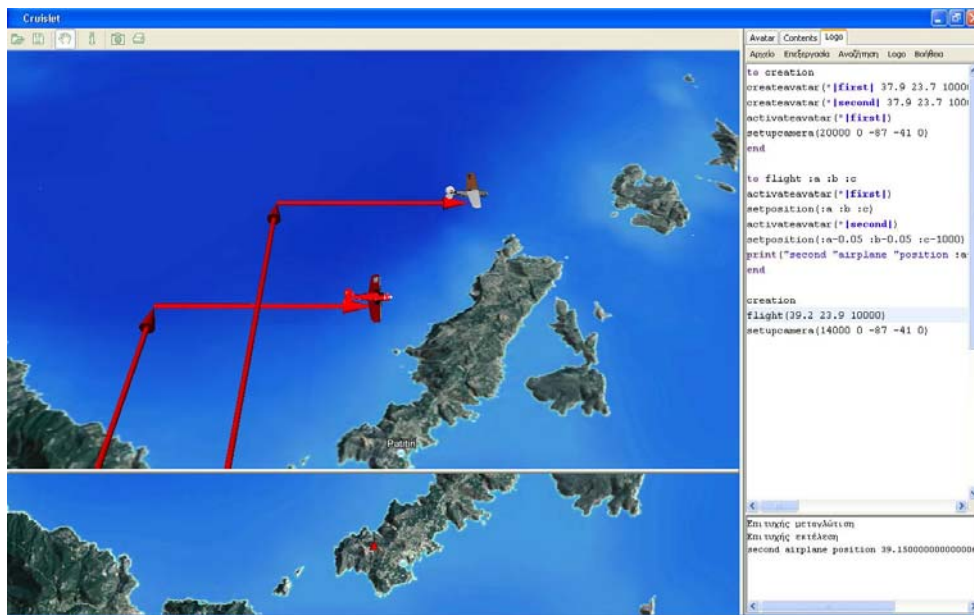


Figure 4: The Cruislet environment, (Logo procedure)

## 2. Concepts

At a first level, the concepts embedded in the software are those of location and orientation in two distinct geometrical systems, the geographical (Cartesian) and the spherical (polar). Points in space can be defined through any of these two systems interchangeably. The idea is that they co-exist so the user can develop methods of how to choose between the two and in which circumstance related to the task at hand. Systematic sequences of points can also be defined either in a stepwise way through individual displacements or by means of functional definitions through Logo procedures. All geometrical and algebraic concepts deriving from this mathematical kernel are available for mathematization from the students such as geometrical shapes and curves, vector operations and functions. The background of Geography is considered as an application area for mathematics just as kinematics in MoPiX. The particular properties of Geographical space and geographical information provide a context for new ways of using mathematics and new ways of making them meaningful.

### 3. Activities

The Crusilet DDA is distant both from the ways in which geometry, analytic geometry and functions are structured in the curriculum and from the software usually used in mathematics education. It was designed to provide a medium with which to question the curriculum structure and to stress the need for applied meaningful mathematics as a medium for mathematical meaning making. Consequently, it aims to innovative exploratory activities concerning the mathematics of positioning, orientation and functional relationships represented as curves in the context of 3d Geographical space. The design of the activities supported by Crusilet involves the engagement of students in the process of navigation avatars in any way by making choices between vectorial and Cartesian displacement controllers and the construction of avatar trips. A lot depends on the activities designed with the use of Crusilet and the context in which it is inserted. It is possible to make distinct connections with the standard curriculum and organize special activities for students to make these connections with school mathematics. It is also however possible to design more exploratory meaning making activities where students gradually mathematize the ways in which they construct avatar trips.

### 4. Pedagogical/intervention agenda

In designing the Crusilet DDA and the respective tasks we gave ourselves some distance from the traditional structure of the mathematics curriculum and looked for learnable mathematics concerning position and orientation in geographical three dimensional (3D) space. In particular, we adopted the approach of students' gradual mathematization within game-like activities in problem situations that are experientially relevant to students. Our intention was to involve students in activities through which they would use symbols, make and verify hypotheses in order to solve a particular real problem in a rich learning environment. Our agenda can be described at two levels, a more global level of steering the mathematics education community towards re-considering what mathematics is rich for meaning-making and a more classroom oriented level of how students can see meaning in existing curriculum mathematics.



## VII. Application of the DDA profile framework to NetLogo

To further assess the profile framework, I applied it to software to which it had not been applied before, the NetLogo agent-based modeling environment.

I'm completing this epistemological profile of NetLogo with emphasis on a) connections to the mathematics of systems that change over time, and b) more general notions of modeling in the form of agent-based code and the visual and quantitative/graphical representations of that executed code.

Here is the NetLogo profile:

### 1. Objects

There are four main “computational objects”, or components from which NetLogo models are constructed: agents, the properties owned by agents, the behaviors executed by agents over time, and the ‘world’ or global properties.

### Representations

#### Agents

Agents are dynamic objects that represent components (such as atoms, animals or individuals) of a system. Agents are akin to the ‘physical objects’ of the MoPiX system, except that it is assumed that many duplicates of a given agent type will exist in a model. Agents are represented as *computational objects* in the computer code based representation of the model, and *visual objects* in the visualization of the model as it is executed.

#### Features:

**Dynamic** / Static: agents change over time

Mathematical / **Computational**: agent behavior and properties/associated quantities are determined computationally

Direct Manipulation / **Menu Driven: (and Code-Driven)**: agents are primarily controlled via computer code, but can be also accessed via a visualization and properties inspected/manipulated via a menu and dialog box

Compatible / **Innovative**: Though there are considerable beneficial uses of NetLogo for mathematics education, it is not easily compatible with most US mathematics curriculum. It is more easily integrated with standard science curriculum.

#### Agent Properties

Agents have a number of default properties such as shape, color, location, and so forth, and users can create any number of additional parameters for agents. Agent properties are primarily represented in the *computer code* based representation of a model, though they are also often represented as a *visual property of each agent*.

#### Features:

**Dynamic** / Static: agent properties change as a model is executed

Mathematical / **Computational**: properties are updated as a result of the execution of computer code, or by hand by the user

Direct Manipulation / **Menu Driven: (and Code Driven)**

Compatible / **Innovative**: not easily compatible with most US mathematics curriculum.

### Agent Behaviors

Agent properties are changed by executing agent behaviors over time: for example, agents can move forward each “tick” to update their position, change color per “tick”, or engage in more complex behaviors (such as ‘eating’ one another, adjusting properties based on the properties of agents spatially near them, and so on). Behaviors are represented primarily in *computer code*, but are often evident in the *visual behavior* of the model as it executes.

#### Features:

Dynamic / **Static**: once behavior code is defined, they remain the same over the period of execution of the model

Mathematical / **Computational**.

Direct Manipulation / Menu Driven: n/a

Compatible / **Innovative**: not easily compatible with most US mathematics curriculum.

### World Properties

World properties are values representing sums, means, and other aggregations of agent properties that model some important aspect of the system of interest. For example, in a model of population growth, individual agents reproduce and their reproduction is driven by ‘local’ circumstances that depend on spatially near agents, but users are ultimately interested in the overall population, a world-level property. World properties are primarily represented via *plots over time*, but can also often be *inspected visually* (for example, the number of agents in a system can be counted in the visual representation) and are included as global variables or reporters in the *computer code*.

#### Features:

**Dynamic** / Static: quantities change as

Mathematical / **Computational**.

Direct Manipulation / Menu Driven: n/a

**Compatible** / Innovative: Although computational code is typically not used in mathematics classrooms, the global quantities of interest in NetLogo models are frequently represented *as time series plots*, a familiar and compatible object for mathematics and science classrooms.

## 2. Concepts

There are three main “mathematical concepts” that are embedded in the notion of a NetLogo model: the *system state*, a “tick” or delta of time representing a *computational derivative* (that is, quantitative changes in the system for a ‘tick’ or single sliver of time), and a *computational integral* (that is, quantitative patterns in the system over several ‘ticks’ or slivers of time). These concepts are not strictly mathematical – indeed, they are primarily computational, but in NetLogo’s role as an alternative means to model systems over time, may nicely fit with mathematical models of systems based in calculus and differential equations.

### System State

At any given ‘tick’ or instantiation of a model, agents and the model world possess properties that model some aspect of the system they represent. These values can be considered solutions to some problem: for example, what a population can be expected to be given certain assumptions

after  $n$  iterations of time, or what potential outcomes one might expect for a gas trapped inside of a container and heated.

Because system states are obtained computationally and often depend on stochastic and local interactions, they should be considered specific *instantiations* of possible solutions, rather than complete or absolute solutions.

### Computational Derivative

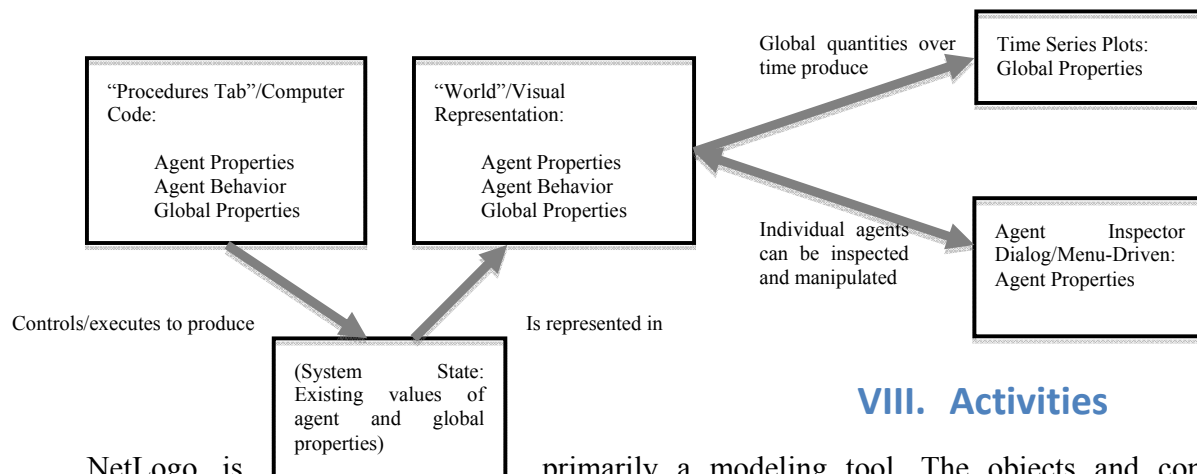
Because agent behaviors ultimately define how agent-based and global properties unfold, and because behaviors are defined in terms of how those agents and that world will behave in the context of a single execution or ‘tick’, one can think of a single execution of a model’s rules and the resulting quantitative changes in quantities of interest as a ‘computational derivative’ at a given point in the model’s execution time.

### Computational Integral

As agent behaviors are executed over several ticks, trends over time in quantities of interest can be tracked, plotted, and otherwise computed. This can be measured as the accumulation of, or trends in, quantities over time or the ‘computational integral’ after a given interval in the model’s execution time.

## 3. Connections

All of the objects described above are primarily driven by computer code, though many are alternatively accessible via menu-based manipulation to access dialog boxes that reveal the properties of individual agents:



## VIII. Activities

NetLogo is primarily a modeling tool. The objects and connections between objects support this task by encouraging the user to consider the elements that comprise systems and the ways that those elements behave, in order to systematize and represent those systems computationally. After executing that code, quantitative/mathematical trends of the system can be examined via plots and other means of collecting quantitative information from the system model.

NetLogo is most frequently used in courses that *utilize* mathematical modeling for specific applied purposes, such as science classes. A large library of preexisting models and collections of models addressing a number of scientific and mathematical topics already exist, and can be given to students in the context of existing curriculum for students to use for exploration activities.

These models can also be manipulated easily, moving the task from primarily exploration-based to modeling-based.

NetLogo also enables users to construct agent-based models from ‘scratch’, although activities such as these are more difficult to incorporate easily into typical mathematics and science classrooms.

#### **4. Pedagogical/Intervention Agenda**

There are three levels at which NetLogo can be incorporated into classrooms or other educational settings: first, existing models can be presented to students for exploration and interaction. Second, existing models can be given to students as “seed” models to be explored, but then manipulated to model similar systems or systems with varying behaviors or parameters. Finally, NetLogo can be used as an open-ended programming environment – primarily for the modeling of multi-agent systems, but indeed for any reasonable computational task.

NetLogo is particularly well suited for mathematization tasks: students can observe specific instantiations of mathematical trends and attempt to mathematize those trends, for example, or try to match their computational models to mathematical models of scientific or social phenomena.