First Version of the Dynamic Digital Artrefacts

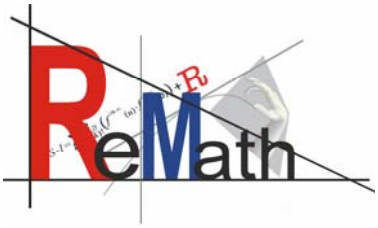
Jean-François Nicaud, Christophe Viudez

Information Society Technologies (IST)

Programme

Project Number: IST4-26751

ReMath
Representing Mathematics with Digital Media



First Version of the Dynamic Digital Artrefacts

Deliverable Number:	D4
Due date of deliverable:	30.11.2006 (month 12)…….
Actual submission date:	30.11.2006……..
Start Date of Project:	01.12.2005
Duration:	36 months
Version:	1
Work Package:	WP2 “Development of Dynamic Digital Artefacts”
Lead Partner:	LEIBNIZ-UJF
Contributing Partners:	CTI, DIDIREM, CNR-ITD, NKUA-ETL, TALENT, IOE-LKL
Authors:	Jean-François Nicaud, Christophe Viudez
Contact:	Christophe Viudez (Christophe.Viudez@imag.fr)

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)		
Dissemination Level		
PU	Public	√
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Project Coordinator:

Research Academic Computer Technology Institute (CTI)
eLearning Sector
"D. Maritsas" Building, N.Kazantzaki str.,
University Campus, GR 265 00, Rio, PATRAS

Project Coordinator	Administration Manager
Chronis Kynigos	Farmaki Maria
kynigos@cti.gr	farmaki@cti.gr

Partners:

1. Université PARIS 7 DENIS DIDEROT, 2 Place Jussieu, 72251, Paris, France,
2. Université JOSEPH FOURIER GRENOBLE 1, Avenue Centrale 621, Domaine Universitaire, 38041, Grenoble, France,
3. Consiglio Nazionale Delle Richerche, Piazzale Aldo Moro 7, 00185, Roma, Italy,
4. Università degli Studi di Siena, Banchi di Sotto 55 , 53100, Siena, Iitaly,
5. National & Kapodistrian University of Athens, Educational Technology Lab, Faculty of Philosophy, Pedagogy and Psychology Secretariat,School of Philosophy (3rd floor), Panepistimioupoli, 157 84 Ilissia, Athens, Greece
6. Talent Anonimos Etaireia Pleroforikis, Plateia Karitsi 4, 10561,
7. Institute of Education, University of London, 20 Bedford Way, WC1HOAL, London, United Kingdom

TABLE OF CONTENTS:

Introduction

This document is the fourth deliverable of the ReMath project and deals with the first version of the Dynamic Digital Artefacts.

The document is divided in two parts; the first one is a description of the work progress for each DDA and the second part is an updated specification of each DDA.

DDA 1: Casyopee

Authors: **JM Gélis, JB Lagrange, Gaye Banfield**

UNIVERSITE PARIS 7 Denis Diderot, DIDIREM Equipe de recherche en didactique des mathématiques

Work progress in 2006

In summary, the work planned for the first year of the ReMath project was fundamentally:

1. to ensure quality display of mathematical expression and multilingualism,
2. to integrate script and graphical design with algebra and calculus by recording the actions and workings during a session, which may be edited or updated
3. to help students to link geometrical and enactive situations with algebraic representations through the implementation of geometry capabilities with the addition of new objects such as curves, straight lines, tangents, segments, circles and free points on a straight line or a plane.

The work completed in this first year of ReMath partially satisfies this aim. Specifically the work carried out includes:

1. multilingualism by externalisation of the interface strings.
2. the use of a Latex module (DLL) for the display of mathematical expression, and of an XML parser to provide a Note Pad able to display mathematical expressions and graphs. Students are able to edit this NotePad by switching to a source mode. They can also print and record it. Export in XML format has still to be done.
3. the design and implementation of a dynamic geometry module with, at the present time, only straight lines, points (specific and free) and the manipulation of those objects (erase and reappearance). We are presently working on linking this module to Casyopée in two ways. One way is from Casyopée to the module, by sending graphs that can be manipulated inside the module, and considered by the module to be Casyopée objects and treated like parameters. The other way is from the module to Casyopée by implementing a new window “measures”. In this window, students are able to build “geometrical calculations” involving distances related to points in the module and to create “geometrical functions” by taking a geometrical calculation as the dependent variable, and a length, or an abscissa related to a free point, as the independent variable

This work will be achieved at the end of the first reporting period. We expect to produce a prototype, but probably not ‘secure’ enough for experimentation by other teams.

Immediate future work will be to produce this ‘secure’ prototype.

1. ensuring total consistency between the module and Casyopée
2. developing the module with new objects
3. ensuring multilingualism

We hope to be able to reach most of these objectives in the first two months of the next report period.

Further work will include: additional enhancements to the interface, facilities for saving, exporting and printing, and a replay module to make sense of the teacher’s log file.

The state of and the objective of DDA 1: an extension of Casyopée at Year 1 End of the ReMaths project

1 Introduction

Casyopée is designed as an advanced "calculator of functions" that can help students to solve algebraically, problems typical at upper secondary level. The educational goal is to provide a tool that provides students with a different management of algebra and calculus by an easier articulation of syntax and semantics. Beside exploration, it is recognised that building and writing a proof is also an important objective.

In addition, the team aims to develop a rapport with teachers, establishing with them at which point of the curriculum Casyopée could be helpful, and developing functionalities so that this objective can be achieved. Acceptance, both at an institutional level and by the teachers, is therefore important. Before deciding to extend Casyopée in the ReMath project, the choice of the institutional context was specifically the French upper secondary education system, focusing on tasks relative to properties of functions, signs, variations and so forth.

The main objective of the extension to Casyopée is to develop a new dimension in students' activity in algebra and calculus, by adding dynamic geometry capabilities in order to facilitate the modelling of geometrical dependencies into algebraic functions. Other objectives are: to ensure a quality display of mathematical expressions and multilingualism, and to provide a means for making sense of the log file produced by the interaction of the student with Casyopée. After achieving these objectives, the Casyopée environment should be easier to use, especially by others teams of the ReMath project. It should address a wider range of problems, allowing easier insertion into other institutional contexts and curricula.

Specifically the objectives and their degree of achievement can be listed as follows:

1. ***To ensure quality display of mathematical expression and graphics and multilingualism.***
   This objective has been nearly completed. The extension to Casyopée makes use of a LaTex module (DLL) for the display of mathematical expression, and of an XML parser to provide a NotePad able to display mathematical expressions and graphs. Students are able to edit this NotePad by switching to a source mode. They can also print and record it.
2. ***To insure multilingualism***
   Multilingualism is achieved by externalisation of the texts of messages, the names of actions and by the parameterisation.
3. ***To provide a means for making sense of the log file produced by the interaction of the student with Casyopée***
   This remains outstanding.
4. ***To help students to link geometrical and enactive situations with algebraic representations through the implementation of geometry capabilities with the addition of new objects such as curves, straight lines, tangents, segments, circles and free points on a straight line or a plane. Thus allowing students to work with enactive representations of 'real world' phenomenon and to introduce more interactivity into the work on algebraic and non-algebraic representations and in the changes of representations (modelling, interpreting, converting…).***

This is achieved by linking a geometric module to Casyopée in two ways. The first is from Casyopée to the module, by sending graphs that can be manipulated inside the module, and considered by the module to be Casyopée objects and treated like parameters. The second is from the module to Casyopée by implementing a new window. In this window, students are able to build "geometrical calculations" involving distances related to points in the module and to create "geometrical functions" by taking a geometrical calculation as the dependent variable, and a length, or an abscissa related to a free point, as the independent variable.
This objective is completed to a large extent. The geometric module has been designed and most of its capabilities have been implemented. Objects from Casyopée like parameters and curves can be used in this module. The new "geometrical calculations" window allows a student to build calculations involving all symbolic Casyopée objects as well as lengths involving points of the geometric module. Students can also choose a variable (length or abscissa) involving a free point to create a "geometrical function". The dynamic link between the geometric module and the "geometrical calculations" window remains to be done. Up to now, areas cannot be involved in geometrical calculations: to express an area a student has to use a combination of lengths.

The next section provides a detailed description of the current functionalities of this extension.

2 The Domain of DDA1: An extension to Casyopée

The domain of the extension is the algebraic modelisation of dependency between a "geometric calculation" and a measure. These terms will be defined below.

2.1 The Geometric Module

In this first year of the project, our goal was not to develop a whole geometric dynamic environment, but to see how geometric and calculus facilities can articulate each other. The geometric module has therefore a limited number of objects and functionalities as shown in Figure 1:

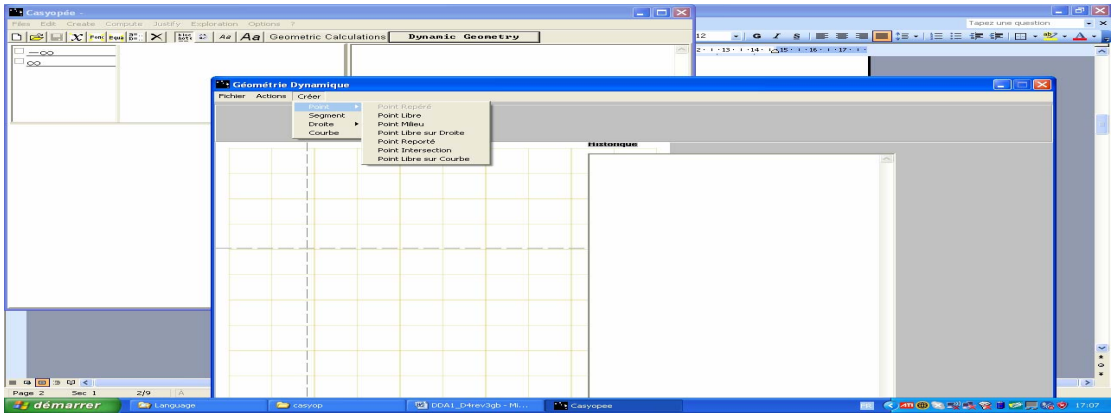


Figure 1 geometric objects available in the Dynamic Geometric Module

To summarize the above, the types of geometric objects that can be created and used in the extension to Casyopée are:

- Points
  Different kind of points can be created:
  - precise point
  - free point
  - frees point on an object (line, curve or segment)
  - mid-point
  - intersection point
- Curves
- Lines (a line defined by 2 points, parallel, perpendicular)
- Segments

At this stage only a basic management of the objects is available. Objects can be animated, hidden or suppressed. Management of the axis is not possible. Names cannot be chosen. Facilities for enabling editing and saving of the objects remain outstanding.
The curves are created by using functions defined in Casyopée's main window. Parameters can be used and the curves are dynamically updated when the user animates the parameters in the main window, as shown in Figure 2:

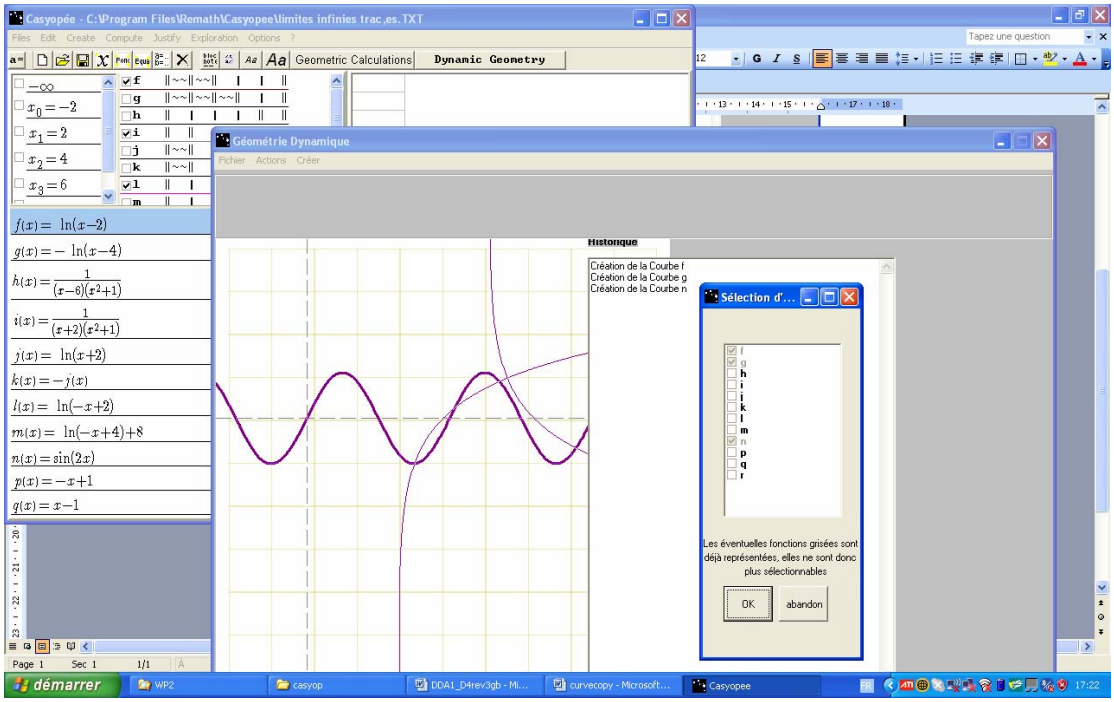


Figure 2 creating curves in the Dynamic Geometric module.

Lines can easily be created for the following cases, as shown in Figure 3:

- line defined by 2 points
- line defined by a point and its slope

- line defined by its equation
- line perpendicular to another
- line parallel with another

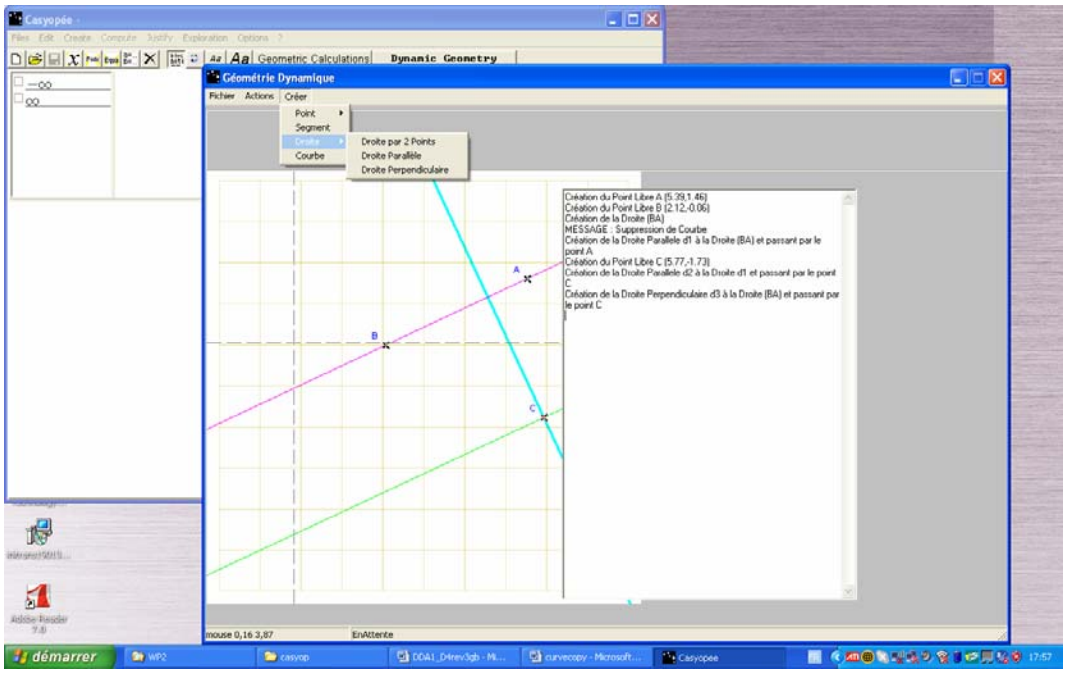


Figure 3 Creating Lines in the Dynamic Geometric Window

A segment can be created by 2 points that have been previously defined, see Figure 4 below.

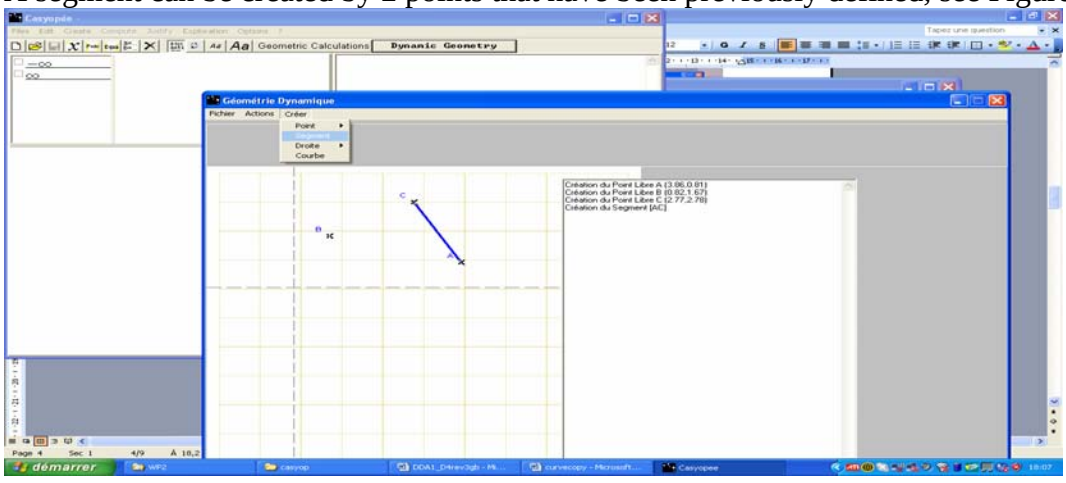


Figure 4 A segment created in the Dynamic Geometric Module from two previously defined points

2.2 The “geometrical calculations” window

The *Geometrical Calculation* window is invoked with a button in order to create a geometrical calculation, as in Figure 5.

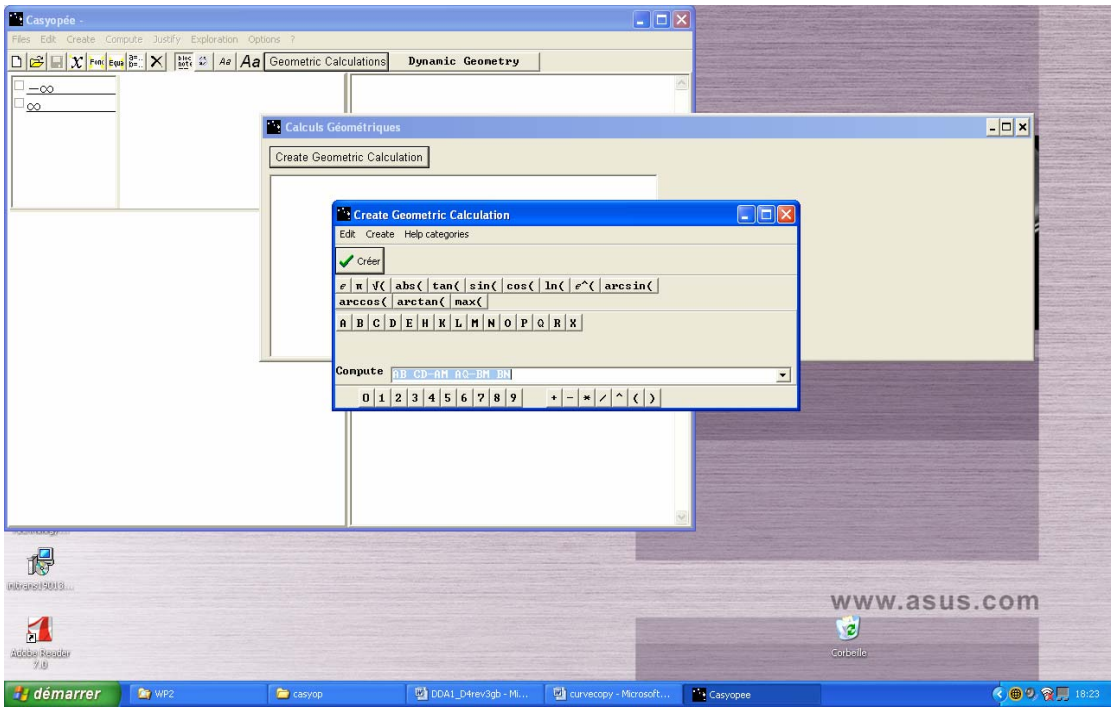


Figure 5 Invoking the *Geometrical Calculations* window

After validation, the new calculation appears in the list. One can create one or more calculations and then exit. When a calculation is selected, a panel appears on the right, as in Figure 6.

Since, there is no dynamic link established between this window and the geometrical module, provisionally, there are vertical cursors to "simulate" the displacements of free points (shown here as M and R).

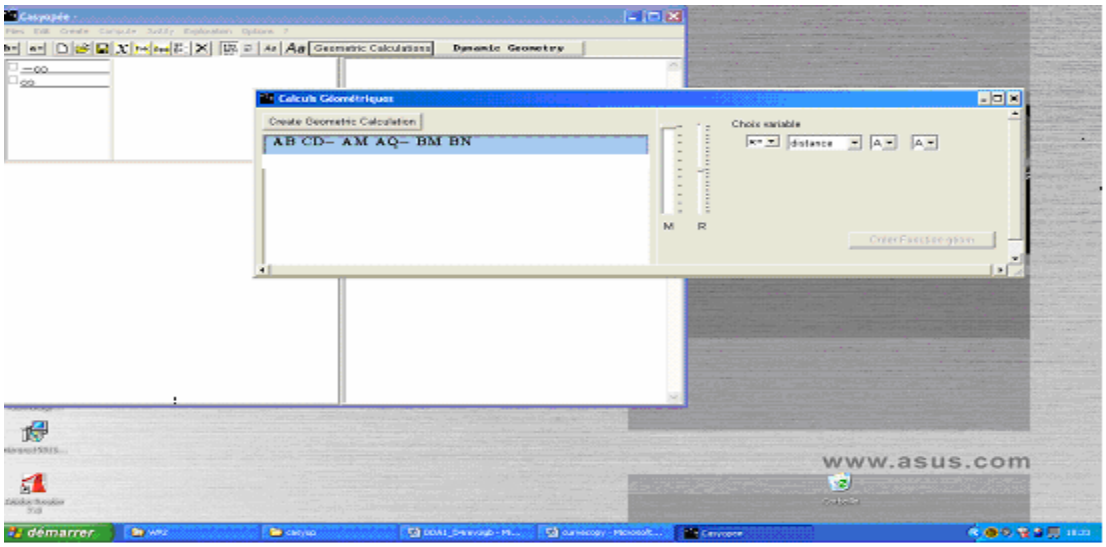


Figure 6 Creating Geometrical Calculations

The value of the calculation for the current position of the points is displayed under this calculation. On right-hand side, one can choose a variable using the 3 Comboboxs (identifier, function, points). A diagnosis is displayed below as well as the value for the current position of the points. If the diagnosis is positive, the button "create geometrical function " is activated as in the following example.

After activating the "Create geometrical function" button, a new function is created. Currently, it is not possible to choose its identifier and there is no trace of it in the Notepad. If the student is operating a calculation ("calculate" menu), the result is given algebraically, as in following example.

2.3 An Example highlighting the functioning of the "geometrical calculations" window – Optimizing an area

The Geometrical Calculation window will be the interface between the dynamic Geometric Module (DGM) and the algebraic module (Casyopée itself). The goal here is to explore the dependence between a geometrical calculation and a geometrical variable. A geometrical calculation is a well formed expression utilizing any Casyopée object that has previously been defined. With reference to Figure 7 assume that the distances between points are defined in the DGM. The distances are identified simply by two points. For example, if A, B and M are defined in the DGM, *a* and *b* are Casyopée parameters and *f* a Casyopée function, *a* sin(AB BM)+ *b* f(AM) is a geometrical calculations.

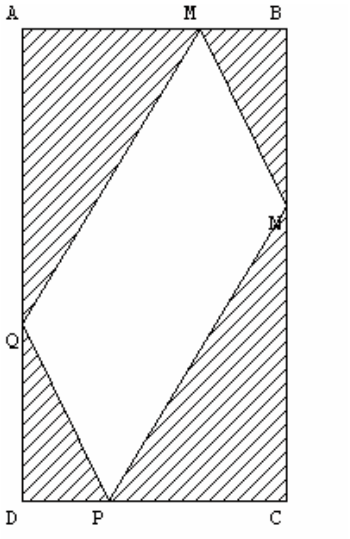


Figure 7. ABCD is a rectangle. MNPQ is a parallelogram constructed within the rectangle ABCD where M is a free point on [AB] and N, P, Q are free points respectively on [BC][CD][DA] such as AM=BN=CP=DQ.

This is an example of use: It originates from an Example of a Scenario - optimising an area, taken from the scenarios of the Deliverable 3 (p.31-33)

The problem:

ABCD is a rectangle, M is a point on [A,B], N on [B,C], P on [C,D] and Q on [D,A] such as AM = BN = CO = DP. How does the area of the quadrilateral MNPQ vary when M is moving on [AB] ? Is there a minimal value?

Figure 7 is the figure of the problem.

The significant stages of work are:

Stage 1: Definition of parameters

Parameters *a* and *b* are first specified in Casyopée see Figure 8. From the Create menu, choose *Parameter,* then *New* so that the parameter ***a*** appears as in Figure 8. It is possible to change the maximum and minimum values, increment and delete the parameter.
For this example it is important that $a < b$

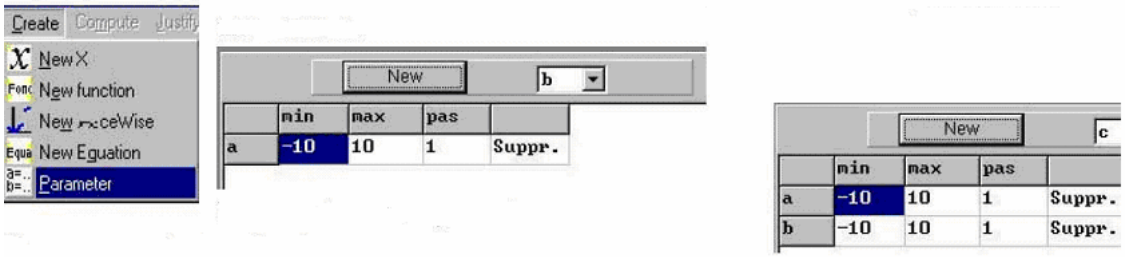


Figure 8 Creating parameters in Casyopée

Stage 2: Construction of the geometric object

After the link between the DGM and Geometric Calculation window is established, the figure ABCD and MNPQ will be created in the DGM as in Figure 7 with O defined as a coordinated point, A is a free point in the plane, B, C, D are constructed from A. The parallelogram MNPQ within ABCD will be defined by M, a free point on [AB], N, P, Q are constructed from M and R is a free point on [AC]

Stage 3: Work on f(x) the area of MNPQ

The *Geometrical Calculation* window is invoked with a button in order to create a geometrical calculation, as in Figure 5. After validation, the new measurement appears in the list. One can create one or more measurements and then exit. When a measurement is selected, a panel appears on the right, as in Figure 6. The two vertical cursors "simulate" the displacements, respectively, of the points M and R (this will form the connection to the DGM). The value of the measurement for the current position of the points is displayed under measurement. On right-hand side, one can choose a variable using the 3 Comboboxs (identifier, function, points). A diagnosis is displayed below as well as the value for the current position of the points.

If the diagnosis is positive, the button "To create geometrical function " is activated as in Figure 9.

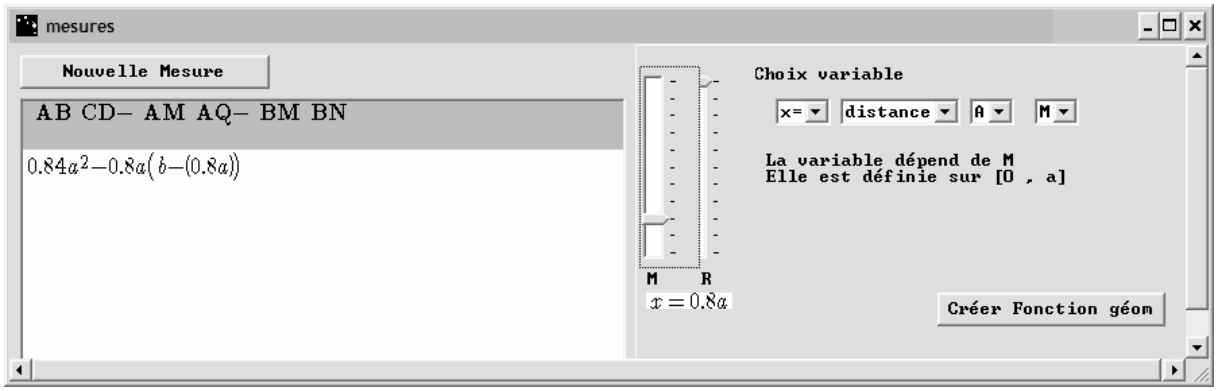


Figure 9 Create Geometrical Functions

Figure 10 shows the “geometric calculation¨ window after instantiations of the parameters *a* and *b*. The approximate values are displayed, which allows the exploration of the dependence between measurement and the variable while moving the point M.

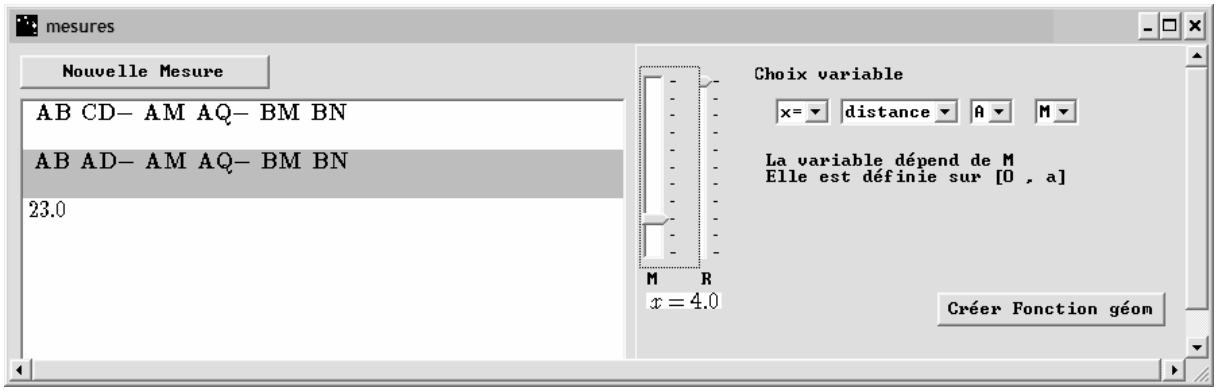


Figure 10 Create Geometrical Function after instantiation of the parameters *a* and *b*

After activating the "To create geometrical function” a new function is created. Currently, it is not possible to choose its identifier and there is no trace in the NotPad. If one is operating in algebraic calculus, the result is given algebraically, as in Figure 11.

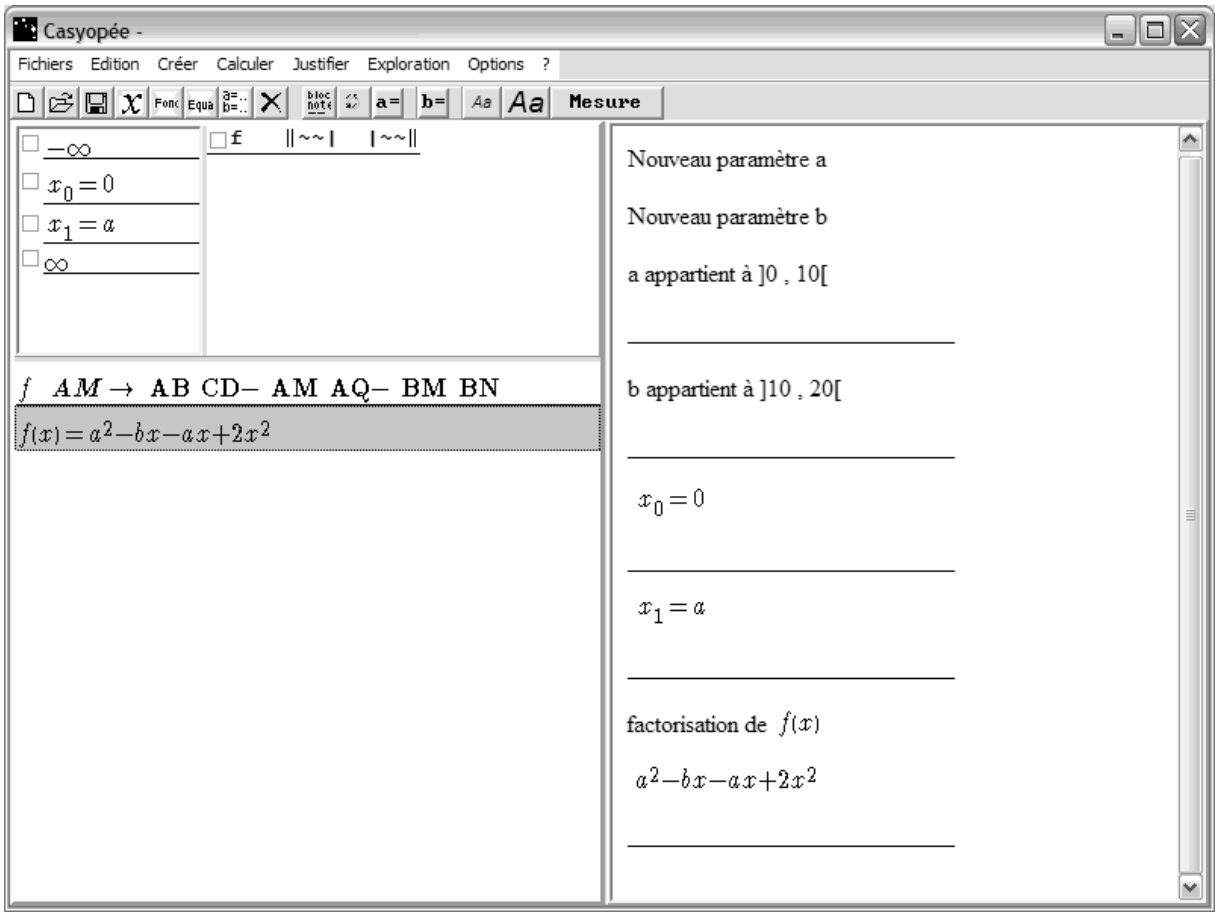


Figure 11 The Geometrical Function is created as shown above

After this, students will be able to solve the problem algebraically and to write a solution using Casyopée as indicated in the Scenario (Del 3).

DDA 2: Aplusix

Authors: J-F. Nicaud, C. Viudez, D. Bouhineau, N. André

UNIVERSITE JOSEPH FOURIER, Laboratoire Leibniz INPG-UJF-CNRS

Work progress in 2006

During the first year, the Leibniz team worked at the development of two new modules of Aplusix: the tree representation and the graphical representation. During the first 6 months, the specifications of these modules have been produced. During the last 6 months, the implementation of the tree representation has been realised for the main functionalities.

Situation of the implementation of the tree representation.
The internal representation, external representation (in text files) and representation at the interface of trees have been implemented.
Three modes have been defined and implemented for editing trees:

- The “Entire Tree Without Scaffolding” in which the tree is entirely expanded and any text (correct or incorrect operator or operand) can be put in the nodes;
- The “Entire Tree With Scaffolding” in which the tree is entirely expanded and only correct operators with a correct arity can be put in the internal nodes;
- The “Mix Representation With Scaffolding” in which the tree is entirely, or partially, or not expanded and only correct operators with a correct arity can be put in the internal nodes; usual representations are put in the leaves;

Two new exercises types have been defined: “Represent an expression (given in the natural representation) as a tree (with the 3 modes)” and “Represent a given tree in the natural representation”. A new module of the teacher editor allowing the building of these new types of exercises is been developed.

Currently, the editing of the trees in the three modes are implemented for insert, modify and delete; they are partially debugged. The comparison of the student’s production with the reference expression is ongoing. The new module of the teacher editor is ongoing.
At the end of month 12, the debugging of the editing functionalities will be well advanced but not achieved; the implementation of the comparison and the new module of the teacher editor will be completed. A prototype will be available in the middle of month 13.
This prototype will contains all the necessary functionalities for the experiments that have been envisaged by the 3 experimenting teams.

During months 13 to 15, the higher editing functionalities (select, cut, copy, past, drag&drop) will be implemented.
During months 13 to 18, the graphical representation will be implemented.

Aplusix Updated Specifications

1. General framework

1.1. The human framework

Algebra contains objects, like numbers, polynomials, functions, and representations of these objects. As for other domains, representations are concrete elements used for different purposes, the two main ones being: (1) communication between agents (on agent builds the representation of an object or a concept, another agent observes and interprets this representation); (2) reasoning (inferences are applied to representations producing new representations). Representations are built within a representation system, which is a language having a grammar. A representation is produced on a medium. The main media we will consider when human agents are concerned are paper, screen and sound. Paper and screen belong to the same family, the visual family; they differ by the tools allowing to produce the representations; in the case of a screen, a piece of software is in charge of the drawing and can have very different sorts of commands and constraints.

In this document, we consider paper representations as references in the sense that they exist for hundred of years (with some evolution) and screen representations as goals. Paper representations need to be observed either for being replicated on screen or for being compared with new representations.

Algebra has a representation system which is quasi universal, which elements are algebraic expressions. The medium of this representation system is paper where elementary symbols are drawn in a two-dimension area, for example:

$$\frac{x^2 - x\sqrt{2} + 3}{x + \frac{1}{3}}$$

Small algebraic expressions can also be represented with the natural language like:

A sum of two terms, the first being the product of 2 and x and the second the quotient of 3 and 5.

This representation concerns the same algebraic expression than the following one:

$$2x + \frac{3}{5}$$

This existence of several ways to express an algebraic expression shows that algebraic expressions are not a concrete representation system. The two concrete representation systems we have used above are:

- The two-dimension drawing of algebraic expressions that we will call the ***natural representation*** of algebraic expressions (because it has been built in the history of mathematics as natural language in the history of humankind);
- The representation in natural language.

However, we will continue to consider algebraic expressions as representation of objects; when this will be useful, algebraic expressions will appear as abstract representation of objects with several possible concrete representations; otherwise, we will confuse the two notions.

Mathematical objects (functions, equations, systems of equations, etc.) can also be represented in other registers than algebraic expressions, particularly consider graphs.

Besides these representations used for the communication between humans, humans have their own representation systems. As a consequence, our human framework has five components.

The five components of the human framework are:

1) Objects
2) Algebraic expressions
3) Concrete representations of algebraic expressions on screen and paper
4) Graphical representations of objects on screen and paper
5) Mental representations of humans.

1.2. The human & computer framework

Let us now introduce computers in the framework. Computers have their own representation systems that are called ***internal representation*** and that are defined and implemented by software designers and programmers. Computers are able to produce concrete representations of algebraic expressions and graphical representations of objects on screen and paper. They have strong difficulties to interpret these representations. For example, if an algebraic expression has been written on paper using the natural representation, e.g.:

$$\frac{x^2 - x\sqrt{2} + 3}{x + \frac{1}{3}}$$

it is possible to scan this representation, producing an internal representation based on colored pixels, but it is very difficult to build a computer representation of the corresponding algebraic expression in a manageable way for the computer, where numbers, variables, operators and the structure have been understood.

Actually, the usual communication between humans and computers is very different for the communication between humans. When two humans communicate, one produces a representation on a medium, the other interprets the representation; both actions, production and interpretation, are independent. Usually, when a human communicates with a computer, he/she uses commands (choices in menus, keystroke) and the representation is built on the screen by the computer. In fact, the computer first builds an internal representation corresponding to the commands, after that, the computer displays the result on the screen. So, the computer has not to produce an internal representation from an external one. Such way of communication is very dependent of the set commands which can be more or less appropriate to the user goals and more or less close to natural ways of building representations.

The media for the communication between computers differ with the human ones. The mains ones are files and networks. Particular representation systems are used for the communication

on these media, we will call them file representations and consider mainly file representations based on texts (successions of characters), including XML representations.

The seven components of the human & computer framework are:

1) Objects
2) Algebraic expressions
3) Concrete representations of algebraic expressions on screen and paper
4) Graphical representations of objects on screen and paper
5) File representations of computers
6) Mental representations of humans
7) Internal representation of computers.

1.3. Taking into account the student

The fact that a category of users is made of students has an important implication at the level of the representation and communication.

Incorrect representations, like ill-formed representations (that do not respect the grammar), and representations that do not correctly correspond to the context (like incorrect calculations), have to be considered in the computer internal representation system and the file representation system.

In the communication between the student and the computer, when the student builds a representation using the computer commands, the nature of the commands and the scaffolding have a strong importance. Commands can be inadequate, leading difficulties for learning the right concepts; with important scaffolding, the student will not be able to produce some error and will not learn from these errors.

1.4. Algebraic expressions and representations of algebraic expressions

1.4.1. Definition of well-formed algebraic expression

We consider a set of atomic expressions: natural numbers in decimal representation, e.g., “12”, decimal numbers in decimal representation, e.g., “12.53”, logical constants, e.g., “true”, and variables, e.g. “x”. An atomic expression has a type (numerical for “12” and “12.53”, Boolean for “true”, the type of a variable depends of its domain).
We consider a set of operators: Minus, Plus, Subtract, Times, Divide, Power, SquareRoot, Equal, And, Or, etc. Each operator has an arity: Minus and SquareRoot are unary (1 argument); Subtract, Divide, Power and Equal are binary (2 arguments); Plus, Times, And, Or are variadic (2 arguments or more). Each operator has type constraints, e.g., Equal applies to numerical argument (numerical includes here variables over a numerical domain). Each operator has type of result, e.g., Boolean for Equal.

Definition
A well-formed algebraic expression is an atomic expression or an operator applied to well-formed algebraic expressions with respect of the arity and of the type.

1.4.2. Representations of well-formed algebraic expression

We can represent an algebraic expression using intermediate letters as in the following example:

Plus of A, B, C
Where
A is Times 2, x, y
B is Minus x
C is Power x, 2

The media supporting this representation are paper, screen, sound, and text files.

With a functional notation, we can represent an algebraic expression without intermediate letters, as in the following example:

Plus(Times(2, x, y), Minus(x), Power(x, 2))

The media supporting this representation are paper, screen, and text files.

We can represent an algebraic expression with a tree[1] having operators as internal nodes and arguments of these operators as children, as in the following example (Figure 1):

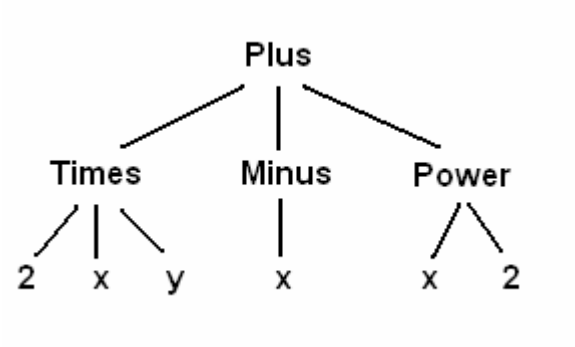


Figure 1. Representing an algebraic expression with a tree

Of course, usual symbols can be used instead of their names.
The media supporting this representation are paper, screen, and image files.

A universal representation of algebraic expression has been established. We call it the "natural representation" (as for natural language). This representation uses two dimensions. An example is:

$$2xy - x + x^2$$

The media supporting this representation are paper, screen, and image files.

[1] A tree is composed of nodes and links. Each node has a mother except one (the root of the tree). Nodes having no children are leaves. Nodes having children are internal nodes.

Remarks
The three first representations are close to the definition: there is no difficulty to locate the arguments of an operator.
The link between the natural representation and the definition is complex. In the natural representation, there are unrepresented operators (times over 2, x, y; Plus before “Minus x”; Power before x and 2), priorities between operators and sometimes the need of parentheses.

2. Domain and goal of the new modules of Aplusix

2.1. The tree representation

Algebraic expressions are introduced at school through the natural representation. No definition is given to students. Sometime, representations in natural language are used during a short period.

Considering that:
1) The natural representation of algebraic expressions is complex and not close to the definition,
2) The tree representation of algebraic expressions is close to the definition,

we believe that the learning of the tree representation and the understanding of the mapping between the natural representation and the tree representation will help students to understand the algebraic expressions.

The tree representation module of Aplusix will have the following main functionalities:
1) Display of tree representations on screen
2) Construction of tree representations by the student
3) Display of the correspondence between tree representations and natural representations.

Three modes are implemented for editing the tree representations, they are defined by two parameters:
- The representation is a full tree representation or the representation combines tree and natural representations,
- The edition has some scaffolding or not.

1 The Free Tree Representation mode (FreeTree):
In this mode, the student can build any sort of ill-formed representation and trees are fully expanded.

2 The Controlled Tree Representation mode (CtrlTree):
In this mode, the computer compels the student to write correct operators in the internal nodes of the tree and helps him/her to respect the arity of the operators; the tree is fully expanded.

3 Controlled Mixed Representation mode (CtrlMixed)
This mode provides the same scaffolding as the previous one but allows to expand or collapse freely the nodes

Note that there is no free mixed representation as the mixed representation needs correct operators and correct arities because it contains natural representation.

Domain

The domain is the current domain of the algebraic expressions of Aplusix. For well-formed expressions, the domain is composed of:

1) Numbers that can be built with digits, decimal separator and operators Plus, Minus, Times, Divide, Power, SquareRoot (they represent a sub-set of the real numbers)
2) Rational functions that can be built with the above numbers and with variables
3) Polynomial equations and inequations of one unknown and degree less or equal 4 that can be built with the above numbers,
4) Rational equations and inequations that can be transformed in the above polynomial equations and inequations,
5) Systems of linear equation with a maximum of 10 equations and 10 unknowns.

Ill-formed expressions in the FreeTree mode have no limitation.

Ill-formed expressions in the CtrlTree mode are limited to missing arguments of operators.

Ill-formed expressions in the CtrlMixed mode are limited to missing arguments of operators and unbalanced parentheses.

Problems

Aplusix contains currently two families of problems:
First, exercises of the following types:

- Numerical calculation
- Expansion
- Factorization
- Equations
- Inequations
- Systems of linear equations.

Second, problems with a text and an expected answer that are provided by the teacher.

The use of tree representations is possible for these problems.

Two new sorts of exercise are added:

1) Asking the student to build the tree representation, in the FreeTree or the CtrlTree mode, of an expression given in the natural representation
2) Asking the student to build the natural representation of an expression given in the tree representation

2.2. The graphical representation

The necessity of combining symbolic and graphical representations in mathematics and in the learning of mathematics is well known. This justifies the introduction of graphical representations in the Aplusix system.

The graphical representation module of Aplusix will have the following main functionalities:

1) Display of graphical representations on screen
2) Display of the correspondence between graphical representations and algebraic expressions.

Note that there is no direct construction of graphical representations by the student.

The correspondence between graphical representations and algebraic expressions will particularly favour the understanding of the link between correct calculations and identical denotation (the denotation being the mathematical object associated to the expression, like a function) by showing through the graphical representations that the objects are the same.

Domain

The domain is a subset of the above domain with the following limitations:

1) Rational functions of one variables
2) No additional limitations for equations and inequations
3) Systems of linear equation with a maximum of 2 unknowns.

Problems

Problems are identical to the above ones.

3. Specifications of the tree representation system

3.1. General point of view on tree representations, global insertion in the Aplusix system

3.1.1. Principles for editing representations

General principles of person-machine interaction include:

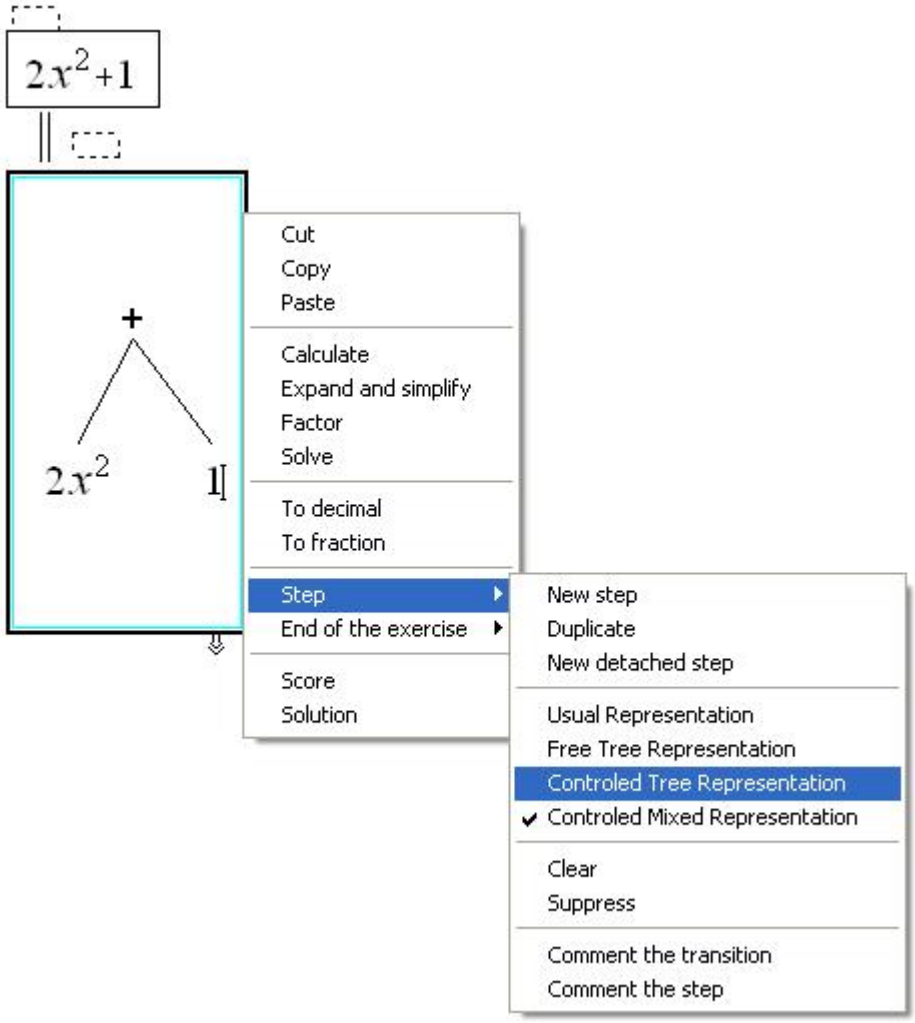
1) When an object is edited, there is generally an insertion point or a selection; there are never both.
2) When a set of objects is edited, there is one and only one edited object at any time.
3) A simple click on an edited object places the cursor (insertion point).
4) A double click on an edited object selects a meaningful small entity (e.g., word).

Equation editors have the following principle:

5) The input of an operator over a selection applies to the selection.

3.1.2. Choice of the different modes in Aplusix

In Aplusix, calculations are divided into steps represented as rectangles. The information about the representation and the mode is located in the steps. For a given step the user can choose with a pop up menu the representation and mode. Aplusix will perform some checking actions before changing the representation.



When a teacher builds problems with the editor of Aplusix, the author can set the representation mode of each section. In this case, the student will be compelled to work with this mode.

3.1.3. Mixed-Reps

Mixed-Representations (or Mixed-Reps) are representations of algebraic expressions mixing tree representations (or Tree-Reps) and natural representations (or 2D representations, or 2D-Reps). See example Figure 2.

Two buttons are available to expand and collapse the tree, like in applications managing trees (e.g., Windows explorer): each node of the tree has a ⊞ button when it can be expanded, a ⊟ button when it can be collapsed, and nothing when is can be neither expanded nor collapsed. These buttons only appear when the mouse is near the node.
These commands change the representation at the interface but do not change the expression. A command will be added to the popup menu in order to get a fully expanded representation from a selected node.

A fully-expanded representation is a tree in which internal nodes are operators and leaves are atomic expressions (integer numbers, decimal numbers, etc., see below). An unexpanded representation is a 2D-rep. A true semi-expanded representation is a tree whose internal nodes are operators and leaves are usual algebraic representations. Note that leaves always contain 2D-reps (of atomic or non atomic expressions).

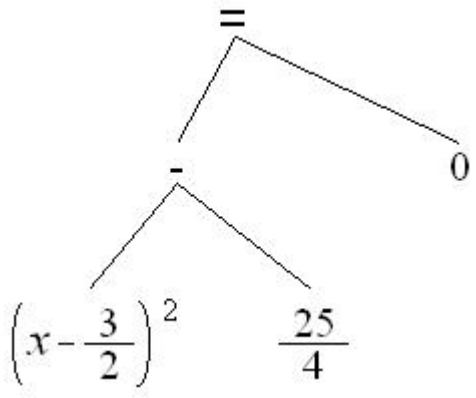


Figure 2. A Mixed-Rep of $\left(x-\frac{3}{2}\right)^2-\frac{25}{4}=0$.

3.1.4. Insertion of Mixed-Reps in Aplusix

Mixed-Reps take place in the steps displayed by Aplusix. In the usual functioning, expressions have the 2D-rep. An item in the popup menu allows shifting to the Mixed-Rep. At this moment, a ⊞ button appears on the left of the representation.
The ⊞ button will have a memory of which nodes have been previously expanded. Like the Windows Explorer, it can store expanded nodes to restitute them and so keeping a previous exploration.
For example:

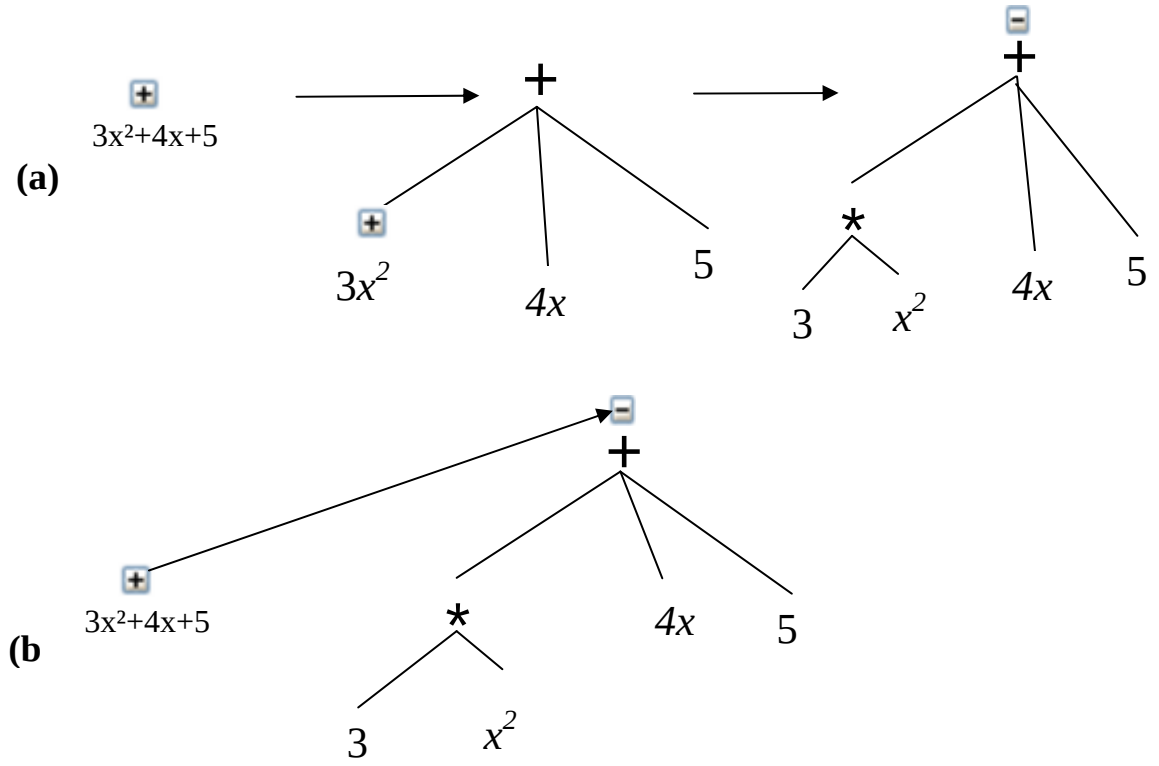


Figure 3. (a) First, the user clicks on the root ⊞ button, then he/she clicks on the ⊞ button over "$3x^2$" , and finally on the ⊟ button over the "+" operator to collapse entirely the tree;
(b) After that, the user performs a click on the root ⊞ button : the tree expands as it was previously expanded.

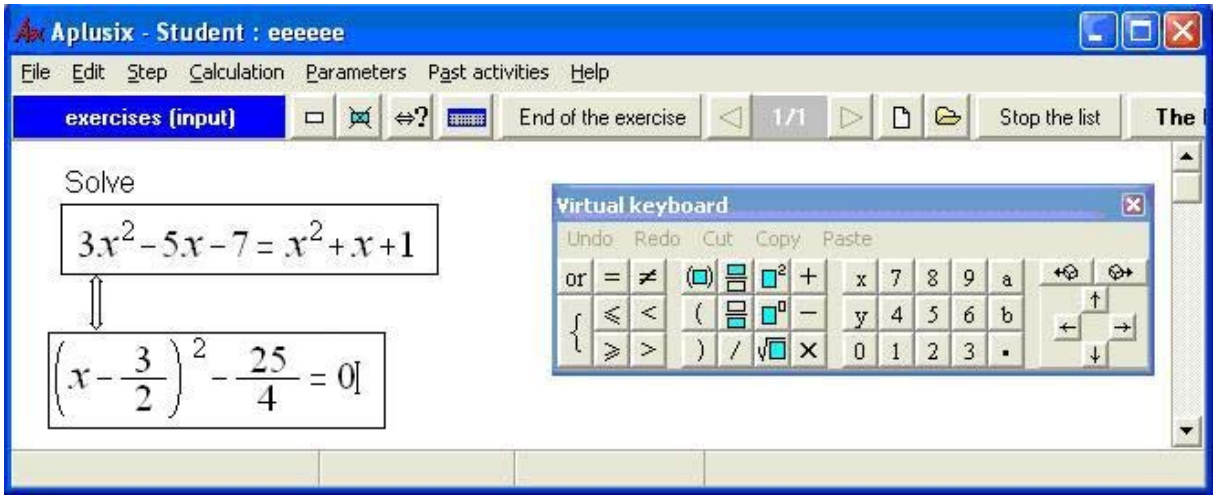


Figure 4. A screen of Aplusix before asking to enter to the Mixed-Rep at step 2. The result of the action is presented in Figure 5.

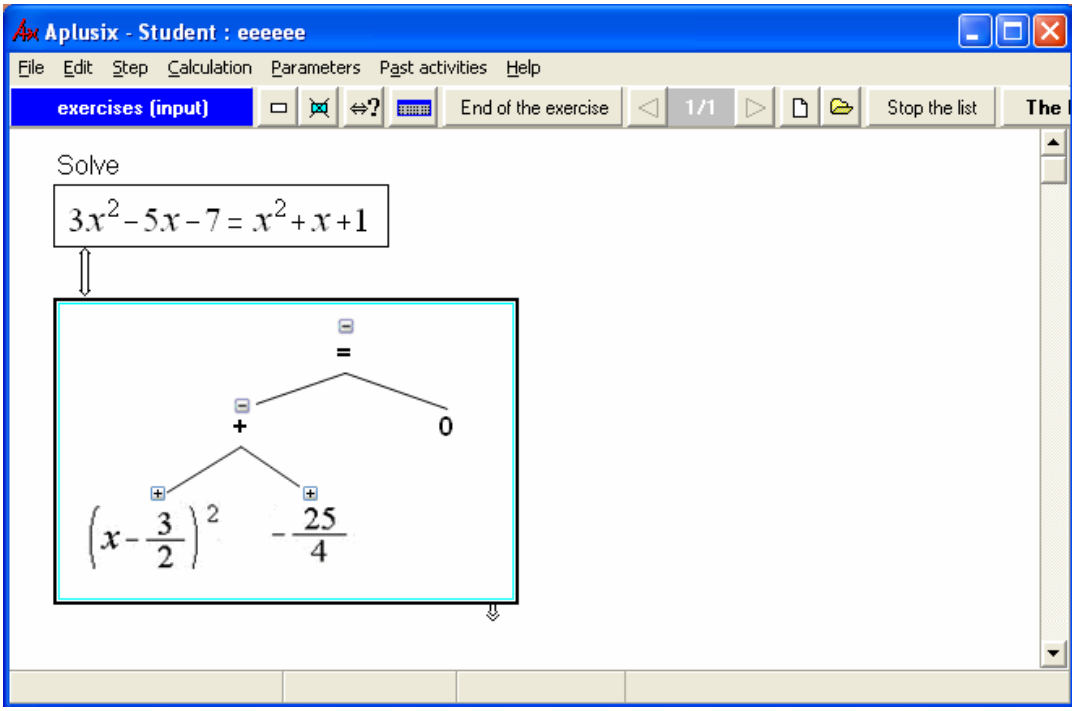


Figure 5. A screen of Aplusix after asking to enter the Mixed-Rep in step 2. The tree can be expanded or collapsed. The expression can also be modified.

The right click over a node or its ⊞ button allows the user to choose in a contextual popup menu if he/she wants to fully expand the tree or expand the tree to reach monomials.
Example:

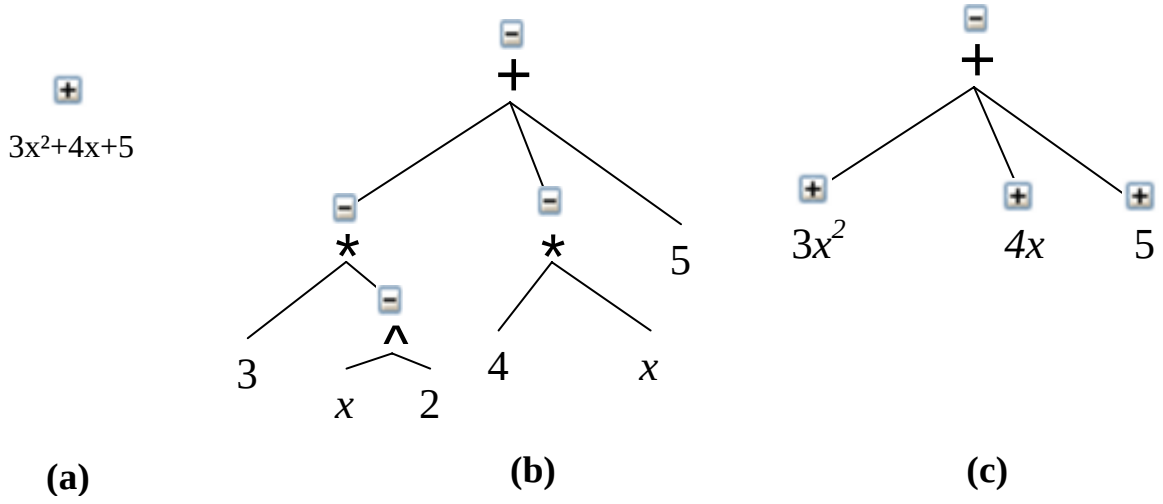


Figure 6. Available choices for expansion with the right click:
(a) unexpanded, (b) fully expanded and (c) monomial expansion

The monomial expansion is not yet implemented.

3.1.5. Free and Controlled modes

Aplusix will provide three possible modes for editing tree-rep: a free and two controlled modes. The free mode allows building many sorts of incorrect representations while the controlled mode is a scaffolding mode compelling to have operators in intermediate nodes and to respect the arity of the operators.
A parameter of Aplusix will indicate the mode. This parameter will be set by then student, the teacher or the exercise.

In the free mode the user freely builds a tree and writes inside the nodes what he/she wants. At some moments, depending of the type of activity (e.g., training, or test), the student will get feedback on the syntax of the construction.

A typical exercise consists of building a tree representing an algebraic expression given in a natural representation, using the free mode. The student starts from a tree containing an empty node and builds the tree using the editing functionalities.

In the controlled mode, the system checks at the end of the input of an internal node whether the nodes contains an existing operator or not, and verifies the arity of each operator.

In case of a non existing operator, a message is sent to the user and the insertion point is placed in the node for a modification of the content. In case of a second error, an empty content with a question mark is put in the node.

In case of an existing operator and an incorrect arity, the system adopts the following behaviour:

- if there are two many children and a correct arity can be obtained by deleting empty children (leaves containing question marks), such deletion is performed from right to left,
- if there are not enough children, empty children are added on the right to verify the arity,

in the other cases, a message is sent to the user and the insertion point is placed in the node for a modification of the content. In case of a second error, an empty content with a question mark is put in the node.

3.1.6. Well-formed expressions and others

Well-formed and fully expanded tree

Well-formed and fully expanded trees are trees whose leaves are atomic expressions:

- Positive and negative integers,
- decimals,
- variables, or
- logical constants (‘true’ and ‘false’ in English),

and whose internal nodes are algebraic operators:

- unary operators: minus ‘-’, square root ‘√’, not ‘not’ (‘not’ in English, ‘non’ in French…),
- binary operators: substract ‘-’, divide ‘/’, power ‘^’, relational operators (‘=’,’<’, …),
- variadic operators (with arity >1): addition ‘+’, multiplication ‘×’, and ‘and’ (‘and’ in English, ‘et’ in French…), or ‘or’ ‘or’ (in English, ‘ou’ in French…).

with respect of the arity and of the type (arguments of ‘+’ are not Booleans, arguments of ‘and’ are not numerical, etc.),

In a fully expanded tree, multiplication is always explicit and parentheses are absent.

Well-formed Mixed-Reps

Well-formed Mixed-Reps follow the above definition except that the leaves of the tree can contain 2D-Reps.

Ill-formed Mixed-Reps

Ill-formed Mixed-Reps are Mixed-Reps that are not well-formed. An ill-formed expression verifies one or several of the following point:

- there are operators or missing arguments represented by question marks,
- there are unknown operators in internal nodes
- there are operators with an incorrect arity
- there are unbalanced parentheses in leaves.

Expanding a 2D-rep

A 2D rep can be expanded if it is neither an atomic expression nor an unbalanced parenthesis. Unbalanced parentheses are considered as high priority operators, so '(3+2x' can be expanded as a '+' having two children, '(3' and '2x'. '(3' cannot be expanded.

Parentheses disappear in the expanded parts.

Collapsing a Mixed-Rep

An internal node of a Mixed-Rep can be collapsed if:

- it contains a known operator,
- the number of children is equal to the arity of the operator,
- the children are 2D-rep or internal nodes that can be collapsed.

Remark

The type of the expression (Boolean or numerical) is used to define well-formed expressions but has no influence when expanding and collapsing the representations.

The '-' sign

Two operators correspond to the '-' sign: the unary minus operator and the binary subtract operator. Note that the subtract operator is not associative, as a consequence it cannot be considered as a variadic operator and a-b-c cannot be considered as a subtraction with 3 arguments. Aplusix considers a-b-c as a sum of three arguments and will expand it as a sum. However, it will expand a-b as a difference, using the subtract operator represented with '-'.

During the construction of Mixed-Reps, '-' is accepted in an internal node with one child or two children.

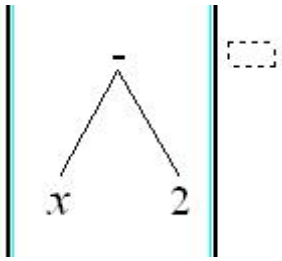


Figure 7. A minus node with two children

The parentheses

Only unnecessary parentheses are displayed in the leaves of trees in the CtrlMixed mode (see Figure 8 and Figure 9 below).
When an expression with unnecessary parentheses is expanded they disappear.

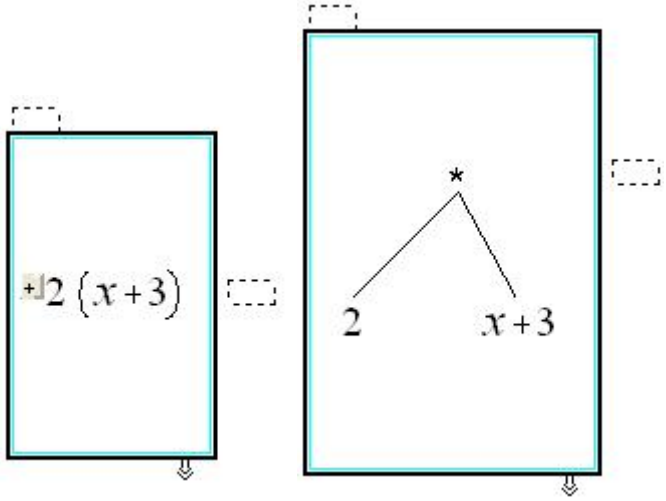


Figure 8. Expanding an expression with necessary parentheses

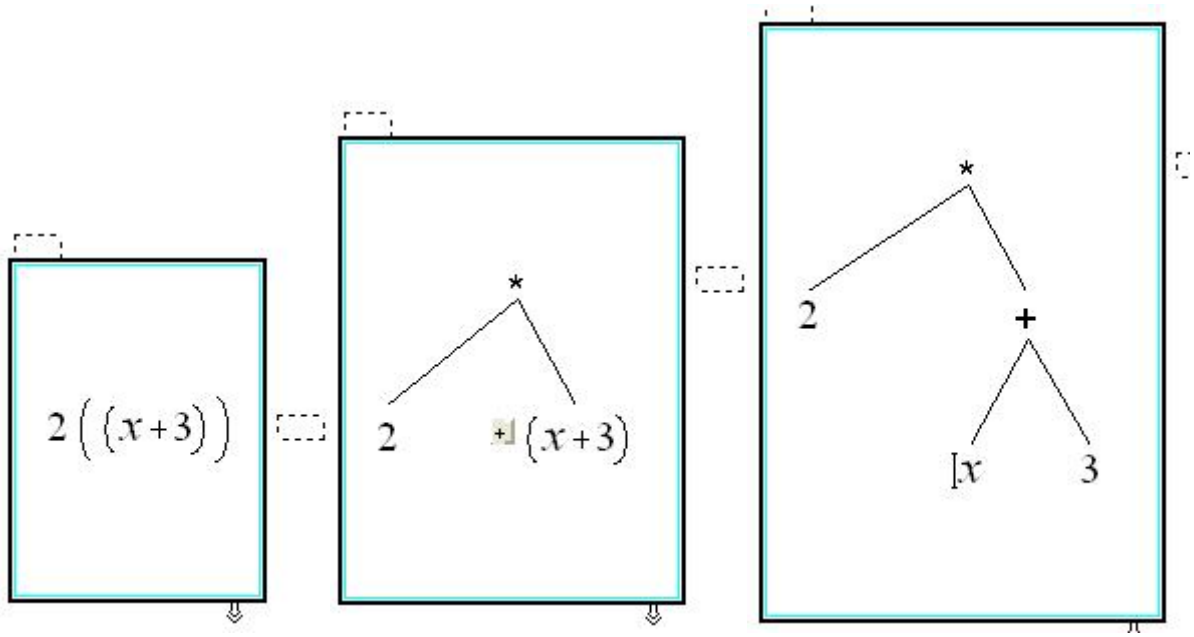


Figure 9. Expanding expression with unnecessary parentheses

3.1.7. The second view

The second view is not yet implemented.

The second view of the expression of a step is a representation of that step in another window linked to the step. See Figure 10 and Figure 11. Each view of a well-formed expression can be expanded or collapsed. When the expression is modified on the main view, the modifications are transferred to the second view (they are two views of an expression). In the case of a modification of an internal node of the first view with an unknown operator or an incorrect arity, and of a second view in which the node is collapsed, it will not be possible to draw the second view; the window will be blank or the node of the second view will be expanded.

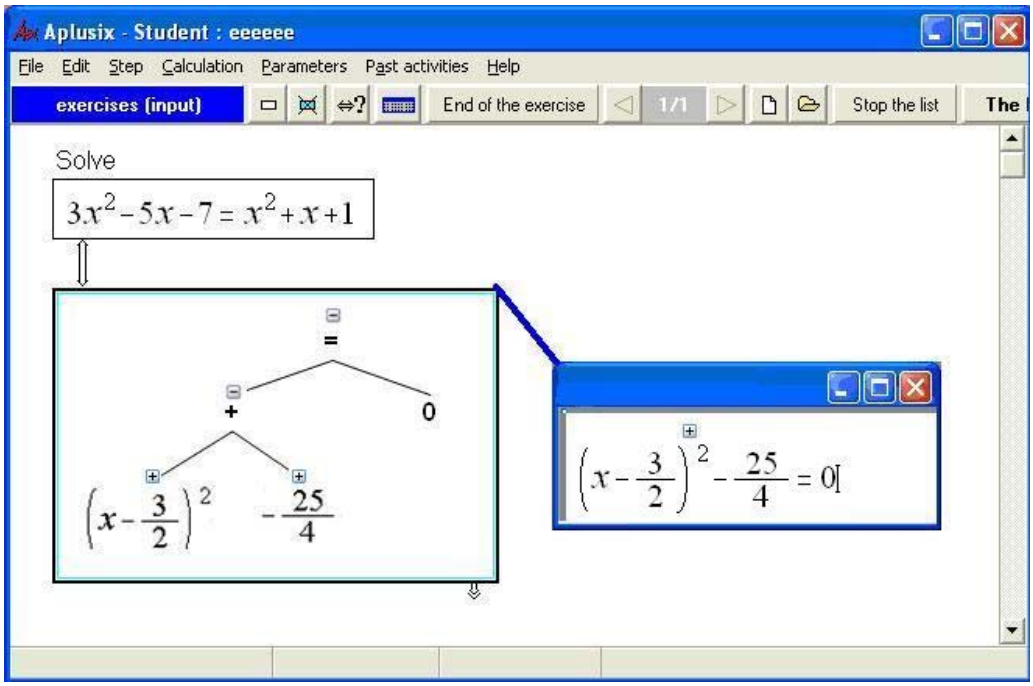


Figure 10. First example of two views.

The representation of the main window can be a 2D-rep (see Figure 11) or a Mixed-Rep and can be changed. The representation of the second window is a Mixed-Rep. Each Mixed-Rep can be expanded or collapsed independently of the other representation.

A link is drawn from one representation to the other to reify the idea of second view.

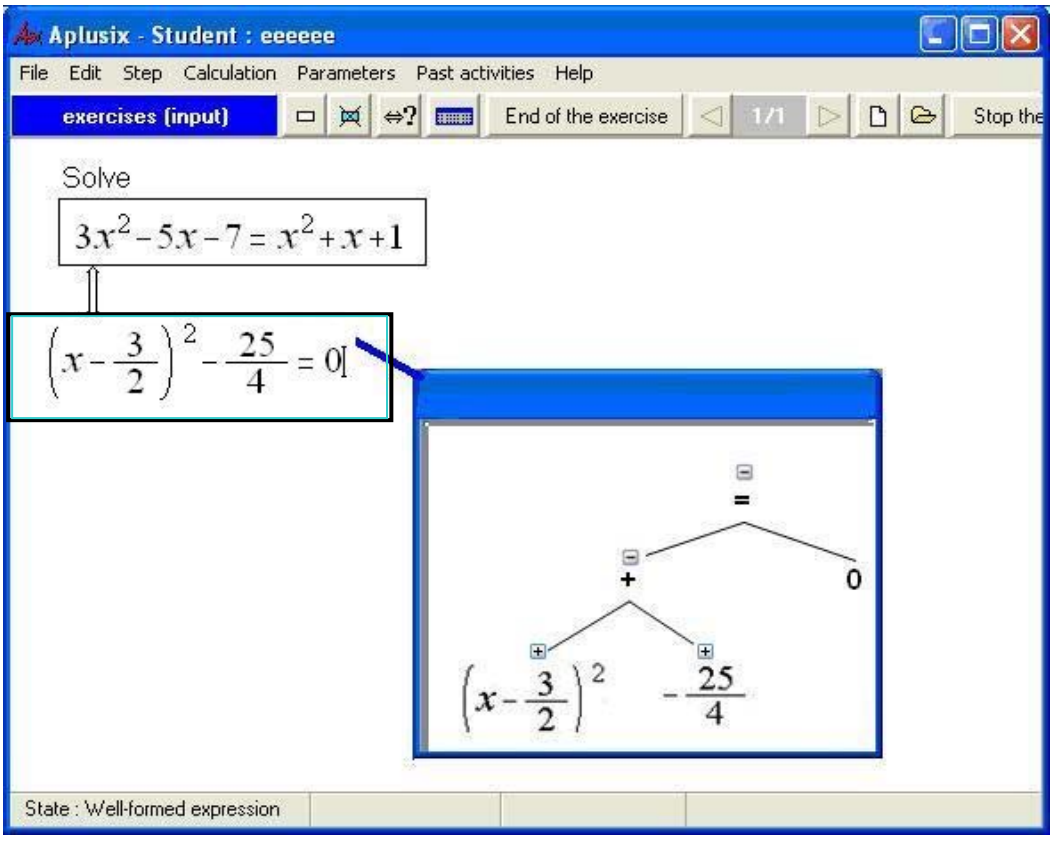


Figure 11. Second example of two views.

Of course, the expression can be fully, partially or un- expanded in each view.

3.1.8. The empty node

The empty node has two roles:

-the starting point of an expression represented as a tree in the FreeTree mode.

-a temporary node during insertion or editing of a new node

Figure 12. Starting point in the FreeTree mode

It is represented with an empty rectangle with a dotted outline
It is allowed both as internal and as leaf node in the FreeTree mode.
Thus, the FreeTree mode will allows “fill in blanks” exercises

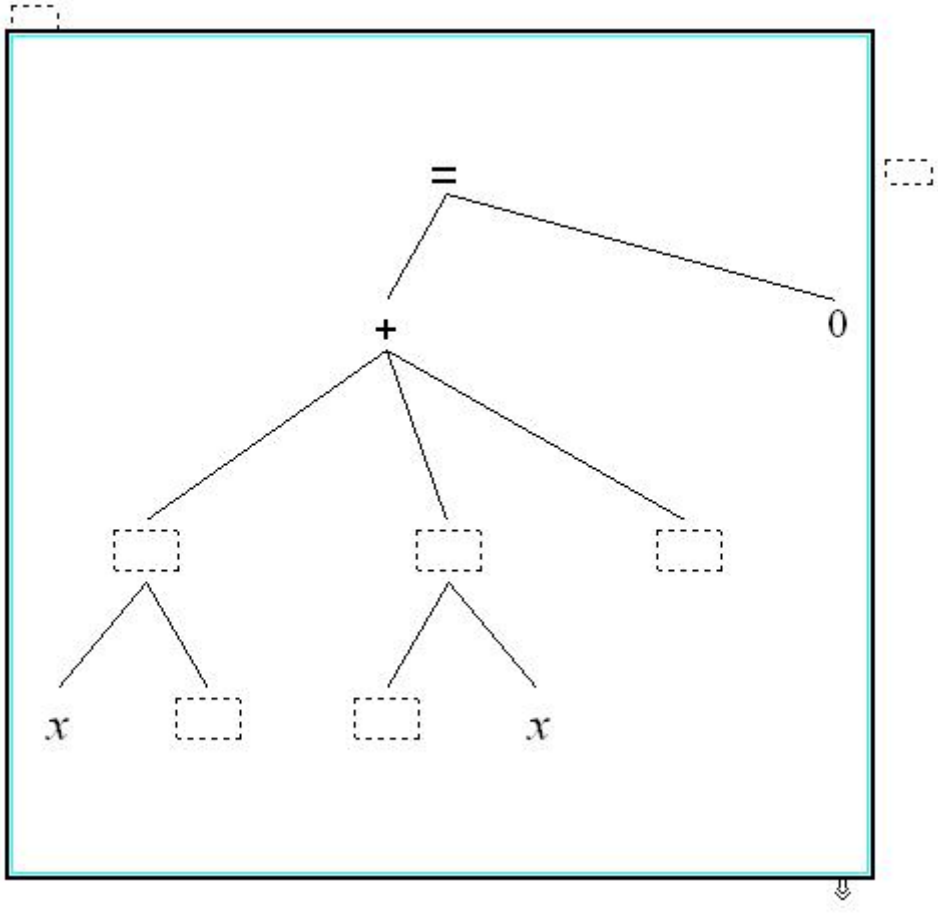


Figure 13. A partially filled tree

In the CtrlTree and CtrlMixed mode, empty nodes are leaf nodes that are represented by question marks.

3.2. Insertion in a tree

3.2.1. Adding a mother operator

Common behaviour

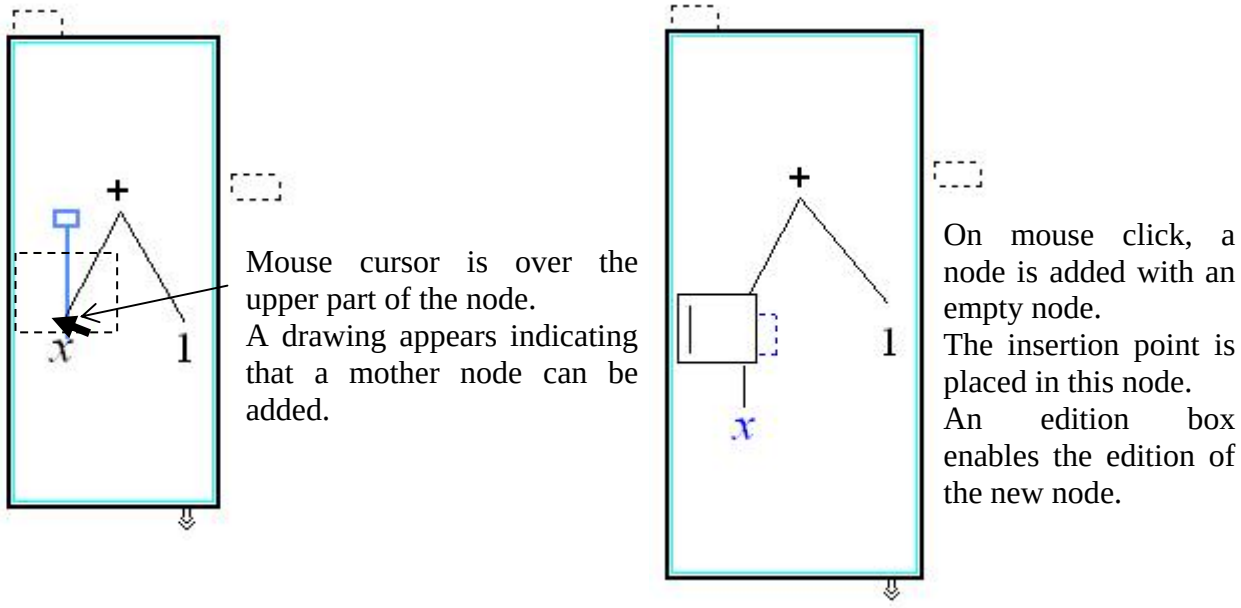


Figure 14. Adding a mother node, common behaviour:when the user moves the mouse over the upper part of a node, an insertion drawing appears If the user clicks in this situation, an empty node is inserted as a mother node and the system is in an edit mode. The user enters the operator with the keyboard. At the end of the input of the operator, the behaviour differs regarding the current control of the mode (free for FreeTree or controlled for CtrlTree and CtrlMixed)

Free Mode

In the free mode, the system does not check the content of the node: unknown operators are accepted and nothing is done for known operators (so the arity may be incorrect).

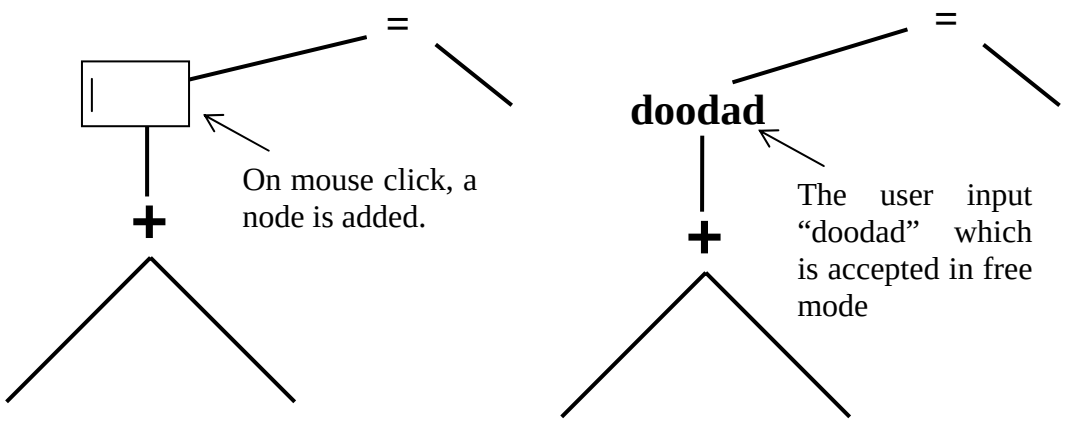


Figure 15. Example of adding a mother node in free mode

Controlled Mode

In the controlled mode, the system checks the new operator.
When the operator is known:

if is arity is one, nothing is done because the situation is correct,

if is arity is greater than 1: the system add empty children on the right to respect the arity.

When the operator is unknown, a message is displayed to inform the user who has to change the content of the node; the user can make a new input or delete the new node by simply hitting BackSpace or Del key on an empty node to return to the previous situation.

For example:

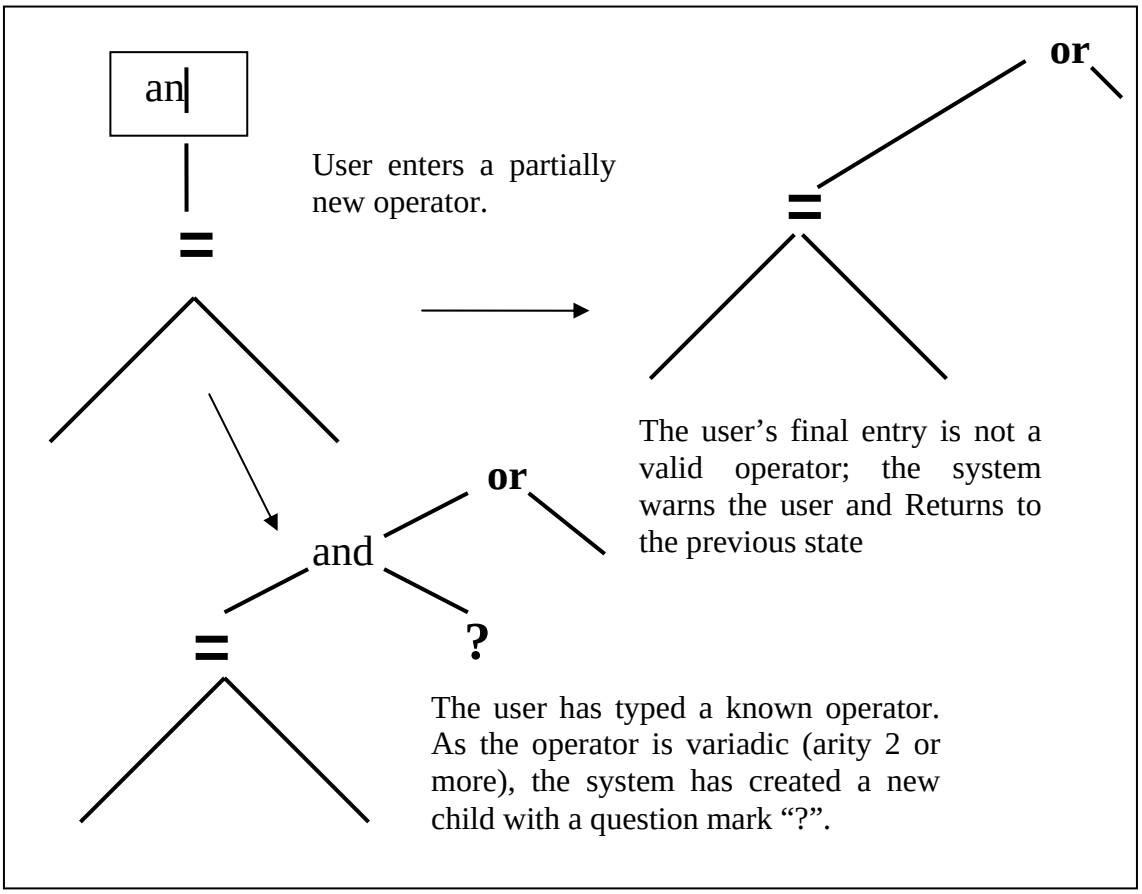


Figure 16. Editing a new mother operator: only operators are allowed in internal nodes. Arity has to be respected.

3.2.2. Adding a child

Common behaviour

When the user moves the mouse over the lower part of a node, an insertion drawing appears in the zone which indicates that a child can be added. If the user clicks, a new empty child is created ..

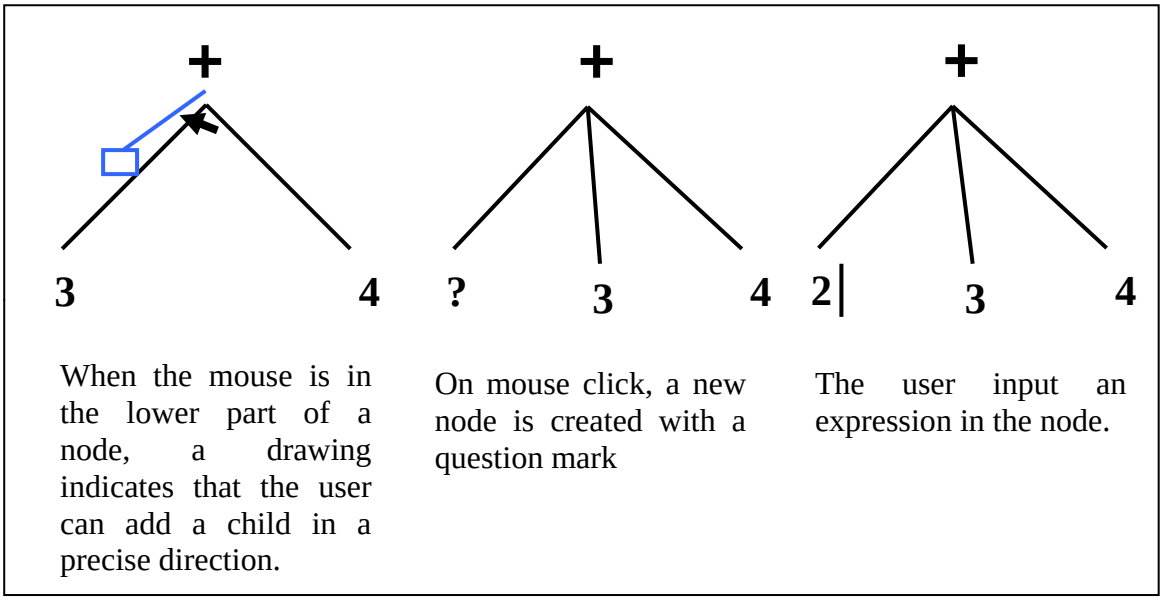


Figure 17. Adding a child

Free Mode

In the free mode, the addition of a child is accepted and a node edition box is displayed. The edition procedure is the same as the edition of nodes

Controlled Mode

In the controlled mode:
If the arity of the operator is constant, the user cannot add a child node. Moving the mouse under the node will not produce any drawing.
If the operator is variadic, the addition of a child is accepted.

In CtrlTree mode: If the new child is an operator, its children are automatically drawn with question marks

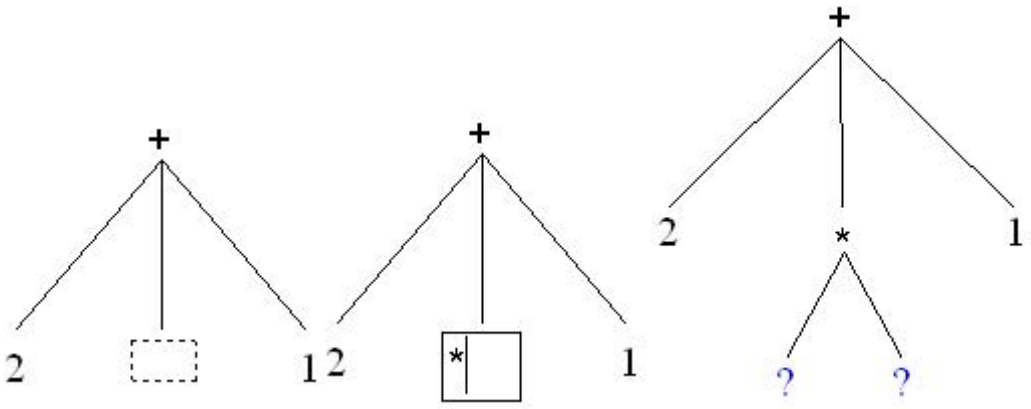


Figure 18. Insertion of new child in the CtrlTreemode

In the CtrlMixed mode the new child appears as an empty node represented by a question mark.

This child can be edited in a 2D natural representation as shown in the figure below.

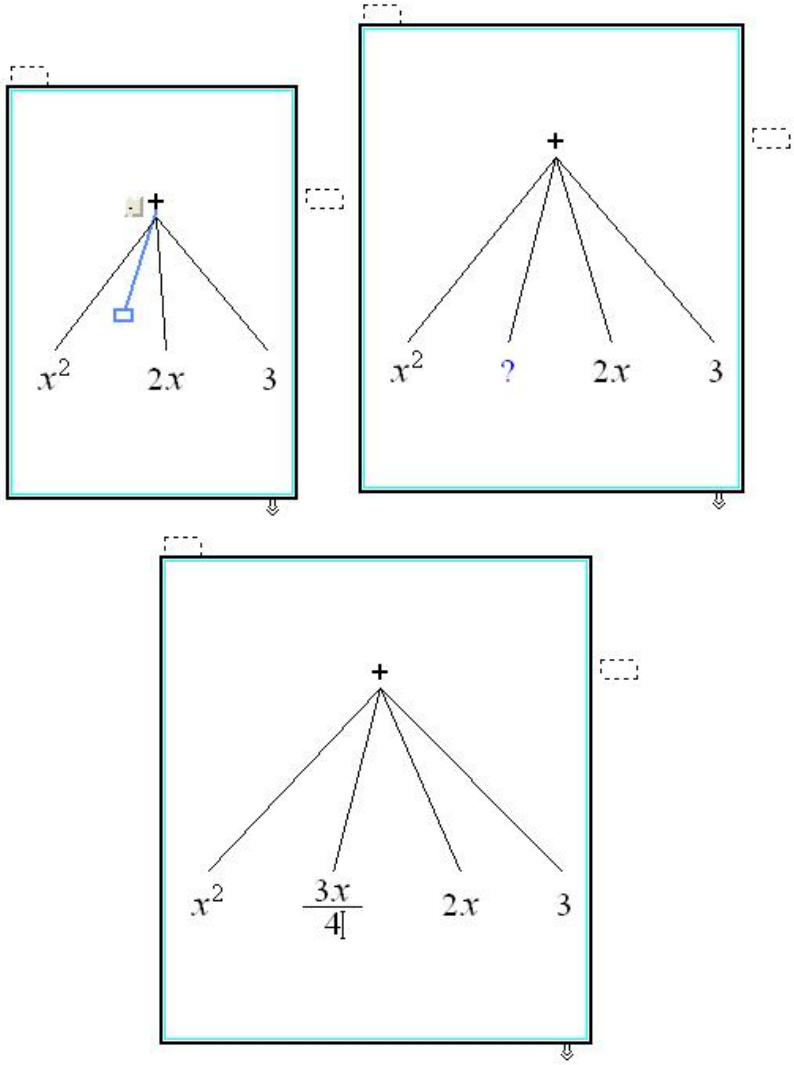


Figure 19. Insertion of a child node in the CtrlMixed mode

3.2.3. Mouse Cursor and Add drawings

The editing of a Mixed-Rep depends on the location of the mouse cursor. When the mouse cursor is inside a leaf, a click places a 2D-insertion point in the leaf. and as leaves are 2D-reps, the editing is the editing of 2D-reps of Aplusix.
When the mouse cursor is not inside a leaf, the cursor keeps its shape but an add drawing appears if the mouse cursor is over a zone defined to detect adding events. editing becomes a tree editing. Add drawings are made of a straight line followed by a small rectangle as shown and are drawn in blue in Figure 20..

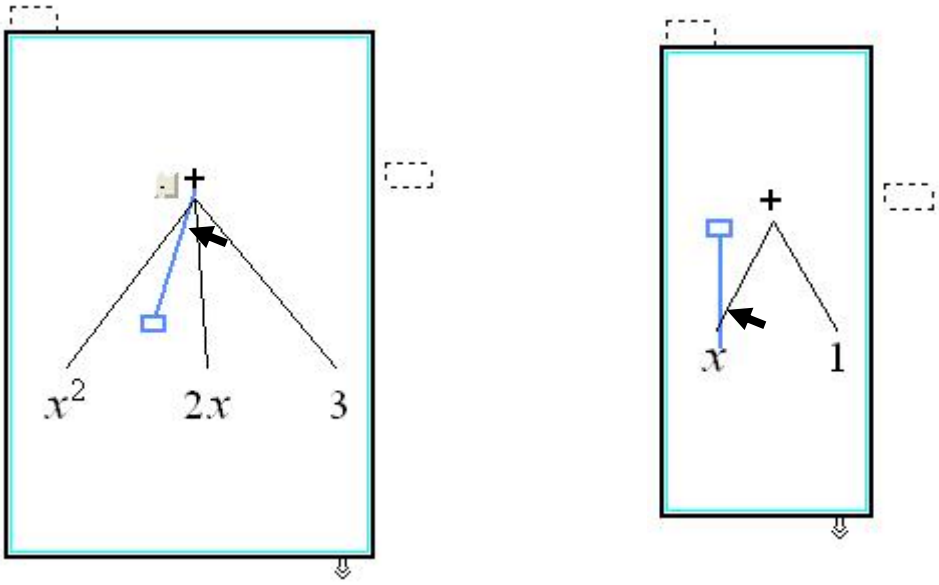


Figure 20. An add drawing appears when the mouse cursor fly over the zone on the top of a node or near the branches of its children.

3.3. Selection in the tree

The 2D-rep selection of Aplusix has an algebraic behaviour. It allows selecting sub-expressions and only sub-expressions. So 3x can be selected in 3x+4y, but x+4 cannot. Several arguments of a variadic operator can be selected using a ctrl-click after a first selection. So 3x+5 can be selected in 3x+4y+5+z. As the natural representation allows interpreting -4x either as the product of -4 and x or as the opposite of 4x, the selection mechanism allows selecting -4 in -4x although the internal representation of -4x is the opposite of 4x.
The selection command in the 2D-rep, as in many applications, is based on the movement of the mouse when the left button is down. It produces the smallest sub-expression that covers the dragged area. As in many applications, ctrl-click is used to provide a complementary part to a selection.

In a tree, a sub-expression is very easy to locate, it is a sub-tree. So it is not necessary to drag over an area to indicate a selection. The selection mechanism that will be implemented for tree-reps is the following:

- A double click on a node (either an internal node or a leaf) selects the sub-tree having this node as root,

- A ctrl-click on another child of the mother node of a selection will add a sub-tree to the selection,
- It will not be possible to selects “-4” in the opposite of 4x, because -4 is a very bizarre area of the tree and we don’t want to suggest that bizarre areas can be selected.

A selection is reified by a triangle with a coloured background over the selected sub-trees.

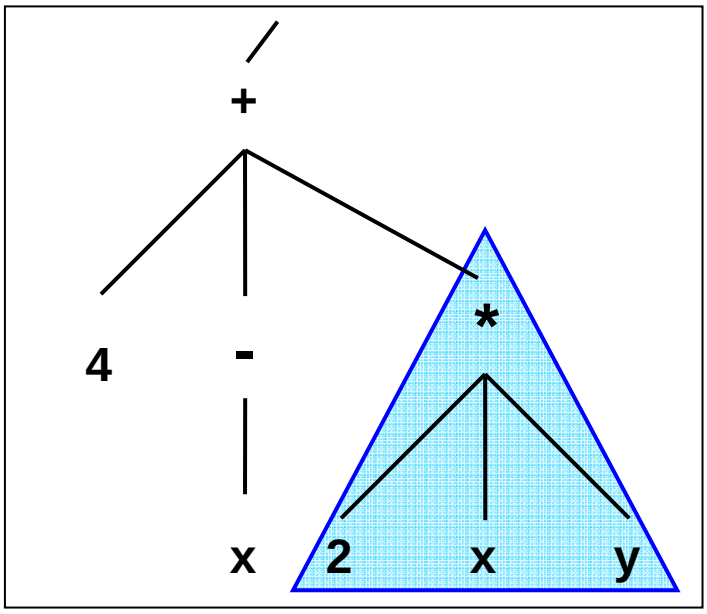


Figure 21. Selection of a sub-tree after a double click on “*”.

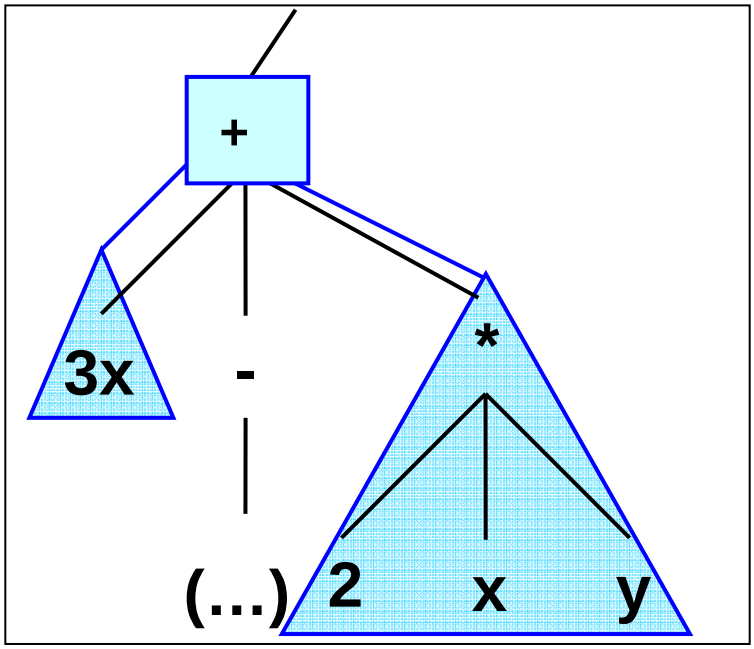


Figure 22. Selection of “3x + 2xy” in “3x-(…) +2xy”. This selection can be obtained: (1) by a double click on “3x” then a Crtl-click on “*”; (2) by a double click on “*” then a Crtl-click on “3x”.

3.4. Modification of an internal node

In the free mode, the user can freely modify internal nodes.

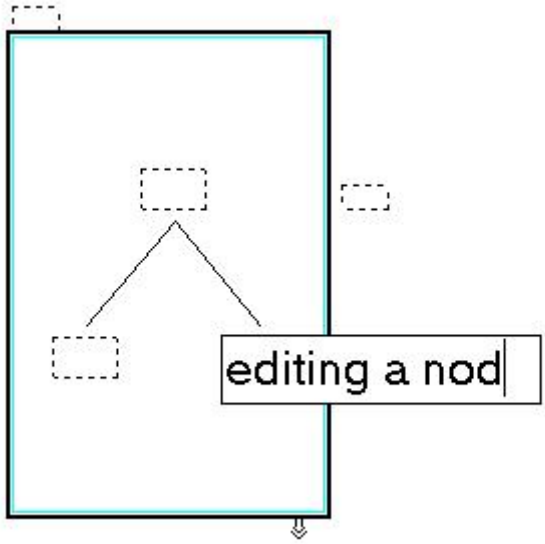


Figure 23. Editing a node in the free mode

In the controlled modes, the modification of an internal node is possible when the new operator is known and the arity is respected or can be respected by adding or subtracting empty children. In other situations, the system refuses the modification, replaces the previous operator.

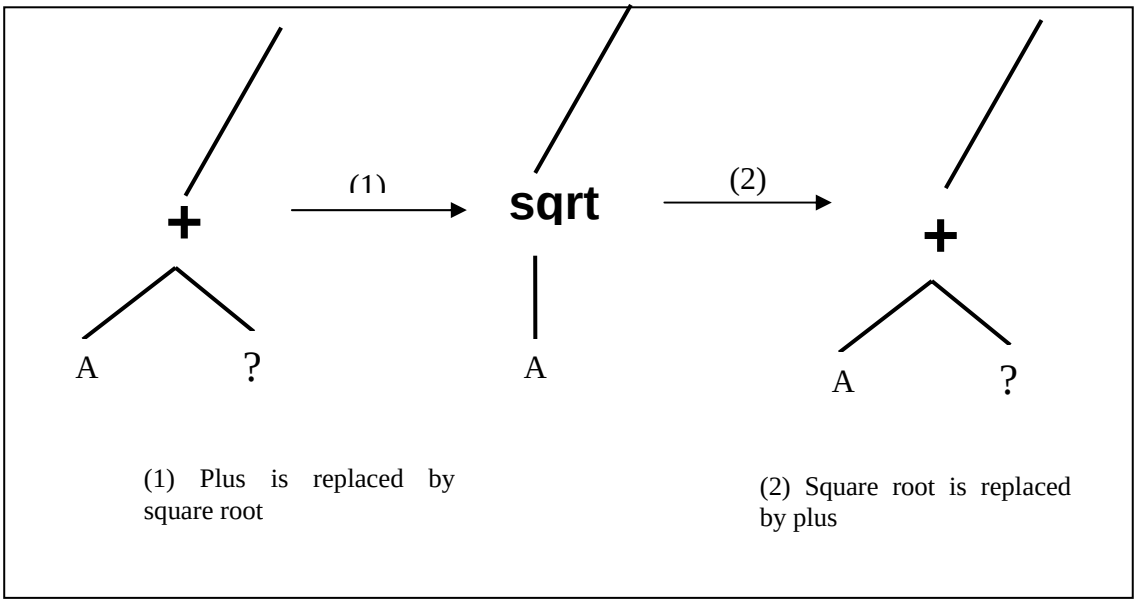


Figure 24. Modification of an internal node in controlled mode when a change of arity is possible by deletion or insertion of empty children.

3.4.1. Insertion on selection

Insertion on a 2D selection works in usual 2D context of Aplusix.

Concerning tree selection, when arguments are inserted (input of a digit or a letter, paste from the clipboard), the selected sub-tree is removed and replaced by the input.

In the free mode, insertion of an operator is free, whereas in controlled mode insertion of an operator over a selection can be made according 3 principles:
Principle 1: *The input replaces the selection.* The selected sub-tree is removed and replaced by the operator with a number of arguments according to its arity, each argument containing a "?".
Principle 2: *The input of an operator applies to the selection*. The selected sub-tree is replaced by the operator and becomes the first argument of this operator.
Principle 3: *The input of a unary operator applies to the selection*. Nothing appends for non unary operator; the insertion of a unary operator works like in principle 2.

These 3 principles can be implemented with a parameter for choosing the active one. Note that the current principle of Aplusix is principle 3.

3.4.2. Deletion

With a 2D-cursor, Backspace (<-) key and Del key (->) delete elements like Aplusix does currently.

Backspace or Del over a selection deletes the selection.

In the free mode only the selection is removed, whereas in controlled mode the following rules are applied.

Consequence of a deletion:
- when the mother operator is variadic with more than 2 arguments, nothing more appends (the operator moves from n to n-1 arguments)
- when the mother operator is variadic with only 2 arguments (the suppressed argument and one another), the variadic operator is removed and replaced by the remaining child,
- when the mother operator is a binary operator:
    - ° if the suppressed child is '?', the operator is suppressed and replaced by the remaining child
    - ° if the suppressed child is not '?', the child is replaced by '?'.
- when the mother operator is a unary operator:
    - ° if the suppressed child is '?', the operator is suppressed and replaced by '?'
    - ° if the suppressed child is not '?', the operator is suppressed and replaced by '?'.

3.5. Cut, copy, paste, and drag&drop

Cut a selection places a copy of the selection in the clipboard and deletes the selection.

Copy a selection places a copy of the selection in the clipboard.

Paste over a selection replaces the selection with the content of the clipboard.

Paste on the insertion point requires an operator to link the displayed expression and the content of the clipboard. When the insertion point is a 2D insertion point in a leave, the behaviour is the current behaviour of Aplusix. When it is in an internal node, the content of the clipboard is added as a child if the operator is variadic, otherwise, nothing is done.

Drag&drop works like cut and paste. It is particularly useful for changing the order of the arguments of an operator.
Example (Figure 25):

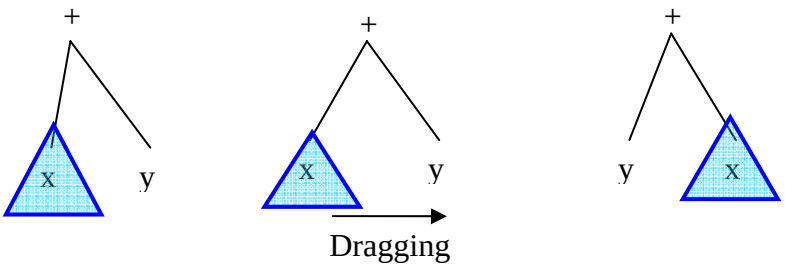


Figure 25. Swapping two branches using drag&drop

4. Specifications of the graphical representation system

4.1. General point of view, global insertion in the Aplusix system

The main objectives of the graphical representation system are:

1) To show curves representing objects corresponding to algebraic expressions (in short, we will say curves representing algebraic expressions),
2) To situate several curves in the same space,
3) To show that equivalent expressions lead to identical curves.

The main principles adopted in this module are:

1) To represent curves with the closest pixels of the exact points,
2) To show the continuity/discontinuity of the curves with contiguous pixels for continuous parts and explicit drawings of discontinuous parts,
3) To allow different thicknesses of curves in order to help to understand relative positions of curves.

The user of Aplusix will have the possibility to ask the system for displaying a graphical representation of an object associated to a given algebraic expression (polynomial or rational expression of one variable, equation or inequation of one unknown, system of equations of two unknowns). For that purpose, the user will use the popup menu of a calculation step and ask for a graphical representation. This graphical representation will appear in a separated window. It can be dynamic or static, according to the student's choice, dynamic meaning that each modification of the expression will be transferred on the graph, leading to a different curve. A legend will be displayed under the graph.

The student has the possibility to display several curves (associated to different calculation steps) on the same window. They will appear with different default colors, default settings are to be defined. Curves are displayed according to the order of the legend. A part of a curve can hide a part of another curve.

The legend will be interactive: The popup menu will allow modification of the thickness and the color; the student will have the possibility to change the order of the items of the legend, changing by this the order of the display of the curves. On a click on the legend, the corresponding object will blink during a while:

- In case of rational and polynomial expression: It is only possible to click on the expression, the corresponding curve blinks.
- In case of equation and inequation:
    i. Click on the left or right part: The corresponding curve blinks.
    ii. Click on the equality, inequality sign: The solution set blinks.
- In case of system of equation :
    i. Click on an equation of the system: The corresponding lines blinks.
    ii. Click on the system (curly brace): The solution set blinks.

The user can remove a curve. To do that, he/she selects the expression in the legend and hit the Del key. The user also can move items of the legend with a drag and drop which change the order of display of the curves.

Adequate choices of thickness and color allow showing merged parts of different curves, see Figure 32 page 66 and Figure 33 page 66 .

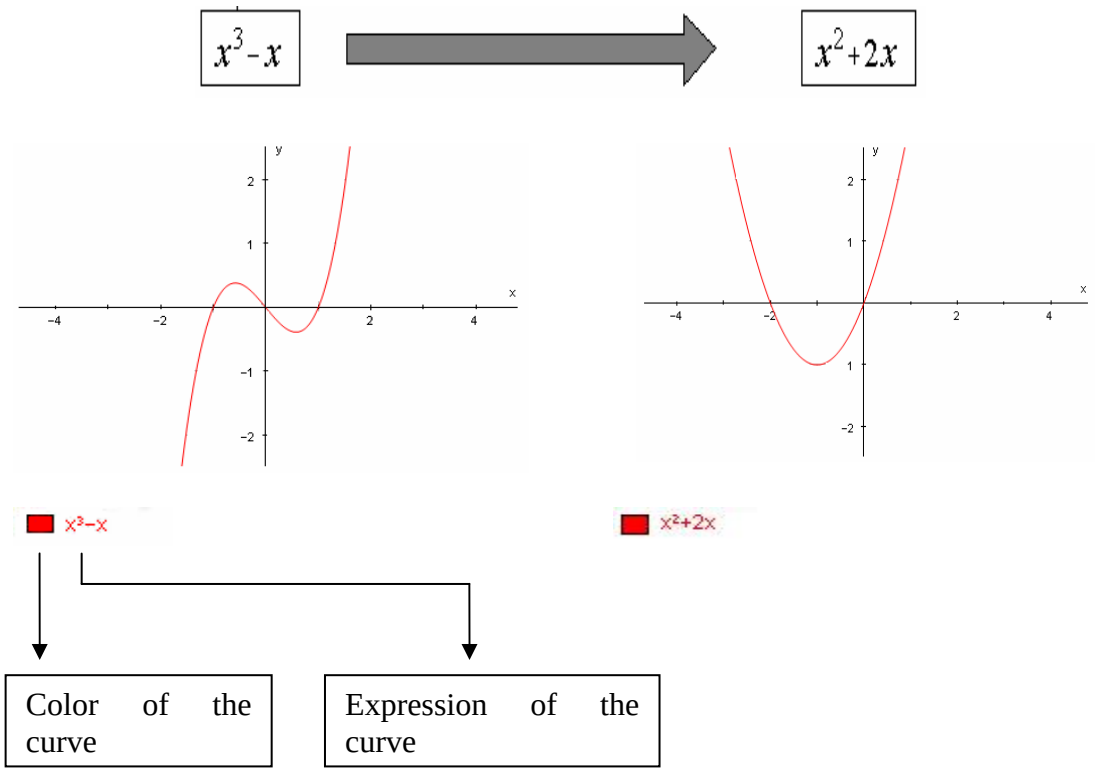


Figure 26. Representation of an expression and dynamic effects. On the left, a curve displays an expression. On the right, the expression has bee modified in the calculation window; this has been transferred to the graphical window.

4.2. Representation of polynomial and rational expressions

Polynomial and rational expressions of one variable are represented as curves in the 2D-space.

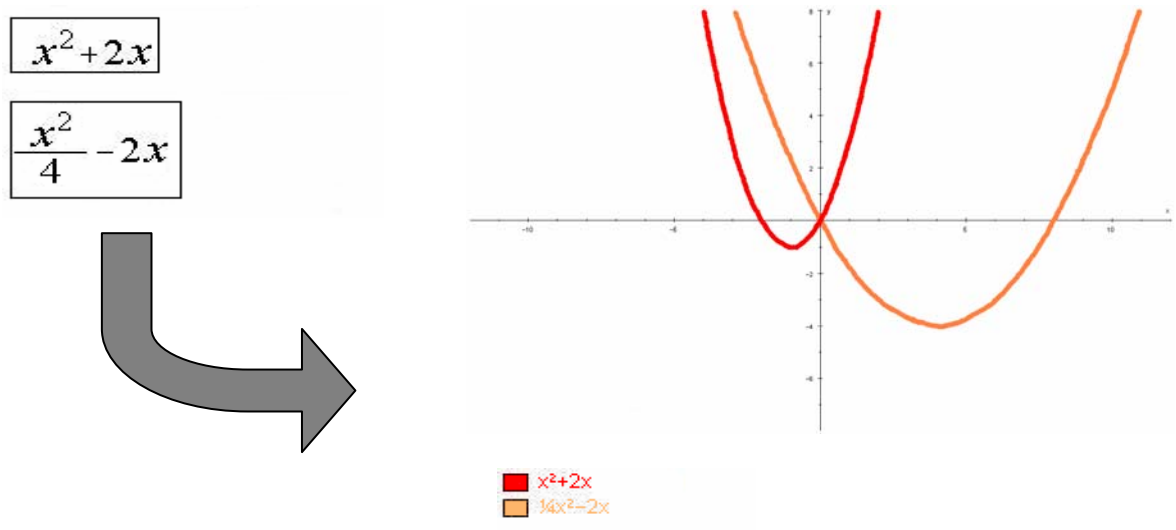


Figure 27. Representation of polynomial expression.

4.3. Representation of equations and inequations

An equation of one unknown of the form f(*x*)=g(*x*) is represented in two spaces: First, in a 2D-space, the curves of f(*x*) and g(*x*) are displayed; Second, in a 1D-space, the set of solutions is displayed. The 1D representation is the true graphical representation while the 2D representation is an intermediary representation to help to understand the 1D one.

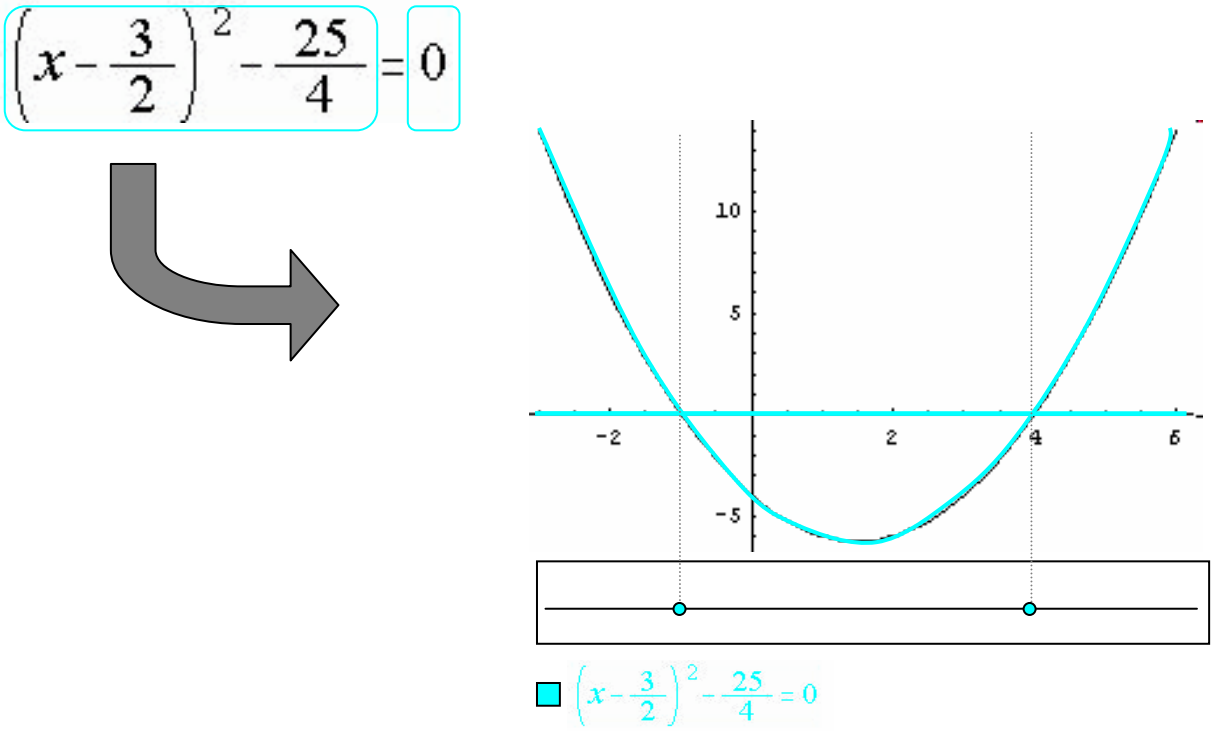


Figure 28. Two representations of an equation.

The user will be able to display multiple equations in only one 2D-space, whereas each solution will be displayed in its own 1D-space.

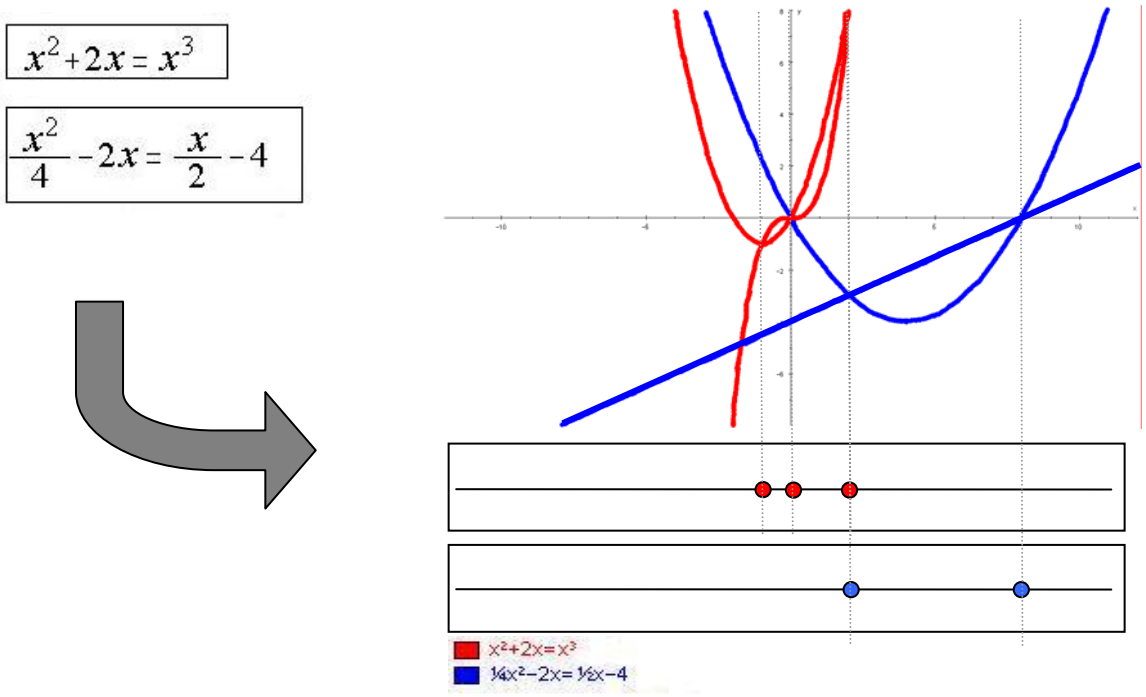


Figure 29. Representation of multiple equations displayed in the same 2D-space and solutions in a 1D space for each equation.

Inequations are analogous to equations.

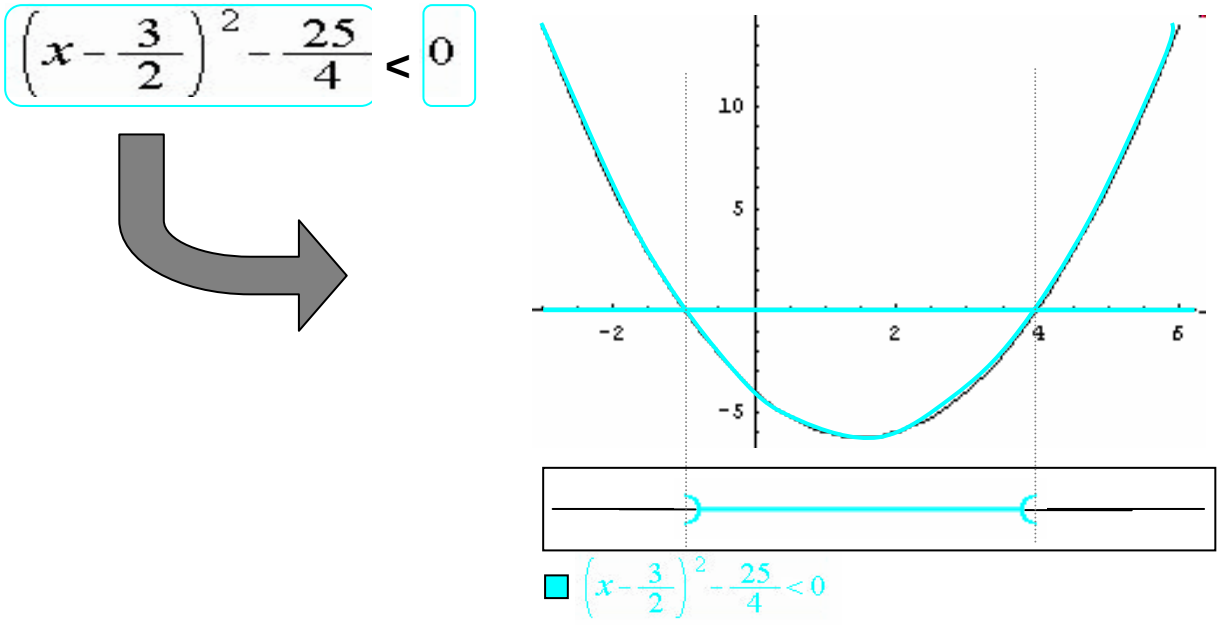


Figure 30. Representation of an inequation and its solution.

4.3.1. Representation of Systems of Equations

Linear systems of equations and 2 unknowns are represented with lines in the 2D-space representing the equations. The intersection of these lines (in the case of 1 solution) is the solution of the system in the 2D-space. If there are an infinite number of solutions, lines representing equations are merged and the set of solutions is the merged lines.

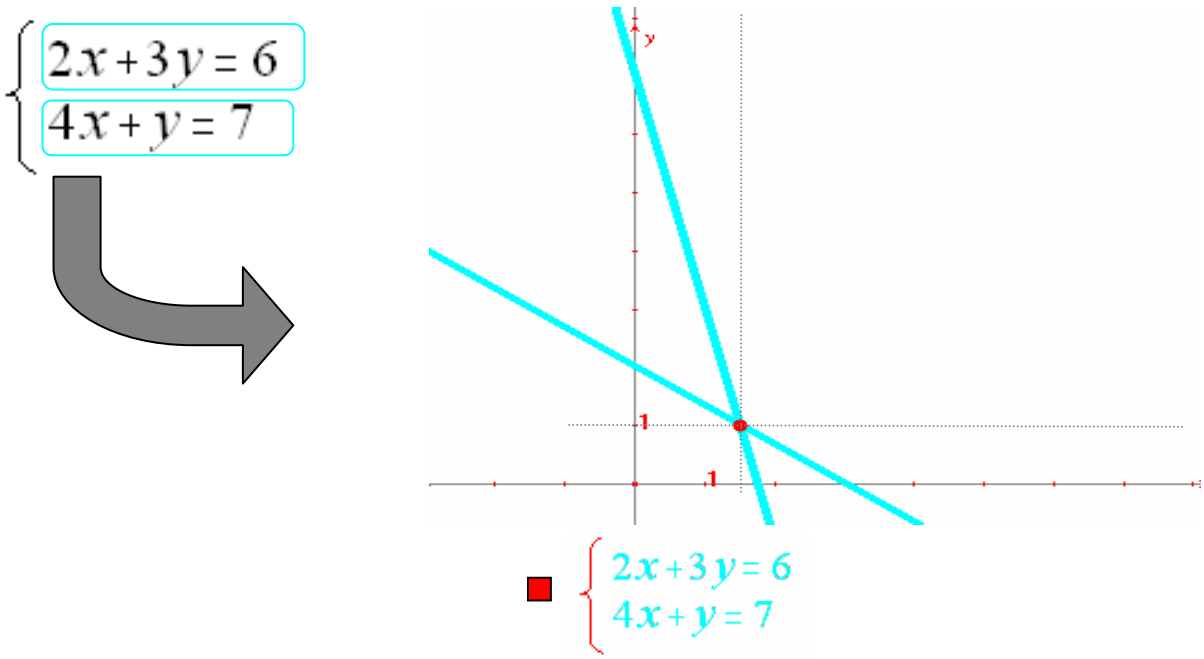


Figure 31. Representation of a system of equations

4.4. Representation of merged parts of curves

The thickness and the order of the display allow, when appropriately chosen, allow seen the merged part of curves as shown in figure 6 and 7...

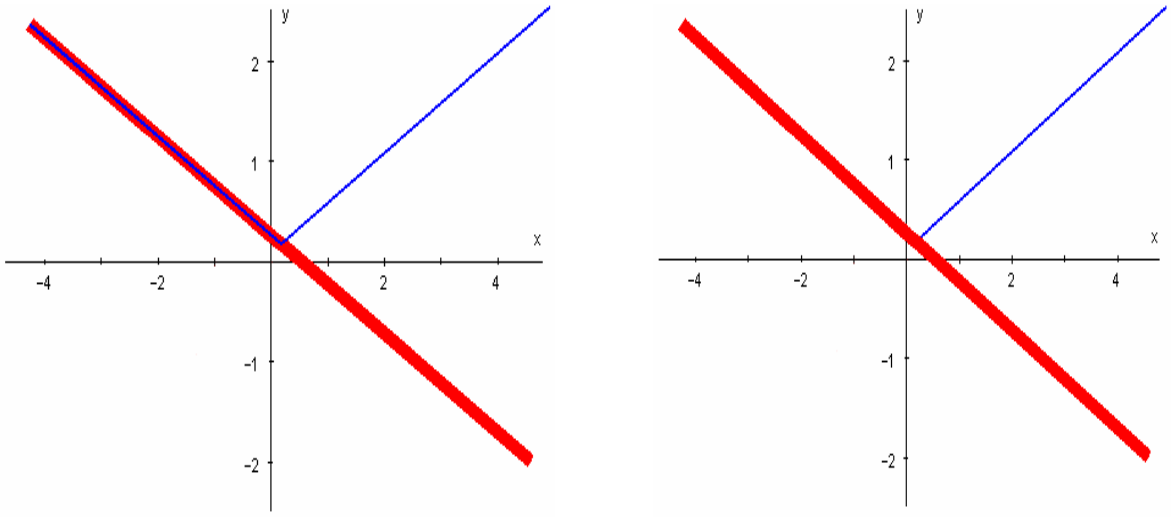


Figure 32. Partly merged curves. On the left, the blue curve has the higher value in the display order; on the right, it has the lower value.

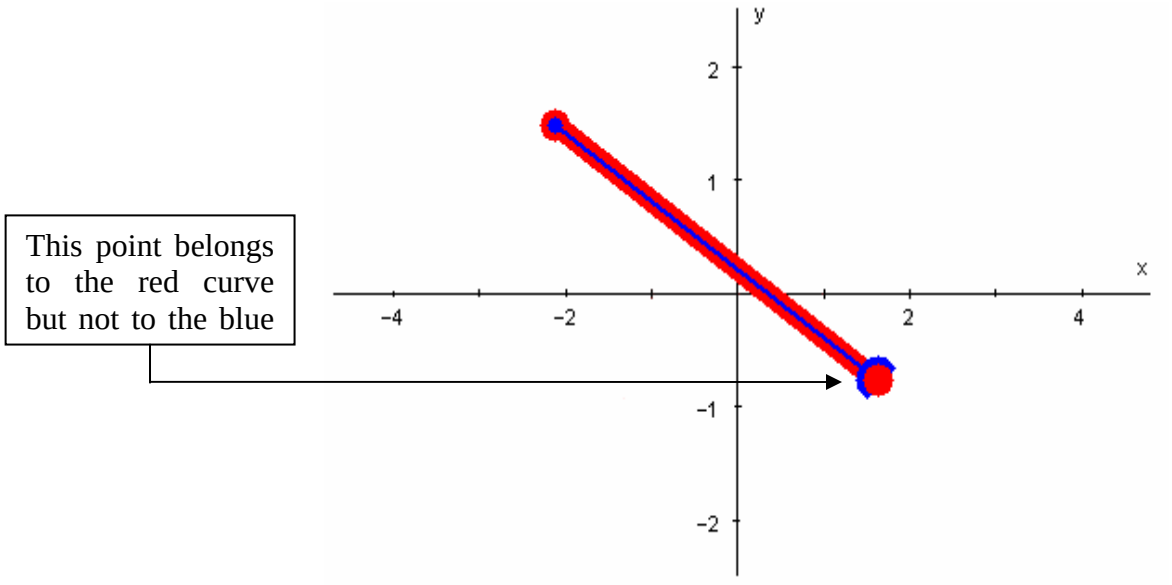


Figure 33. Example of points belonging to a curve or not.

DDA 3: Alnuset

Authors: G.Chiappini, B. Pedemonte, E.Robotti

Istituto Tecnologie Didattiche - CNR

Work progress in 2006

During the first year of ReMath project ITD team has worked at the development of the two components of ALNUSET, namely the Algebraic line and the Algebraic manipulator components.
* As far as the Algebraic line component is concerned, the work has been focused on the internal and the external representation of this component, namely on the DoubleLine rendering (coordinates transforms, two lines, ticks, number labels) and on the DoubleLine tool (architecture and operations: ADD, MUL POW). Moreover, the working domain (N,Z,Q,R), the variable and its type (integer, rational, real) and the variable drag have been implemented. Finally, polynomial root tool, auto-scrolling and zoom function, a displayed full post-it for operands have been developed.
* As far as the Algebraic manipulator component is concerned, the work has been focused on the internal and external representation of this component. In particular, expression subset management (representation, picking, rendering), algebraic Manipulator display component, a partial set of rules, and a first version of the user rules (theorems) management have been implemented. Moreover, the selection of part of expression, the activation of available rules, and the execution of the rules have been developed.

At the end of November ITD team will make available to the experimenter team a prototype of ALNUSET constituted by two separate Java applications, the Algebraic Line component and the Algebraic Manipulator component.
Their integrations and some particular functions of these components will be ready in 2007, at the end of January. In particular for this date the “truth set” function of the Algebraic line that has been presented in the last version of the “ALNUSET DDA specification” will be implemented. Moreover for this date, equation, inequations and the factorisation functions not yet implemented in the algebraic manipulator (since they require the integration of the two components) will be make available.

ALNUSET DDA SPECIFICATION

1. INTRODUCTION

The main problem of the didactic of algebra regards the relationships of the students with the algebraic language, that is to say their relationships with the construction, interpretation and transformation of algebraic expressions and propositions.

An algebraic expression is a writing composed of numbers and/or letters connected by the symbols of mathematical operations (addition, subtraction, multiplication, division, power of a number, extraction of root); this expression indicates the result of the operations performed in sequence.

If the algebraic expression contains only numbers and operands, it is a numerical expression otherwise it is a lettering expression. The main difference between them concerns the fact that the former indicates, in a *determinate* way, the number that is the result of the operations performed in sequence and it denotes such a number (i.e "3+2*5" and "11+2" denote the same number) while the latter indicates, in a *indeterminate* way, the result of the operations performed in sequence and it denotes the function that associates the result of the operations performed to each value of the letters (i.e . "2a+1" denotes the function f={ x→2x+1})

An algebraic proposition is a writing that expresses a relationship between two algebraic expressions by means of the signs =, >,<. It denotes a truth value (true/false) that is univocally determined whether the two expressions of the proposition are numerical (i.e the proposition 2*3+2=10-2 denotes "true"; the proposition 2*3=5 denote "false") while it depends on the value assumed by the letter when at least one of the two expressions contains letters (i.e the proposition "2x-5=x-1" denotes "true" for x=4 while it denotes "false" for all the other numerical values of x).

Research has highlighted great difficulties when students pass from the use of numerical expressions and propositions that characterize the arithmetic activities to the use of lettering expressions and propositions that characterize the algebraic activities. Difficulties concern the control of what a lettering expression indicates in undetermined way or what is the truth value of proposition according to the value assumed by the letters (parameters and unknown) that characterize it. Two types of specific behaviours that reveal a bad relationship of the students with the algebraic language and a lack of algebraic thought have been evidenced by the research.

There are students who operate only at a syntactic level. They consider the algebraic writings as strings of arbitrary symbols which are governed by arbitrary rules, namely by rules that have not to be justified. These students are unable to imagine the entities (numbers, function, truth values) that are the referent of such writings. For these students, algebraic expressions and propositions are "things" in themselves that do not stand for anything else. Their algebra is purely syntactic; for these students a mathematical expression or proposition denotes itself.

In contrast other students are unable to use the algebraic language and the methods of algebra for facing the assigned tasks. They prefer to elaborate solution strategies using different representations such as the verbal language and the methods of arithmetic.

Research evidenced that in both the two considered cases there is not a development of a genuine algebraic thinking seen as capability of constructing algebraic expressions to generalize properties of the numerical sets or to name the elements characterizing a problem situation and to construct an algebraic proposition for solving it.

Students' difficulties are mainly due to a lack of control of the fact that:

- numeric values can be associated to a lettering algebraic expression (i.e. y(2x+y)),
- these values depend on those assigned to x and y (the denotation of the expression is a function) and that these correspondences of values are not modified by algebraic transformation
- a proposition denotes a truth value that is defined by a specific set (the truth set of the proposition) and that the symbolic transformation of a proposition by means the application of properties of equalities and inequalities does not change the truth set.

To overcome these difficulties new methods and tools are necessary to support students in:

- grasping the invariance of the denotation as regards the changes of sense of an algebraic expression or proposition
- recognizing that a syntactic structure of an expression can reflect in iconic way property of a numeric set and constructing lettering expressions that generalize specific properties
- recognizing the sense of an algebraic expression in relation to the aim for which an algebraic expression or a proposition is constructed or transformed

Through the ReMath project we are working to develop a new educational digital artefact named ALNUSET (ALgebra on the NUmerical SETs) with the aim to pursue the previously described educational goals.

1.1. Brief presentation of ALNUSET

ALNUSET comprises a set of closely integrated components. In this document we consider the two components of ALNUSET that ITD-CNR has undertaken to develop in the ReMath project, namely the Algebraic Line and the Algebraic Manipulator components. Both components are destined for use by students in lower and upper secondary school (ages from 12-13 to 16/17).

The Algebraic Line component is firstly an operative and representative environment for the construction of mathematical expressions involving numbers and letters defined on a specific numeric set (natural integers, relative integers, rational numbers, rational numbers extended to rational powers) and for the representation of lettering expressions as mobile points on the line.

Secondly, it is an operative and representative environment for the research of the polynomials roots and for the exploration and the definition of the truth sets of algebraic propositions (equations and inequations)

The Algebraic Line is based on a representation which has been built by mathematicians in preceding centuries, the *Numeric Line*. Mathematicians have always used the numeric line as a metaphor of the number systems. As the matter of the fact natural numbers, integer numbers, rational numbers and real numbers are points on a line. Through the inferential geometrical structure of the numeric line it is possible to construct ideas concerning properties of those numbers and of the operations among them. For example, it is possible to construct numerical expressions operating geometrically on the numeric line and observe that these expressions are well defined points on the line denoting numbers.

In the ReMath project we intend to exploit the possibility of visualisation, interactivity, dynamicity and computation offered by technology to transform the numeric line into a *digital algebraic line*, making available a new operative and representative possibility, not available on the numeric line: to use mobile points on the line marked

by letters to express, in general form, the generic element of the considered numeric set. The mobile point on a numeric line is a useful metaphor to conceptualize the notion of algebraic variable and it is a useful operative tool to geometrically construct algebraic lettering expressions on the line and to externalize what they denote.
Hence, mobile points on the lines correspond both to algebraic variables and to algebraic lettering expression depending on those variable. They are important semiotic tools to bring under the control of the perception what the algebraic language indicates in indeterminate way. Moreover, they are at the basis of other algebraic operative and representative possibilities shaped by technology and rooted in the subject's perceptive experience (i.e. research of polynomial roots, exploration of the truth sets for algebraic proposition) that will be presented in section 3.
We do believe that the Algebraic Line can exist only as digital artefact because its algebraic nature is due to the introduction of mobile points on the line and this characteristic is possible only through digital technology. ALNUSET will be the first digital artefact that will embed an Algebraic Line as its component.

The Algebraic Manipulator component is a structured symbolic calculation and multiple representation environment for the manipulation of symbolic algebraic expressions, and for the solution of equations and inequations.
In general, structured symbolic calculation environments are designed for educational activity. They allow the user to act on expressions and equations with step-by-step transformations. These characteristics make such environments quite different from the CAS (Computer Algebra Systems). CAS are designed for professional mathematical activities; they allow the user to handle algebraic transformation with black-box algorithms, producing solutions that in many cases are hard for students to understand.
The Algebraic Manipulator of ALNUSET is designed to allow students to approach algebraic manipulation by constructing meanings well founded in algebraic activity and with a clear theoretical basis. The algebraic manipulator functions of ALNUSET are designed considering both epistemic and pragmatic aspects that can affect algebra learning.
Epistemic aspects concern the role of these functions in constructing an idea of symbolic algebra as an autonomous discipline, where symbolic manipulation is based on the principle of the conservation of formal properties when the numerical structure are extended passing from natural numbers to real numbers.
We note that the pragmatic educational perspective has also been taken into account, with the aim of making the system a useful tool for constructing the technical competencies for subsequent use of a CAS environment.
The Algebraic Manipulator component of ALNUSET has been designed for integrated use with the Algebraic Line component previously described.

2. THE DOMAIN OF ACTIVITIES

The Algebraic Line and Algebraic Manipulator components are designed for integrated use within the educational practice to construct, represent and transform algebraic expressions and to calculate solutions to equations and inequations.
The domain of mathematical expressions that can be constructed and represented on the Algebraic line and/or that can be symbolically transformed through these two components is defined in the following way.
Let E be the set of expressions handled by the Algebraic Line and Algebraic Manipulator components. E is defined by the following rules: All natural integers 0,1,.. are in E; a fixed number of letters, including at least "x", "y", and "z", are in E; if e

and f are in E then e+f, e-f, e/f, e*f are in E, if e is in E and p is a positive natural integer, then e^p and $e^{1/p}$ are in E; if e and f are in E and do not contain the operator <,>,= then e=f, e<f, e>f are in E.
The domain of equations and inequations that are handled by the manipulator are polynomial equation/inequations of one unknown with integer coefficients and degree less than or equal to 4.

2.1 Activities supported by the Algebraic line component

The main activities supported by the Algebraic line component are:

- *The construction of algebraic expressions containing numbesr and letters and their representation as points on the Algebraic line*
- *The externalization of what variables and algebraic expressions depending on them denote;*
- *The identification of real roots of polynomials with integer coefficients and degree ranging from greater than 1 to less than or equal to 4.*
- *The exploration and the identification of the truth set of algebraic propositions*

Construction of new mathematical expressions
One of the main feature of the Algebraic Line is to allow the user to construct new expressions using number, letters and expressions represented on the line.
As a basis for constructing new mathematical expressions, the Algebraic line component makes available a predefined range of integers on the Algebraic line and provides the opportunity to use letters as names of variable points constructed on the Algebraic line.
Integers represented on the Algebraic line, letters corresponding to variable points and mathematical expressions already represented on the Algebraic line can be joined up by means of the geometrical models of mathematical operations in order to create new mathematical expressions.
The component provides a set of tools to graphically construct sums, differences, products, ratios, powers and roots of existing expressions.
Every new mathematical expression constructed in this way is associated to a point on the Algebraic line that indicates the result of the operations performed in sequence.

Some examples of mathematical expressions:

- "2/3" is a mathematical expression produced by combining the integers 2 and 3 with the operation "/". The expression created in this way is associated to a point on the Algebraic line and inserted in the post-it associated to this point. Any equivalent expressions of 2/3 constructed by the user are contained in the same post-it. "a/b" is a mathematical expression produced by combining the letter "a", corresponding to a variable point on the Algebraic line, and the letter "b", corresponding to another variable point on the Algebraic line, with the operation "/". Since the two variable points a and b always occupy a specific position on the Algebraic line corresponding to a decimal value, the expression a/b is represented on the Algebraic line by a point whose position depends on the values of a and b. The expression a/b and its decimal value are contained in the post-it of that point.
- 2*x+1 is a mathematical expression produced through two constructive steps: construction of the expression 2*x by combining the integer 2 and the letter x by means of the operation "*"; construction of the expression 2*x+1 by combining the previous expression 2*x and the integer 1 by means of the operation "+".We note that the expression x+(x+1) is an equivalent expression

of 2*x+1, i.e. it makes reference to the same point on the Algebraic line, and so it will appear in the same post-it of that point.

Externalization of what variables and algebraic expression denote

Once a mathematical expression that contains letters in its structure has been constructed and represented on the Algebraic line, it is possible to drag the variable points on the Algebraic line referring to those letters. When a mathematical expression depends on a variable, the dragging of the variable point on the Algebraic line will refresh the expression depending on it (i.e. moving the "x" point will make the "2*x+1" and "x+(x+1) points move accordingly).

Hence, the dragging of variable points on the Algebraic line is a way of externalizing what a mathematical expression denotes (e.g. in the previous example, the set of odd numbers).

We observe that this is the very characteristics that transform the numeric line into an algebraic line.

The student can exploit this new operative and representative dimension to validate the constructed expression since they allow him to control what the constructed expression denotes.

The drag action can be used to highlight different meanings of the use of letters and of mathematical expressions in algebra.

For example, when the user drags the variable x point to verify what the expression 2x+1 denotes, s/he uses the letter as a means of numerical generalisation and the drag action is a way of reifying the universal quantifier "for each".

When the user drags the variable x point searching for the value(s) for which two points relating to the A(x) and B(x) expressions coincide, s/he uses the letter as an unknown and this meaning is reified in the way s/he uses the drag.

A variable point assumes the meaning of a parameter when the drag is used to instantiate only some specific values of the associated letter with the aim of evaluating the corresponding results of a mathematical expression containing that letter.

Finding the exact roots of a polynomial

One particularly significant activity that can be undertaken using this component of ALNUSET is looking for the exact roots of a polynomial in a variable with integer coefficients and degree ranging from greater than 1 to less than or equal to 4. Consider for example the following polynomial: $P(x) = 18x^3-45x^2+32x-4$. Once this has been represented on the Algebraic line it is possible to seek its exact real roots using the approximate values the user finds by dragging the polynomial variable to a point on the Algebraic line where the polynomial is null or tends to be null.

So in ALNUSET the root of a polynomial can be found by means of a perceptive-motor approach and this activity is structured in such a way as to foster the construction of an appropriate meaning of the notion "root of a polynomial" and thus of "value of the variable whereby the polynomial is null". This characteristic of the artefact helps the user to assume an active role and one that is well focused in epistemological terms with regard to the basic operation underpinning the procedure of factorizing the polynomial in the Algebraic Manipulator component.

Exploration and identification of the truth set of algebraic propositions

The research of the truth set of algebraic propositions, namely the truth set of equations and inequations is a very important activity of the algebra curriculum. The Algebraic Line of ALNUSET makes available specific functions to develop this activity through a constructive and explorative approach in integration with the Algebraic

Manipulator.
The drag of the mobile point characterizing at least one of the two expressions of the proposition is a way to explore the approssimate values of the variable for which the relation between the two expressions of the proposition is respected. The truth set function exploit the results obtained by means of the root function to define the truth set for equation and inequation through a specific interface.

2.2 Activities supported by the Algebraic Manipulator component

The main activities supported by the Algebraic Manipulator component are:

- *symbolic transformation of mathematical expressions*
- *solution of equations and inequations less than or equal to four*

Symbolic transformation of algebraic expressions
Algebraic transformation of mathematical expressions is performed by applying basic manipulation commands which are available via the interface. These commands incorporate symbolic rewriting rules that refer to properties and computation rules of addition, multiplication and powers. The set of algebraic manipulation commands available to the user is open ended: new commands can be added to those already available and used in subsequent manipulations
The ALNUSET manipulator allows the user to control denotation invariance during the manipulation of symbolic expressions. This is achieved thanks to the possibility to coordinate formal manipulation of expressions in the Algebraic Manipulator on the one hand with, on the other, the representation of associated mathematical expressions on the Algebraic line and exploration of the value the manipulator denotes via the dragging of the related variable points.
In this way the user can grasp the fact that the transformed expression denotes the same point on the Algebraic line as the original expression, and is contained in the same post-it associated to that point, along with any other transformation of the variable points on which the original and transformed expressions depend.

Solution of equations and inequations
Equations and inequations can be solved by applying the algebraic manipulation commands previously described as well as commands incorporating the properties of equivalence and inequivalence. The ALNUSET manipulator allows the user to control denotation invariance during the solution of equations and inequations coordinating the symbolic manipulation activity with the representation activity on the Algebraic Line. For example, the user can grasp the fact that the transformation of the equation A(x)=B(x) into C(x)=D(x) on the basis of equality properties does not change denotation, i.e. the true value of the two equalities that depend on the same specific number set X does not change.
This can be carried out:

- In approximate manner, by representing A(x), B(x), C(x), D(x) on the Algebraic line and observing that when dragging the variable point x, the equalities A(x)=B(x) and C(x)=D(x) turn out to be true for the same values of x (for these values of x the point A(x) seems to coincide with B(x) and the point C(x) seems to coincide with D(x) on the Algebraic line)
- In exact manner, by using the manipulator to transform A(x)=B(x) into E(x)=0, by representing the polynomial E(x) on the Algebraic line, by using the specific command to find its roots, and by dragging x and observing that for those values of the roots of E(x) the equalities A(x)=B(x) e C(x)=D(x) are true.

3. THE ALNUSET MENU BAR

Here is the menu bar of ALNUSET, with the list of commands associated to each of the four items available on the bar.

File	Edit	Domain	Components
New	*Undo*	*Natural Integers*	*Algebraic Line*
Open	*Copy*	*Relative Integers*	*Algebraic Manipulator*
Save	*Paste*	*Rational Numbers*	
Save as	*Delete*	*Full Range*	
Print	*Edit*		
Exit	*Redo*		

- The FILE popup menu makes available standard commands typical of applications for the production of documents and objects. We note that ALNUSET provides state input/output as XML files.

- The EDIT popup menu makes available commands for operating on active components. The Copy and Paste commands can be used to import/export mathematical expressions to be shared between ALNUSET and other applications that adopt the MathML format.

- The DOMAIN popup menu allows the user to define the numeric domain s/he wants to operate in. We note that the Full range domain is the extension of rational numbers to rational powers. Choosing the domain effects all the components of ALNUSET (more details are provided in the description of each component).

- The COMPONENTS popup menu allows the user to select the desired component. The Options command will allow the user to specify operating parameters such as language; more details are provided in the description of each component. Each component has its own menu bar with specific commands for that component.

4. THE ALGEBRAIC LINE COMPONENT OF ALNUSET

4.1 The main features of internal representation

- The Algebraic line component manages and represents a set of real numbers in the current set (non-negative integers, integers, rational numbers, rational numbers extended to rational powers).

- This component displays and operates on a set of expressions involving integers and letters (x, y, z,...), for example 3*x+1, 2^(1/3)+1. Internally, each expression is represented as a tree. All variables are defined in the set and have a specific value, so all expressions involving variables can be numerically evaluated, and displayed by the component.

Expressions are displayed on two parallel marked axes. The component displays values in one interval, which the user can zoom/translate. Additional functions can be added to display two or more intervals of the same set of expressions.

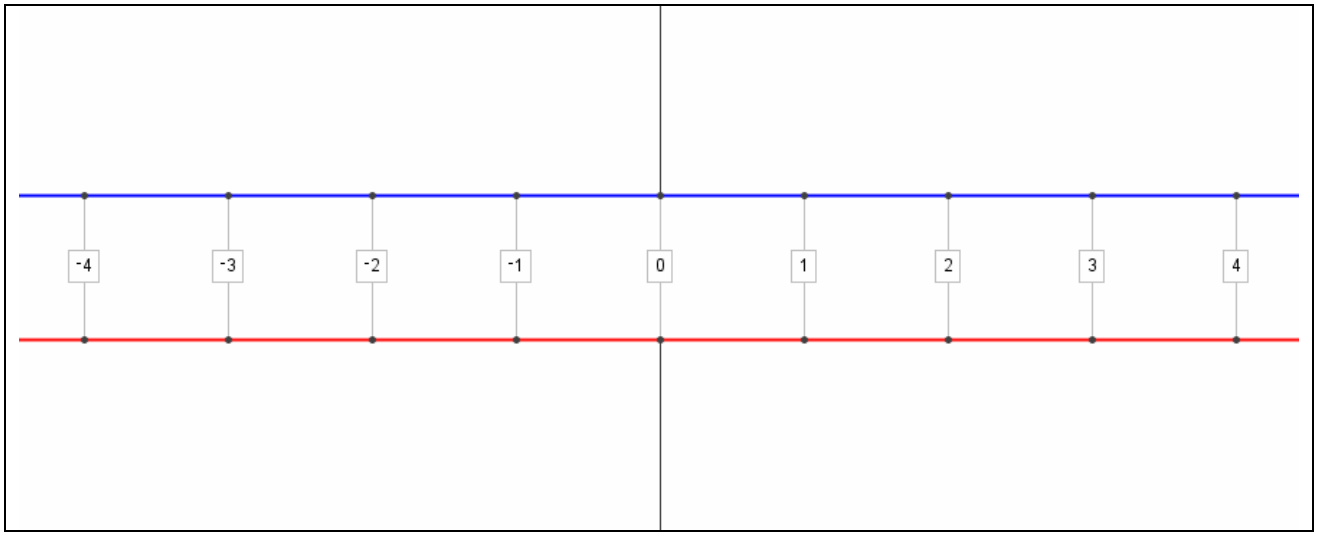
- The component interface allows the dragging of variables (i.e. moving the "x" point to change its numerical value), and will refresh all expressions depending on these (i.e. moving the "x" point will make the "3*x+1" move accordingly). Variables take decimal values corresponding to the pixel <=> real mapping.
  The main feature of the component interface is that it allows construction of new expressions using existing ones. The component provides a set of tools to graphically construct sums, differences, products, ratios, powers and roots of existing expressions. The double lines are used as a support for these constructions, using elementary properties (i.e. Thales' theorem). This provides visual feedback for otherwise abstract algebraic operations.

- Initially, only integer numbers are represented on the line. Subsequently all expressions are based on these numbers, and on the created variables. All expressions are represented in a symbolic way, but the component maintains an approximation of the value of each expression. This is usually computed using hardware double precision arithmetic, and can be extended to higher precision software arithmetic when comparing very close values.

- Expressions can be exported from and imported to the component, for their use by other components of ALNUSET, or by other applications supporting the MathML standard.

4.2 Interface

This is an image of the graphical representation displayed on the screen when the Algebraic Line component is accessed (in the cases of rational or full range domain selection).



- The operative and representative environment offered by this component of

ALNUSET comprises two parallel lines intersected by a perpendicular line. The two points of intersection define the points 0 on the two Algebraic Lines, which are characterized by the same dynamic unit of measure. A label is associated to each point constructed on the Algebraic line reporting the constructed formal expression. If equivalent expressions to one already represented on the line are constructed, a post-it associated to the common point will report all these equivalent expressions

- By activating specific commands, the user can construct and visualize new symbolic expressions on the double lines using already existing expressions. These commands allow the user to graphically construct sums, differences, products, ratios, integers powers and rational powers of existing expressions. Geometrical models embedded in these commands provide visual feedback that can be very useful for assigning a sense to the expression the user is constructing. The dragging of variable points allows the user to take into account of what is concretely denoted by an expression that depends on these variables.

- All the expressions constructed and displayed on the Algebraic Line are also shown in a special space at the bottom of the interface. This space can be used as a navigation aid for searching for and visualizing the points on the Algebraic line associated to each expression. Moreover this space can be used for setting the representation modes of expressions shown on the Algebraic Line.

The main functions available in the Algebraic Line component are:

- Selection of the numerical set
- Zooming
- Scrolling
- Insertion of variable points
- Mathematical operations
  - *Addition /subtraction*
  - *Multiplication/division*
  - *Integer Power /Rational Power*
- Polynomial root
- Truth set
- Editing
- Setting the state and modes of representation on the Algebraic Line
  - *Show/Hide the geometrical construction of the point*
  - *Show/Hide the point*
  - *Delete the point*
  - *Display expressions in two or more intervals.*

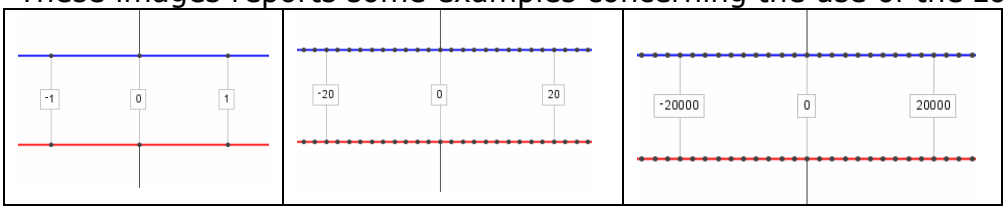
Some of these functions are available on ALNUSET's main menu bar while others can be accessed from the specific bar of this component. The following section describes the interaction features of each function, as well as their associated input and output characteristics. The description of the input concerns the nature of the data acquired by means of the interaction and used by the internal representation and algorithms of the function to produce the output.
The main characteristics of the interaction through which the input is acquired and transformed into an output will also be briefly described.

4.3 Interaction

Selection of the numerical set
This function is available via a command on the Domain popup menu. It allows the user to choose the numerical set (N, Z, Q, Q extended to rational power) in which he/she wants to operate.
Input: A string character corresponding to each numerical set.
Output: A suitable representation of the Algebraic line in accordance with the chosen numerical set.
Interaction : the numerical set is chosen by selecting the corresponding item from the menu with the mouse.

The choice of numerical set modifies how data are displayed on the Algebraic Line: for example, if the domain of natural integers is selected, only positive numbers are displayed. The choice also poses limits on operations that are not closed in the chosen domain and on the movement of any variable point constructed on the line: these aspects are described in detail later.

Zooming
This function allows the user to increase or reduce the scale of representation of the numbers on the Algebraic Line.
Input: a numerical value.
Output: Transformation of the Algebraic line representation according to the numerical value assigned as input.
Interaction: the input of this function can be assigned in two different ways: by dragging point 1 along one of the two straight lines; using the mouse wheel.
These images reports some examples concerning the use of the zoom function



Scrolling
This function allows the user to scroll along the Algebraic line in order to visualise numeric points that are off the screen.
Input: a numerical value.
Output: Scrolling of the Algebraic line represented on the screen and of the points visualized on it.
Interaction: The input of this function can be assigned in two different ways: by dragging the X scroll bar of the window with the mouse; by clicking the right mouse button and dragging along the X dimension of the window.

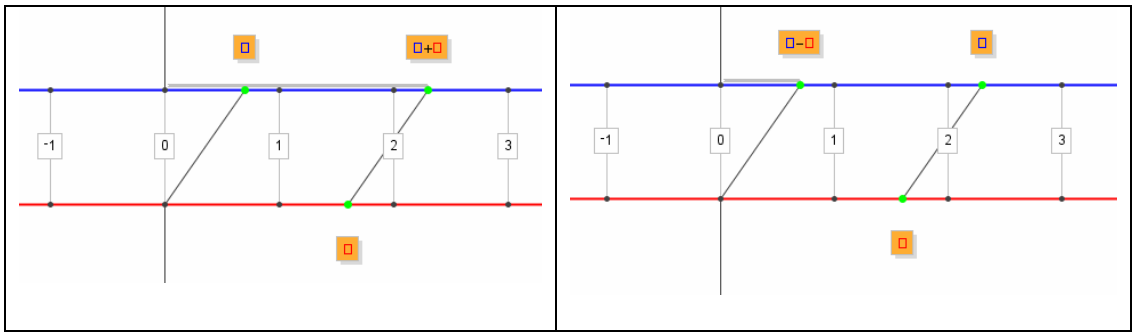
Insertion of variable points
This function allows the user to insert variable points on the Algebraic line and to assign them a name in the form of a letter.
Input: a decimal value to set the point on the Algebraic line and a letter to name that point.
Output: Construction of the point on the Algebraic line and of the associated label containing the point's name and decimal value.

Interaction: The numerical input of this function is assigned by clicking the mouse on a specific pixel on one of the two Algebraic Lines. The variable takes on decimal values corresponding to the selected pixel. At the conclusion of this operation, it is possible to name the point constructed on the Algebraic line by inserting a letter into a field of communication window that appears on the screen close to the constructed point.
Variable points can be dragged along the lines with the mouse in accordance with the restraints imposed by the chosen numerical domain: for example, if the domain of natural numbers has been selected, the variable point can only be dragged over natural numbers.

Mathematical operations
For performing mathematical operations on numbers and expressions represented on the Algebraic Line, three functions are available: Addition/Subtraction, Multiplication/Division, Power/Extraction of root. In each case the user can perform either the direct operation or its inverse by exploiting the geometrical model embedded in the function. These geometrical models constitute the dynamic and interactive interface device through which the symbolic expressions of the numbers can be instantiated for constructing a new expression that indicates the result of the operation performed.
The choice of numerical domain poses limits on operations that are not closed in that domain.

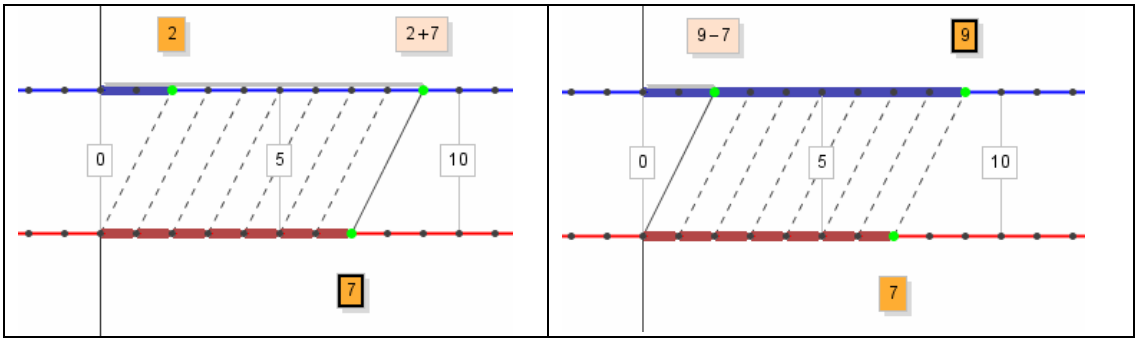
Addition/subtraction
This function allows the user to perform additions and subtractions on numeric expressions represented on the Algebraic Line and to represent the result as a new expression associated to a point on the same line.
Input: Additions and subtractions are binary mathematical operations and need two numerical expressions as input.
Output: Two outputs are associated to this function: (i) A point constructed on the Algebraic line corresponding to the result of the addiction or subtraction performed; (ii) A label associated to the point of the result containing the symbolic expression of the operation performed.

Interaction: The two images below show how the geometrical model for addition and subtraction embedded in this function can be used as an interface device for selecting the operation to be performed (addition or subtraction) and for instantiating the expression to be added or subtracted.



Selection of the operation to be performed (addition or subtraction) depends on the choice of the two points of the model that the user uses to instantiate the expressions. Instantiation of the expressions is performed by dragging the point of the

geometrical model onto the point of the Algebraic line corresponding to the expression one wants to add or subtract.



The choice of numerical domain poses limits on subtraction. In the set of natural integers, the point "a-b" can only constructed for values of b whereby b<=a.

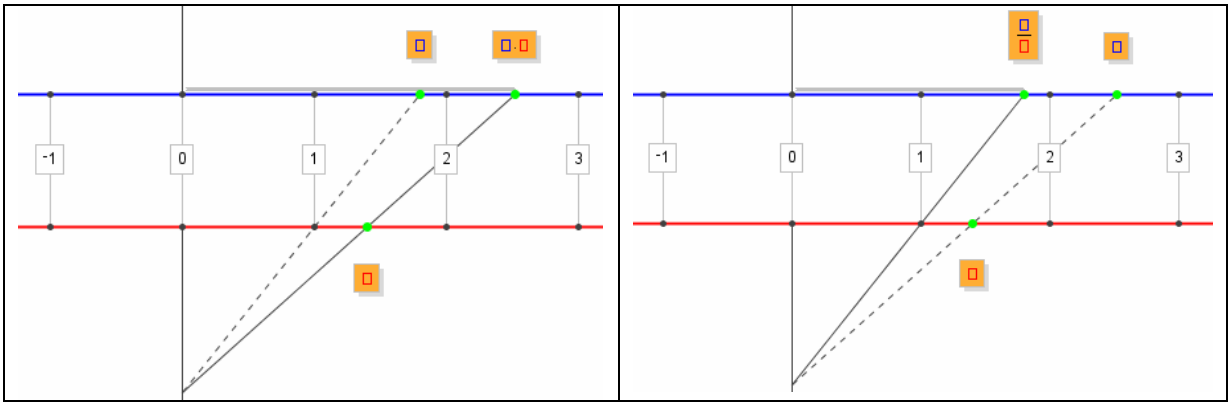
Multiplication / division

This function allows the user to perform multiplication or division on numeric expressions represented on the Algebraic Line and to represent the result as a new expression associated to a point on the same line.

Input: Multiplication or division are binary mathematical operations and need two numerical values as input.

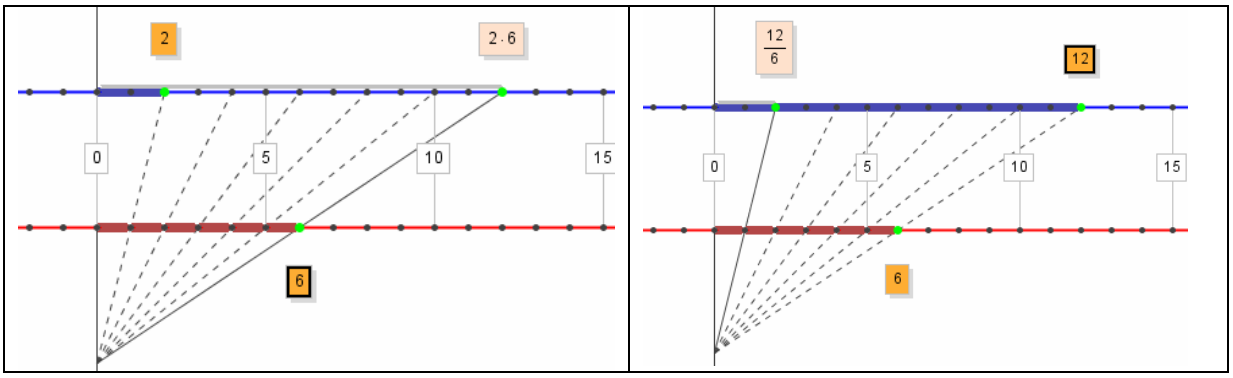
Output: Two outputs are associated to this function: (i) A point constructed on the Algebraic line corresponding to the result of the multiplication or division performed; (ii) A label associated to the point of the result containing the symbolic expression of the operation performed.

Interaction: the two images below show how the geometrical model for multiplication or division embedded in this function can be used as an interface device for selecting the operation to be performed (multiplication or division) and for instantiating the expressions to be multiplied or divided.



Selection of the operation to be performed (multiplication or division) depends on the choice of the two points of the model that the user uses to instantiate the expressions.

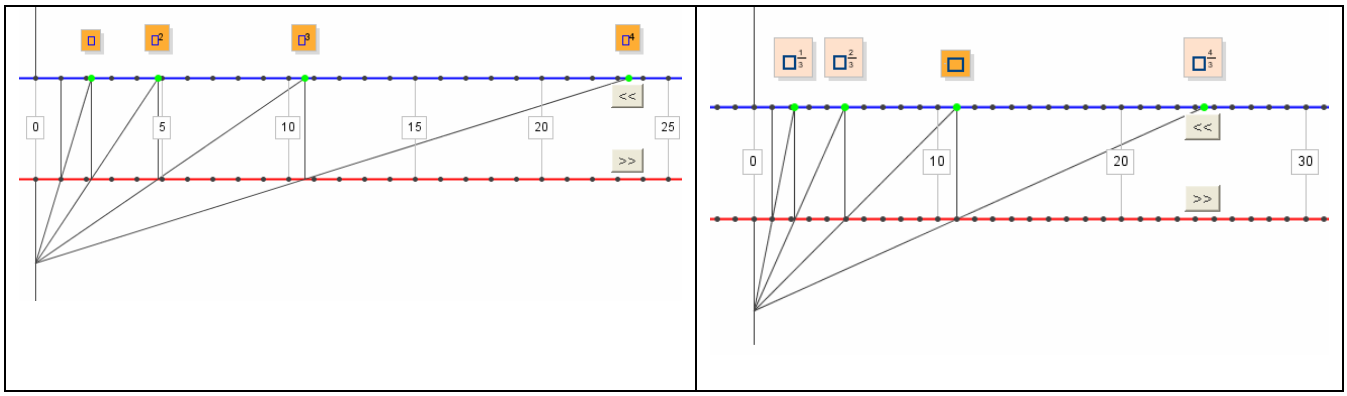
The instantiation of the expressions is performed by dragging the point of the geometrical model onto the point of the Algebraic line corresponding to the expression one wants to multiply or divide.



The choice of numerical domain poses limits on division. In the set of natural and relative integers, the point "a:b" can only constructed for values of a and b whereby the result of a:b belongs the chosen domain.

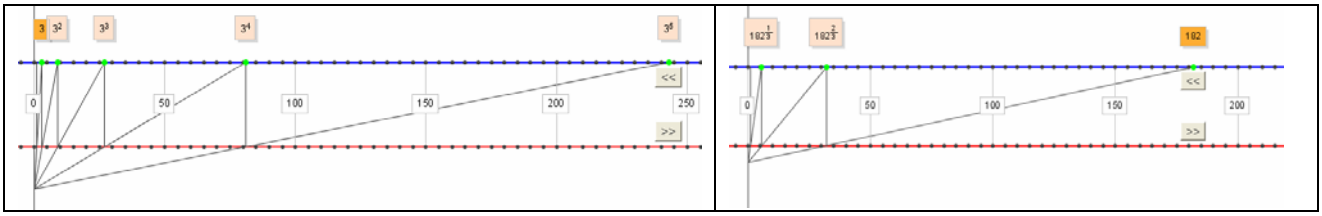
Integer Power/Rational Power.
This function allows the user to perform integer power or rational power of an expression represented on the Algebraic Line and to represent the result of the performed operation on the same line.
Input: Once the grade of the operation is defined, integer power or rational power are unary mathematical operations that need a numerical value as input.
Output: Two outputs are associated to this function: (i) A point constructed on the Algebraic line corresponding to the result of the power or extraction of root performed; (ii) A label associated to the point of the result containing the symbolic expression of the operation performed.

Interaction: The two images below show how the geometrical model for integer power or rational power can be used as an interface device for selecting the operation to be performed (integer power or rational power), for defining its grade and for instantiating the expression on which to perform the selected operation. The grade of the operation can be dynamically increased or decreased by using two specific buttons. Selection of the operation to be performed (integer power and rational power) depends on the choice of the point of the model used to instantiate the expression.



Instantiation of the expression is performed by dragging the point of the geometrical

model onto the point of the Algebraic line corresponding to the expression of which one wants to determine the integer power or the rational power.

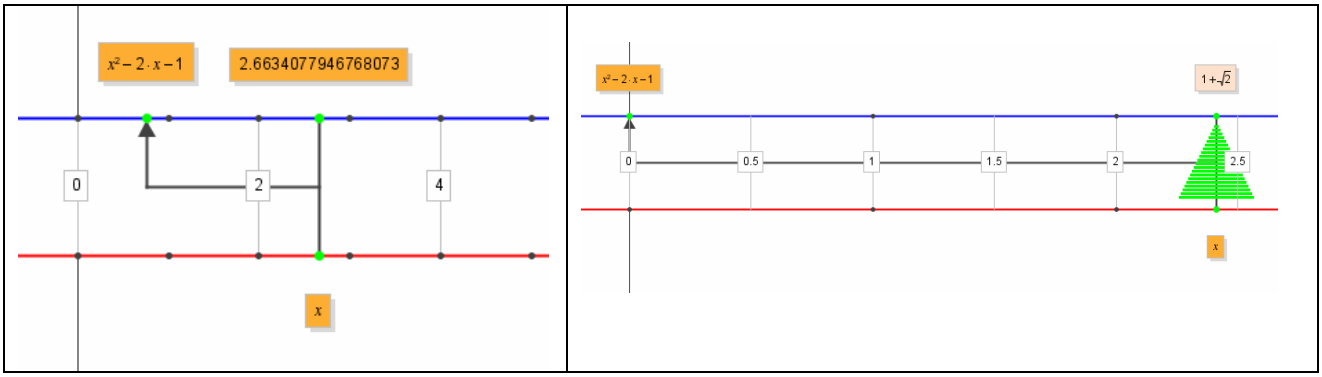


The choice of numerical domain poses limits on the rational power. In the set of natural and relative integers and of rational numbers, the point corresponding to a rational power can only be constructed for numerical values whereby the result of the rational power belongs the chosen domain.

Polynomial root

This function allows the user to find real roots of a polynomial (with integer coefficients and degree ranging from greater than 1 to less than or equal to 4) starting from an approximate value the user has found by dragging the polynomial variable to a point on the Algebraic line where the polynomial is null or tends to be null. The results obtained from this function can be used by the "factorize" function in the Algebraic Manipulator to build the factorized polynomial (see later, in section 5).

Input: the numerical value of the variable point that the polynomial depends on for which the polynomial is null or tends to be null.
Output: exact root of the polynomial represented by a new point on the Algebraic line (if this has not already been constructed) or associated to the post-it of an existing point
Interaction: Communication of the input is completely mouse based. The value of the variable point is assigned by dragging that point



The "root of a polynomial" command can be controlled via a perceptive-motor approach and this activity is structured in such a way as to foster the construction of an appropriate meaning of the notion "root of a polynomial" and thus of "value of the variable whereby the polynomial is null". This characteristic of the command helps the user to assume an active role and one that is well focused in epistemological terms with regard to the basic operation underpinning the procedure of factorizing a polynomial.

Truth set
This function allows the user to explore and to define the truth set of algebraic propositions, namely the truth set of equations and inequations. Once the user has selected this command, he(she) has to define the proposition he(she) wants to solve. To this aim the interface makes available a graphical model on the line to define the two expressions of the proposition and the type of relation between them. The proposition defined in this way by the user is automatically reported in a specific space at the bottom of the interface. Moreover a new line is visualized on the screen between the two already existing lines. The user can drag on this new line the points already represented on the other lines that he considers important points to define the truth set of the proposition. Moreover, in case of inequations the user can define on this line the interval where the defined proposition is "true". This definition is completely mouse based. The action performed by the user on the line (dragging of points, definition of intervals) are automatically expressed in formal language by the system and reported in the specific space containing the proposition. A specific command reported in this space allow the user to know by the system whether his solution is correct and complete.

Input: The two expressions of the proposition, points and interval on the line that define the truth set
Output: The expression of the truth set of a proposition in formal language in a specific window
Interaction: Communication of the input is completely mouse based.

Editing
This function allows the user to edit an expression s/he wants to represent on the number line. The point corresponding to that expression is automatically constructed on the Algebraic line and the corresponding expression is inserted into the label associated to that point. This function allows the user to represent expressions on the Algebraic line without directly using the geometrical models of the operations.
Input: the string corresponding to the expression one wants to represent on the Algebraic line.
Output: construction of a point on the Algebraic line associated to the symbolic edited expression and insertion of the expression in the label of that point.
Interaction: the string is edited in a specific edit space.

Setting the state and modes of representations on the Algebraic Line
All the expressions constructed and displayed on the Algebraic Line are also displayed in a special window that can be used as a navigation window when searching for and visualizing the points on the Algebraic line associated to each expression. For each expression represented on the lines and displayed in this window it is possible:
Show/Hide the geometrical construction of the point
Show/Hide the point
Delete the point
Display points in two or more intervals.
These functions are available clicking on the right button of the mouse when its pointer is on the point whose the user wants to change the state or on the correspondent expression reported in the window

Show/Hide a geometrical construction
The geometrical construction of a point is displayed on the screen until the conclusion of the construction process. The show/hide check box makes it possible to visualise or hide the geometrical construction (default: hide).

Show/Hide a point
This function makes it possible to show or hide points constructed on the Algebraic Line. This function can be useful for hiding points that are not necessary for the activity at hand or that can be visually confusing when seeking to observe and interpret the mathematical phenomena involved (default: show).

Delete the point
This function allows the user to cancel a point that has been constructed on the Algebraic Line.

Display expressions in two or more intervals
Expressions are displayed on the parallel lines in one interval, and functions for zoom/translation of this interval are available by means of interface commands.
When variable points are moved on the Algebraic line, the point corresponding to the related symbolic expression may not be visualized in the same interval within the screen window.
This modality of representation allows the user to maintain multiple expressions on the screen by displaying these expressions in multiple intervals on the Algebraic line (default: one interval; Max. number of intervals: three)

Input: The point or the expression reported in the specific window whose the user wants to change the state.
Output: Setting the state and modes of representations on the Algebraic Line
Interaction: Communication of the input is completely mouse based.

5. THE ALGEBRAIC MANIPULATORCOMPONENT OF ALNUSET

5.1 The main features of the internal representation

This component manages algebraic manipulations of a single expression and the solution of equations and inequations of degree less than or equal to 4.

The interface provides a set of fundamental commands for symbolic manipulation that is specific to the numeric domain chosen by the user via the Domain popup menu (Natural integers, Relative integers, Rational numbers, Full range domain).

The basic commands for algebraic transformation have been designed to allow the user to control each step in the algebraic transformation solution process. The commands are related to a system of operations properties and calculation rules; if they are applied to one part of an expression that can be suitably manipulated with those commands, then the entire expression is transformed in accordance with the embedded transformation rules. Hence manipulations are performed under the control of the user, using a set of symbolic rewriting rules that refer to the basic properties of the operations (for example a+b ←→ b+a: commutative property of addition) and to specific rules of computation (for example n → p_1*……*p_k: Prime factor decomposition).

The symbolic rewriting rule associated to each command generates an expression that is equivalent to the original one.

The set of algebraic manipulation commands available to the user is open ended: new commands can be added to those already available and used in subsequent manipulations. For example, once a^2-b^2 in (a+b)*(a-b) has been transformed using the basic commands, this outcome can be converted automatically into a new symbolic rewriting command. It should be noted that construction of a new rule is strictly related to the manipulation that has just been performed: the definition of the new symbolic rewriting command is under the full control of the system. The user can only decide whether or not to create a new command.
The ALNUSET manipulator allows the user to control denotation invariance during the manipulation of symbolic expressions. This is achieved thanks to the possibility to coordinate formal manipulation of expressions in the Algebraic Manipulator on the one hand with, on the other, the representation of associated mathematical expressions on the Algebraic line and exploration of the value the manipulator denotes via the dragging of the related variable points.

The expressions are represented internally using the same tree as the Algebraic line component. The Algebraic Manipulator displays all steps of the algebraic manipulation performed, from the initial expression to the current one. The interface allows selection of a sub-expression of the current expression, and then the application of a symbolic rewriting rule so as to obtain the new expression. This guarantees the equivalence of all expressions.

Equations and inequations are manipulated using the same set of symbolic rewriting rules for manipulation already described and a set of symbolic rewriting rules that are specifically related to the properties of equalities and inequalities.

Here too, expressions can be exported from and imported to the component, to be used by the other component of ALNUSET, or by other applications supporting the MathML standard.

5.2 Interface

The Algebraic Manipulator component is available in a special window via a command on the COMPONENTS popup menu bar. The interface comprises the bar of commands for manipulation activity and the working space in which manipulation is carried out.

The commands structure is constituted by:

- Command for editing the text of the symbolic expression to be transformed or the equation/inequation to be solved.
- Basic commands incorporating symbolic rewriting rules related to the properties of operations and of specific computation rules
- Command for turning the manipulation that the user has just performed into a new symbolic rewriting rule
- Algebraic manipulation commands created by the user
- Commands for solving equations/inequations.

The manipulation working space is a field that can be scrolled both horizontally and vertically.
The four tables below (Tab. 1, Tab. 2, Tab. 3, Tab. 4) list the set of basic commands available for symbolic manipulation within the various numerical domains.

TAB. 1: Rules for Natural Integers		
a, b, c are numbers or expressions with numbers and letters defined in this domain. n, p, q are natural integers		
N1	$a+b \leftrightarrow b+a$	Addition commutativity
N2	$(a+b)+c \leftrightarrow a+(b+c)$	Addition associativity
N3	$a+0 \leftrightarrow a$	Addition neutral element
N4	$a-a \leftrightarrow 0$	Definition of subtraction
N6	$a*b \leftrightarrow b*a$	Multiplication commutativity
N7	$(a*b)*c \leftrightarrow a*(b*c)$	Multiplication associativity
N8	$a*1 \leftrightarrow a$	Multiplication unit element
N9	$a*0 \leftrightarrow 0$	Multiplication zero element
N10	$a*(b_1 + \ldots + b_k) \leftrightarrow a*b_1 + \ldots + a*b_k$	Distributivity
N11	$a^n \leftrightarrow n*\ldots..*n$	Definition of power
N12	$a^0 \leftrightarrow 1$	Extension to power 0
N13	$a^p * a^q \leftrightarrow a^{p+q}$	Composition of powers with the same basis
N14	$n \rightarrow p+q$	Decomposition (p obtained by user input)
N15	$n \rightarrow p_1*\ldots\ldots*p_k$	Prime factor decomposition
N16	$p : p \leftrightarrow 1$	Definition of division
N17	$p_1 + \ldots\ldots + p_k \rightarrow n$	Sum
N18	$p_1 * \ldots\ldots * p_k \rightarrow n$	Product
N19	$expr \rightarrow a$	Expression evaluation
N20	P(X), a root of $P \leftrightarrow (X-a)*Q(X)$	Polynomial factorisation using a known root

Tab. 2: Rules for the Relative Integers		
a, b, c are numbers or expressions with numbers and letters defined in this domain. n, p, q are natural integers		
Z1	$a+b \leftrightarrow b+a$	Addition commutativity
Z2	$(a+b)+c \leftrightarrow a+(b+c)$	Addition associativity
Z3	$a+0 \leftrightarrow a$	Addition neutral element
Z4	$a + {}^{-}a \leftrightarrow 0$	Addition opposite
Z5	$a*b \leftrightarrow b*a$	Multiplication commutativity
Z6	$(a*b)*c \leftrightarrow a*(b*c)$	Multiplication associativity
Z7	$a*1 \leftrightarrow a$	Multiplication unit element
Z8	$a*0 \leftrightarrow 0$	Multiplication zero element
Z9	$a*(b_1 + \ldots + b_k) \leftrightarrow a*b_1 + \ldots + a*b_k$	Distributivity
Z10	$a^n \leftrightarrow n*\ldots..*n$	Definition of power
Z11	$a^0 \leftrightarrow 1$	Extension to power 0
Z12	$a^p * a^q \leftrightarrow a^{p+q}$	Composition of powers with the same basis
Z13	$n \rightarrow p+q$	Decomposition (p obtained by user input)
Z14	$n \rightarrow p_1*\ldots\ldots*p_k$	Prime factor decomposition
Z15	$p : p \leftrightarrow 1$	Definition of division
Z16	$a-b \leftrightarrow a+{}^{-}b$	Definition of subtraction using opposite
Z17	${}^{-}a \leftrightarrow ({}^{-}1) *a$	Link between opposite and ${}^{-}1$ (sign factor)
Z18	$({}^{-}1) * ({}^{-}1) \leftrightarrow 1$	Sign cancellation
Z19	$p_1 + \ldots\ldots + p_k \rightarrow n$	Sum
Z20	$p_1 * \ldots\ldots * p_k \rightarrow n$	Product
Z21	expr → a	Expression evaluation
Z22	P(X), a root of P $\leftrightarrow$ (X-a)*Q(X)	Polynomial factorisation using a known root

Tab. 3: Rules for the Rationals Numbers		
a, b, c are numbers or expressions with numbers and letters defined in this domain. n, p, q are natural integers		
Q1	$a+b \leftrightarrow b+a$	Addition commutativity
Q2	$(a+b)+c \leftrightarrow a+(b+c)$	Addition associativity
Q3	$a+0 \leftrightarrow a$	Addition neutral element
Q4	$a + {}^{-}a \leftrightarrow 0$	Addition opposite
Q5	$a*b \leftrightarrow b*a$	Multiplication commutativity
Q6	$(a*b)*c \leftrightarrow a*(b*c)$	Multiplication associativity
Q7	$a*1 \leftrightarrow a$	Multiplication unit element
Q8	$a*0 \leftrightarrow 0$	Multiplication zero element
Q9	$a*(b_1 + \ldots + b_k) \leftrightarrow a*b_1 + \ldots + a*b_k$	Distributivity
Q10	$a^n \leftrightarrow n*\ldots..*n$	Definition of power
Q11	$a^0 \leftrightarrow 1$	Extension to power 0
Q12	$a^p * a^q \leftrightarrow a^{p+q}$	Composition of powers with the same basis

Q13	$1/a^p \leftrightarrow a^{-p}$	Equivalence between opposite power and inverse
Q14	$n \rightarrow p+q$	Decomposition (p obtained by user input)
Q15	$n \rightarrow p_1*\ldots\ldots*p_k$	Prime factor decomposition
Q16	$a-b \leftrightarrow a+^{-}b$	Definition of subtraction using opposite
Q17	$^{-}a \leftrightarrow (^{-}1) *a$	Link between opposite and $^{-}1$ (sign factor)
Q18	$(^{-}1) * (^{-}1) \leftrightarrow 1$	Sign cancellation
Q19	$a*1/a \leftrightarrow 1$	$(a \neq 0)$.Multiplication inverse
Q20	$a/b \leftrightarrow a*1/b$	Definition of general fraction
Q21	$1/a * 1/b \leftrightarrow 1/ab$	Product of two inverses
Q22	$p_1 + \ldots\ldots + p_k \rightarrow n$	Sum
Q23	$p_1 * \ldots\ldots * p_k \rightarrow n$	Product
Q24	expr → a	Expression evaluation
Q25	P(X), a root of P$\leftrightarrow$ (X-a)*Q(X)	Polynomial factorisation using a known root

Tab. 4: Rules for rational numbers extended to rational powers (Full Range)		
a, b, c are numbers or expressions with numbers and letters defined in this domain. n, p, q are natural integers		
P1	$a+b \leftrightarrow b+a$	Addition commutativity
P2	$(a+b)+c \leftrightarrow a+(b+c)$	Addition associativity
P3	$a+0 \leftrightarrow a$	Addition neutral element
P4	$a + ^{-}a \leftrightarrow 0$	Addition opposite
P5	$a*b \leftrightarrow b*a$	Multiplication commutativity
P6	$(a*b)*c \leftrightarrow a*(b*c)$	Multiplication associativity
P7	$a*1 \leftrightarrow a$	Multiplication unit element
P8	$a*0 \leftrightarrow 0$	Multiplication zero element
P9	$a*(b_1 +\ldots + b_k) \leftrightarrow a*b_1+ \ldots + a*b_k$	Distributivity
P10	$a^n \leftrightarrow n*\ldots..*n$	Definition of power
P11	$a^0 \leftrightarrow 1$	Extension to power 0
P12	$a^b * a^c \leftrightarrow a^{b+c}$	Composition of powers with the same basis
P13	$1/a^b \leftrightarrow a^{-b}$	Equivalence between opposite power and inverse
P14	$n \rightarrow p+q$	Decomposition (p obtained by user input)
P15	$n \rightarrow p_1*\ldots\ldots*p_k$	Prime factor decomposition
P16	$a-b \leftrightarrow a+^{-}b$	Definition of subtraction using opposite
P17	$^{-}a \leftrightarrow (^{-}1) *a$	Link between opposite and $^{-}1$ (sign factor)
P18	$(^{-}1) * (^{-}1) \leftrightarrow 1$	Sign cancellation
P19	$a*1/a \leftrightarrow 1$	$(a \neq 0)$.Multiplication inverse
P20	$a/b \leftrightarrow a*1/b$	Definition of general fraction
P21	$1/a * 1/b \leftrightarrow 1/ab$	Product of two inverses
P22	$p_1 + \ldots\ldots + p_k \rightarrow n$	Sum
P23	$p_1 * \ldots\ldots * p_k \rightarrow n$	Product
P24	expr → a	Expression evaluation
P25	P(X) and a root of P, P(X)$\leftrightarrow$ (X-a)*Q(X)	Polynomial factorisation using a known root

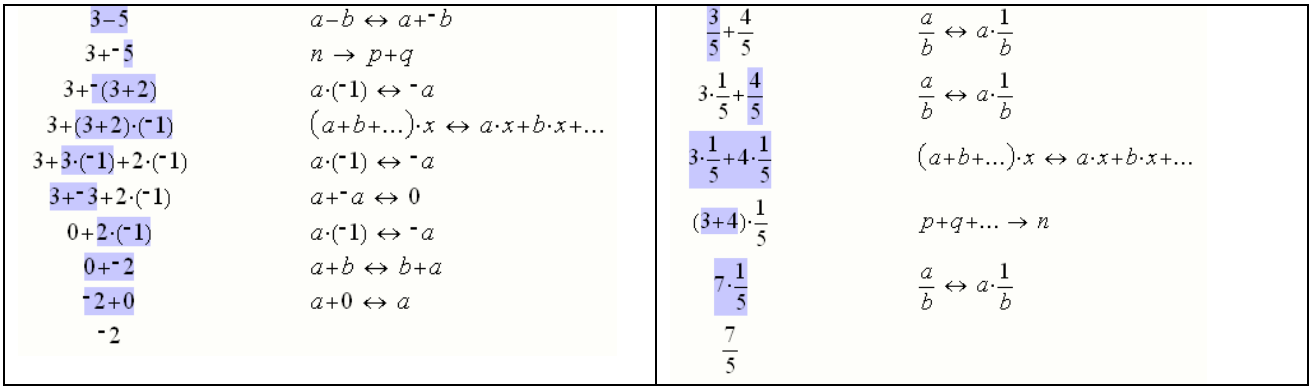
Table 5 lists commands available for solving equations/inequations.

Tab.5: Rules for equations and inequations solution

a, b, c are numbers or expressions with numbers and letters defined in the chosen numeric domain. n is natural integers		
E1	a = b ←→ a + c = b + c, ∀ c	Addition property of equality
E2	a= b ←→ a * c = b *c, ∀ c #0	Multiplication property of equality
E3	a<b ←→ a+c<b+c ∀ c	Addition property of inequality
E4	a<b ←→ a*c < b*c ∀ c, c>0	Multiplication property of inequality
E5	a<b ←→ a*c > b*c ∀ c, c<0	Multiplication property of inequality
E6	a<b ←→ $a^n<b^n$ ∀ a, b; ∀ n ε N , n odd	Power property of inequality
E7	O<a<b ←→ $a^n<b^n$ ∀ a, b; ∀ n ε N, n even	Power property of inequality

Observations

The command "expr → a" (N19, Z20, Q24, P24) provides a level of computation that can only be accessed by selecting the Options command in the COMPONENTS popup menu. The aim is to make available different operational levels of computation which may prove beneficial for learning. Without this command it would be necessary to undertake a long series of steps even to tackle the simplest of calculations, such as those shown on the example below.



This is particularly useful in the approach to calculation with relative and rational numbers, since it entails referring to basic properties of symbolic manipulation and computation for proving and framing, from a theoretical viewpoint, the more immediate and automatic calculation procedures involved with these number sets. However, once the students have gained experience with these procedures and they have grasped and interiorized a certain general computation rule, a more advanced level of computation could be proposed.
The command "expr → a" allows the student to obtain the result of the computation of numerical and lettering expressions.

The command P(X)←→ (X-a)*Q(X) makes it possible to factorize polynomials of 2nd, 3rd, 4th degree in a variable and with integer coefficients on the basis of its root "a" found in the Algebraic component. We note that the factorize command is available in all Computer Algebra Systems. These systems calculate polynomial roots via complex series of internal procedures that are not transparent for the user. In these systems the factorize command can be considered as a black-box command. By contrast, in

the ALNUSET symbolic manipulator, the factorize command draws on the results obtained with the "polynomial root" command available in the Algebraic Line component for performing polynomial factorization. We recall that this command allows the student to determine the exact real root of a polynomial using the approximate root value s/he has already found by dragging the polynomial variable to a point on the line where the polynomial becomes, or tends to become, null.
Let's examine how the factorize command works in ALNUSET. If the student has already found the root of the polynomial *P(x*) in the Algebraic Line component and wishes to factorise it, s/he applies the factorize command to it in the Algebraic Manipulator and obtains the factorisation based on the original root.
This command plays a crucial role in solving equations and inequations above the first degree.

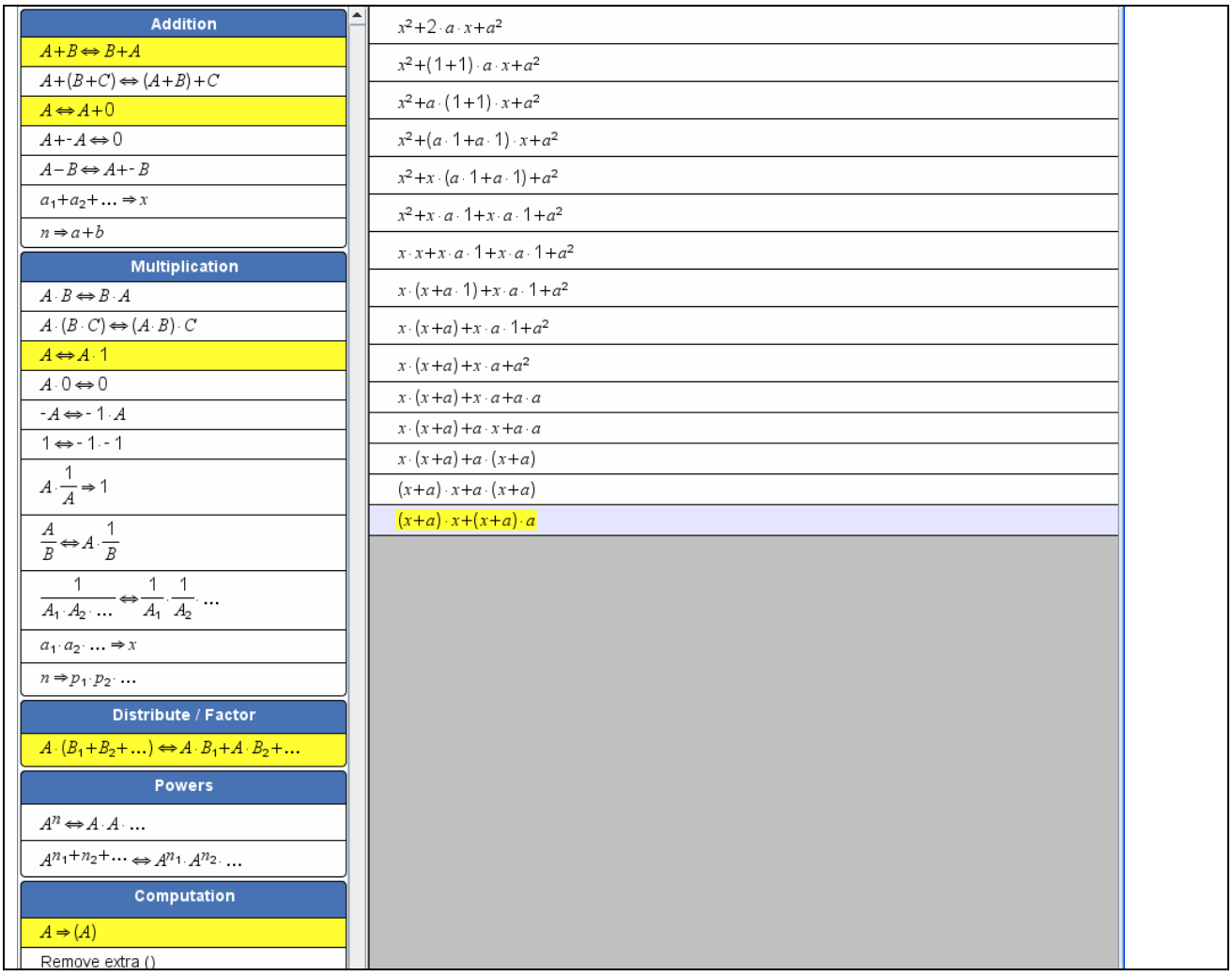
5.3 Interaction

Activities in the Algebraic Manipulator begin with the student in putting the text of the expression s/he wishes to transform. A special Edit command is available from the EDIT popup menu for editing the algebraic expression to be manipulated: this is done in a special window with a field for sequential and linear editing of the expression, equation or inequation. The system checks that the edited object is congruent and provides relevant feedback. Symbolic expressions generated in the Algebraic Line component or in other applications based on the MathML standard can be imported into the window using the Paste command.

Once the symbolic representation of the algebraic object has been edited for symbolic manipulation, it is represented in the window in standard algebraic format.
Symbolic manipulation is performed using two actions:

- Selection of the part of the algebraic object to be manipulated (mouse click and drag);
- Execution of the command.

The part of the object selected for manipulation is highlighted and comes under the system's control. The student can only select parts of the symbolic representation that are compatible with the hierarchical organisation of its structure.
The system is designed to reveal the hierarchical structure of a symbolic representation. When the mouser pointer is positioned over any part of the object, the system dynamically displays all the elements in the hierarchical structure, beginning with the lowest level elements up to those at the highest level.
When a part of the symbolic representation is selected for manipulation, only those commands that can be applied to that part are active and accessible.
The figure below illustrates the interaction. It can be noted that, in accordance with the current selection, only some commands are active.



This example also illustrates what happens when a command incorporating a symbolic rewriting rule is applied to the selected part of the symbolic representation: the rule that has been applied is displayed beside the transformed expression.
Once the algebraic manipulation proving that expression E is equivalent to expression F has been completed, a command is available for:

- creating a new symbolic manipulation command associated to the rule E ←→ F. This new command is immediately added to the list of user-created commands available in a specific window;
- saving the new symbolic manipulation command in a personal file along with all the other new commands the user has created.

At the beginning of a work session, the user can load his/her personally generated commands via the "Import personal commands" item in the program's Options menu.

6. TECHNICAL FEATURES

ALNUSET will be developed in Java as a stand alone application
A web enabled version of the ALNUSET components as a Java applet may be realized, depending on the necessity of the ReMath project.
Internally, each component is a Java class derived from the Swing JComponent class. This allows inclusion in any Java applet (provided by a remote web server) or

Java application (running on the client), independent of the platform used (the only requirement is an up-to-date Java Runtime Environment for the platform).
ALNUSET will provide state input/output as XML file.
Specific communication functions may be incorporated in ALNUSET to handle the downloading and the uploading of its XML file from/to the MathDiLS platform
ALNUSET makes possible the import/export of mathematical expressions among its components and from/to other applications in MathML format.

DDA 4: Machine Lab

Authors: Psycharis, Markopoulos, Kyrimis, Latsi, Papadopoulou, Antoniou, Alexopoulou

National and Kapodistrian University of Athens, Educational Technology Lab

Work progress in 2006

During the first year of ReMath, MachineLab has been developed as a stand-alone application developed in Borland Delphi, using the 3Impact game engine (http://www.3impact.com), and runs under Microsoft Windows. In the version that we will have at the end of November, parts of MachineLab will have been ported to Java, for easier maintenance, using ActiveX and JNI to communicate with the native parts of the application. Because of the dependency on Windows native code, the application will remain Windows-specific. MachineLab accepts, as input, scripts written in Logo or PascalScript. When executing a Logo script, it is translated to PascalScript, which is what is actually executed. In the version that we will have at the end of November, we will have replaced PascalScript with Java, which will be compiled using either the compiler provided by Sun in the Java Development Kit, or the free compiler Jikes (http://jikes.sourceforge.net/).

By the end of November we will have:

- *Turtle*. 3D turtle following Logo primitives.
- *Turtle Scene (TS).* Three-dimensional interface.
- *Logo Editor*. Logo-like programming interface linked to the TS.
- *Logo-commands*. Extension of Logo-like commands in 3D space.
- *Uni-dimensional Variation Tool (1dVT).*
- *Turtle's trace.* It will be a selectable, 3D object (e.g., a thin cylinder).
- *Selection of points in 3d space.* The user will be able to select the trails that the turtle draws when it moves.
- *Re-execution of Logo procedures* by manipulating the sliders of the 1dVT.

Remain to be developed:

- Two-dimensional Variation Tool (2dVT).
- Vector Variation Tool (VVT).
- Vector operations.
- Cartesian and polar coordinates.

The above tools/functionalities will be developed by the end of month 16.

MachineLab specifications of the first version

1. Introduction

Machine-Lab is a programmable environment for the creation and exploration of interactive virtual reality simulations which within the ReMath project will be designed as microworlds. The main part that has beeen developed in the first version of MachineLab consists of one microworld for vectors in geometry based on Logo as a programming language. We call this environment **"MachineLab Turtleworlds" (MaLT)**. It has been based on a 3D Turtle Geometry system combining Polar and Cartesian coordinates. The representational infrastructure provided by Machine-Lab has been specially designed to afford representations as tools to manipulate variation of variable values, to navigate agents in space, to define their properties and behaviors.

2. Mathematical domain

We have chosen the notion of vector as a means to represent the link between 2D and 3D representations, since vectors can be considered as basic components underpinning the study of geometry and motion in space facilitating the study of 3D spatial thinking. In MaLT the mathematical nature and identity of vectors is planned to integrate with their use to control and measure geometrical figures and virtual phenomena.

In Malt, vectors will be considered as embedded in turtle's movement which takes place in a specially designed 3D Turtle Geometry Environment where cartesian and polar geometrical systems co-exist. The user will be able to capitalise upon the power of Logo as a programming language to drive the turtle in body-syntonic and at the same time mathematical formal way and to observe the graphical output. The turtle's state after a move (concisting of its position and heading) will be described through numbers that at the same time define a vector.

The mathematical concepts of reference for MachineLab are thus:

(a) the notion of vector as an object, as a set of properties, and as a representation of a vectorial measure in 3D space in either Cartesian and Polar coordinate system.

(b) the notion of mathematical variation and its manipulation related to the representation of vectors in simulating and controlling move in 3D space (e.g. construction and transformation of geometrical solids).

The meaningful activities supported by the MachineLan will involve the vector representations and its manipulations in

- controlling move in 3D space
- constructinge and transforming geometrical figures.

3. MaLT specifications

MaLT :

- is an open learning environment that enables learning by constructing and designing
- enables multiple perspectives and representations for mathematical meaning-making
- features facilities such as Logo-like programming language and dynamic manipulation means that enable the expression of mathematical ideas.

In MaLT, vectors operate as one of the tools to dynamically manipulate the variable values of procedures of Logo programming language which in turn operates as a hybrid between Turtle Geometry Environments (TGE) and Dynamic Geometry Environments (DGE). In fact the entire geometrical environment of MaLT is about variables, change and rate of change.

The first version of MaLT consists of three components (Figure 1) namely:

- *Turtle Scene (TS).* The main part of this component is a three-dimensional interface. It will provide rich representations of geometric objects and behaviours allowing navigation within the 3D virtual space.

- *Logo Editor (LE).* This component is the Logo-like programming interface that engages students in logical procedural thinking and it is linked to the TS. The user is able to write, run and edit Logo programs.

- *Variation Tools*. This component provides means to handle and represent variation of variable values in MaLT. In the present version of MaLT it consists of the *Uni-dimensional Variation Tool (1dVT).* The main part of this component consists of 'number line'-like sliders, each corresponding to one of the variables used in a Logo procedure.

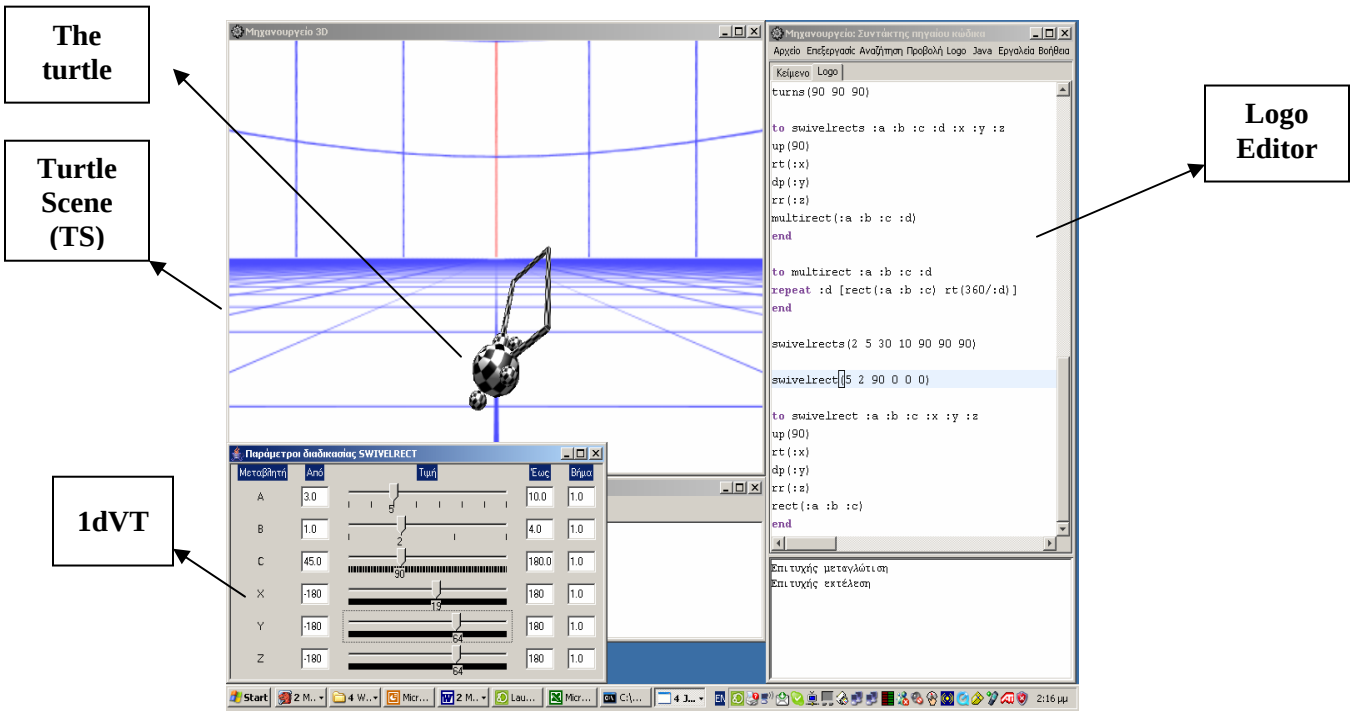


Figure 1: The main part of the first version of MaLT.

3.1 Objects – Entities

Turtle's trace

In TS there is a 3D turtle following Logo primitives (e.g., `forward 10`). Whenever the turtle moves it leaves behind it a trace (see TS in Figure 1). This trace is selectable, three-dimensional object, e.g., a thin cylindrical line. This object corresponds to the line drawn by the turtle in standard, 2D Logo. In a similar manner to 2D Logo, the trace is only drawn when the turtle's pen is "down". The trace is not a 3impact "trail", because of its limitations: limited number of segments, no control of what each segment is, and inability to select individual segments.

Vectors

Vector as an entity embodied in the 3D Turtle Geometry. In MaLT vectors are considered to be embodied in turtle's movement which takes place in TS. In 2D Turtle Geometry (TG), a Polar and a Cartesian geometrical system co-exist. This is manifested by the available Polar primitives (the absolute `setheading` command and the usual relative `forward`, `back`, `right`, `left` commands) and Cartesian primitives related to the `setposition` command. The Polar system essentially embodies differential

geometry, in the sense that the turtle is a mathematical entity with a state consisting of its position and heading. The Logo language is the driver of the mathematical entity and consists of commands through which the position and heading change. Position changes are accompanied by a linear trace which remains on the screen at all times. These basic state-changing commands can be given in any combination and sequence through structured variable procedures of the Logo programming language.

The language itself is functional, procedural and recursive with manifest data types, and its syntax is aligned with the syntax of mathematical formalism. The state of the turtle 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. From the definitions above, it follows that any change in the turtle's state represents a vector. In the case where the turtle has not moved, we have a vector with zero length. In this sense, the trace of any turtle move can be perceived as the tangent to a mathematical curve which is the best fit into a series of turtle moves and, in this way, the difference between two consecutive vector inclinations represents the curvature (the differential) as the length of the vector tends to zero. It is in this sense that Turtle Geometry is perceived as a 'Geometry of vectors' and it operates as the basis for the representation and functionality of vectors in MaLT.

A vector in MaLT -embodied in turtle's behavior- is thus considered as an *oriented journey* from one (initial) point to another, allowing dynamic manipulation and programmed behavior.

Logo procedures

Logo procedures written and run in the Logo Editor are immediately visualized in TS. The programming language in MaLT inherits the power of Logo that allows users to define variables and procedures. By using procedures students will be able to create new primitives and recursive procedures.

Logo commands

- The Logo commands that have been developed are the following:

PRINT`(java.lang.Object... x)`
Print a list of items.

TYPE`(java.lang.Object... x)`
Print a list of items.

INTEGER`(java.lang.Object n)`
Returns its argumnent cast to an integer.

INT(java.lang.Object n)
Returns its argumnent cast to an integer.

VARIANT(java.lang.Object n)
Returns its argumnent cast to an object

CHR(java.lang.Object n)
Returns its argumnent cast to a character.

STRING(java.lang.Object s)
Returns its argumnent cast to a string.

COUNT(java.lang.Object x)
Count the number of items in an object.

LENGTH(java.lang.Object x)
Count the number of items in an object.

BUTFIRST(java.lang.Object x)
Returns all but the first item of an object.

BUTLAST(java.lang.Object x)
Returns all but the last item of an object.

FIRST(java.lang.Object x)
Returns the first item of an object.

LAST(java.lang.Object x)
Returns the last item of an object.

ITEM(java.lang.Object i, java.lang.Object x)
Returns the i-th item of an object.

ISLIST(java.lang.Object x)
Checks if an object is a list.

ISARRAY(java.lang.Object x)
Checks if an object is an array.

NOT(java.lang.Object x)
Checks if an object is false.

FPUT(java.lang.Object x, java.lang.Object list)
Add an object at the front of an item.

LPUT(java.lang.Object x, java.lang.Object list)
Add an object at the end of an item.

ITEMPUT(java.lang.Object n, java.lang.Object list, java.lang.Object x)
Replace the n-th element of an item.

ARRAY(java.lang.Object n)
Create an array.

ARRAYCOUNT(java.lang.Object x)
Returns the size of an array or list.

ARRAYGET(java.lang.Object list, java.lang.Object n)
Return the n-th element of an array or list.

ARRAYPUT(java.lang.Object list, java.lang.Object n, java.lang.Object x)
Set the n-th element of an array or list.

WAIT(java.lang.Object t)
Delays further execution for a specified amount of time.

SETCALLBACKSENABLED(java.lang.Object enable)
Enables or disables callbacks.

ABS(java.lang.Object a)
Obtain the absolute value of a word.

SIN(java.lang.Object n)
Returns the sine of a number.

COS(java.lang.Object n)
Returns the cosine of a number.

TAN(java.lang.Object n)
Returns the tangent of a number.

ARCSIN(java.lang.Object n)
Returns the arc sine of a number.

ARCCOS(java.lang.Object n)
Returns the arc cosine of a number.

ARCTAN(java.lang.Object n)
Returns the arc tangent of a number.

ARCTAN2(java.lang.Object y, java.lang.Object x)
Returns the inverse tangent y/x in degrees.

XOR(java.lang.Object a1, java.lang.Object a2)
Returns the exlusive or of two arguments.

AND(java.lang.Object a1, java.lang.Object a2)
Returns the logical and of two arguments.

OR(java.lang.Object a1, java.lang.Object a2)
Returns the logical or of two arguments.

SQRT(java.lang.Object n)
Returns the square root of a number.

ALLOF(java.lang.Object... x)
Returns the logical AND of its arguments.

ANYOF(java.lang.Object... x)
Returns the logical OR of its arguments.

BITNOT(java.lang.Object x)
Returns the bitwise NOT of the argument.

BITAND(java.lang.Object x1, java.lang.Object x2)
Returns the bitwise AND of its arguments.

BITOR(java.lang.Object x1, java.lang.Object x2)
Returns the bitwise OR of its arguments.

BITXOR(java.lang.Object x1, java.lang.Object x2)
Returns the bitwise exclusive OR of its arguments.

LSHIFT(java.lang.Object x1, java.lang.Object x2)
Shifts the first argument by the amount specified by the second argument.

ASHIFT(java.lang.Object x1, java.lang.Object x2)
Shifts the first argument by the amount specified by the second argument.

ROUND(java.lang.Object x)
Rounds its argument to the nearest integer.

MINUS(java.lang.Object x)
Returns the negation of its argument.

SUM(java.lang.Object x1, java.lang.Object x2)
Returns the sum of its arguments.

DIFFERENCE(java.lang.Object x1, java.lang.Object x2)
Returns the difference of its arguments.

PRODUCT(java.lang.Object x1, java.lang.Object x2)
Returns the product of its arguments.

QUOTIENT(java.lang.Object x1, java.lang.Object x2)
Returns the quotient of its arguments.

REMAINDER(java.lang.Object x1, java.lang.Object x2)
Returns the remaineder of the division of its arguments.

POWER(java.lang.Object x1, java.lang.Object x2)
Raises the first argument to the power of the second argument.

EXP(java.lang.Object x)
Returns *e* raised to the power of teh argument.

LOG(java.lang.Object x)
Returns the natural logarithm (log to the base *e* of the argument.

RANDOM(java.lang.Object x)
Returns a random integer between 0 (inclusive) and a specified upper limit (exclusive);

RANDOMIZE(java.lang.Object x)
Sets the seed of the random number generator to a specified value.

RADSIN(java.lang.Object n)
Returns the sine of a number.

RADCOS(java.lang.Object n)
Returns the cosine of a number.

RADTAN(java.lang.Object n)
Returns the tangent of a number.

RADARCSIN(java.lang.Object n)
Returns the arc sine of a number.

RADARCCOS(java.lang.Object n)
Returns the arc cosine of a number.

RADARCTAN(java.lang.Object n)
Returns the arc tangent of a number.

RADARCTAN2(java.lang.Object y, java.lang.Object x)
Returns the inverse tangent `y/x` in radians.

PI()
Returns an approximation of the value of pi.

- The developed Logo commands for the movement of the Turtle are the following:

LEFT`(java.lang.Object a)`
Turn the turtle left.

RIGHT`(java.lang.Object a)`
Turn the turtle right.

FORWARD`(java.lang.Object l)`
Move the turtle forward.

BACK`(java.lang.Object l)`
Move the turtle backward.

UPPITCH`(java.lang.Object a)`
Pitch the turtle upwards.

DOWNPITCH`(java.lang.Object a)`
Pitch the turtle downwards.

LEFTROLL`(java.lang.Object a)`
Roll the turtle to the left.

RIGHTROLL`(java.lang.Object a)`
Roll the turtle to the right.

PENUP`()`
Disable the turtle's pen.

PENDOWN`()`
Enable the turtle's pen.

POS`()`
Get the position of the turtle.

POSX`()`
Get the x location of the turtle.

POSY`()`
Get the y location of the turtle.

POSZ`()`
Get the z location of the turtle.

SETPOS`(java.lang.Object x, java.lang.Object y, java.lang.Object z)`
Set the position of the turtle.

SETPOSX(java.lang.Object x)
Set the x location of the turtle.

SETPOSY(java.lang.Object y)
Set the y location of the turtle.

SETPOSZ(java.lang.Object z)
Set the z location of the turtle.

- The acronyms of the main Logo commands developed for the navigation of the turtle are:

FD(java.lang.Object l)
Move the turtle forward.
BK(java.lang.Object l)
Move the turtle backward.

RT(java.lang.Object a)
Turn the turtle right.

LT(java.lang.Object a)
Turn the turtle left.

RR(java.lang.Object a)
Roll the turtle to the right.

LR(java.lang.Object a)
Roll the turtle to the left.

UP(java.lang.Object a)
Pitch the turtle upwards.

DP(java.lang.Object a)
Pitch the turtle downwards.

Geometrical constructions

A geometrical construction in MaLT appears as a result of a Logo procedure. The user will be able to dynamically manipulate the figure by using the 1dVT (for a detailed presentation of the functionalities of the 1dVT as well as the possible interactions with it see §3.2).

3.2 Interface

Turtle Scene

The graphical representation of the turtle's movements in MaLT is designed to take place in TS where a 3D turtle can be visualised in a grid-like environment (Figure 2). The background of the TS can be changed by using a Logo primitive command ("set Background") choosing a background from a library list with different options like colors, grid, or the limits of TS space. By using a Logo command it will also be possible to change the color as well as the thickness of turtle's trace which is a 3D object looking like a thin cylindrical line.

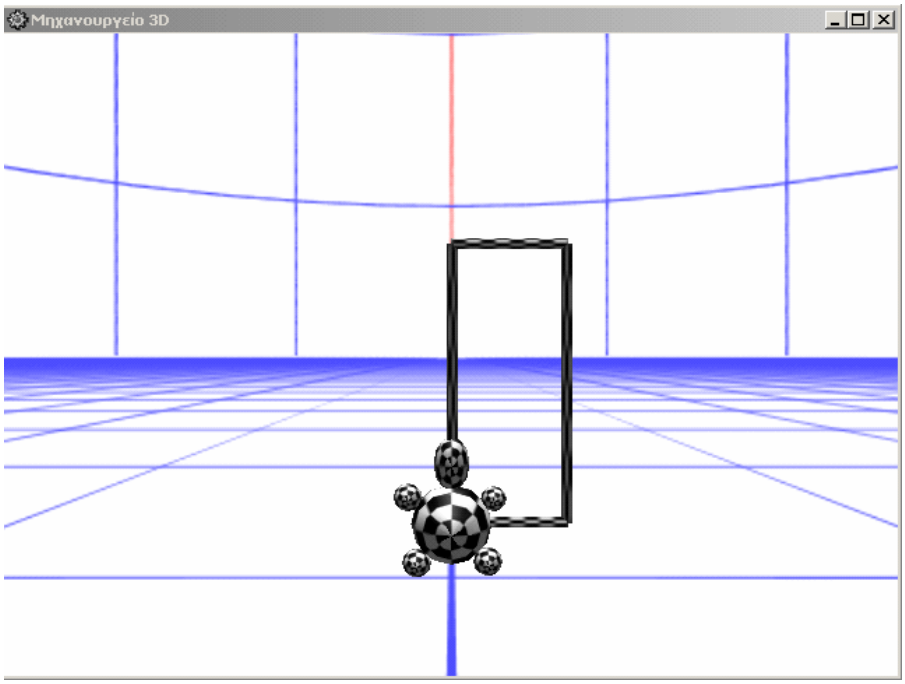


Figure 2: The turtle and its trace in the TS of MaLT.

Logo Editor

The Logo Editor is a rectangular window appearing on the right of the TS similar to the one shown at Figure 3 consisted of:

- The Editing component
- The Logo Response component
- The Menu Bar

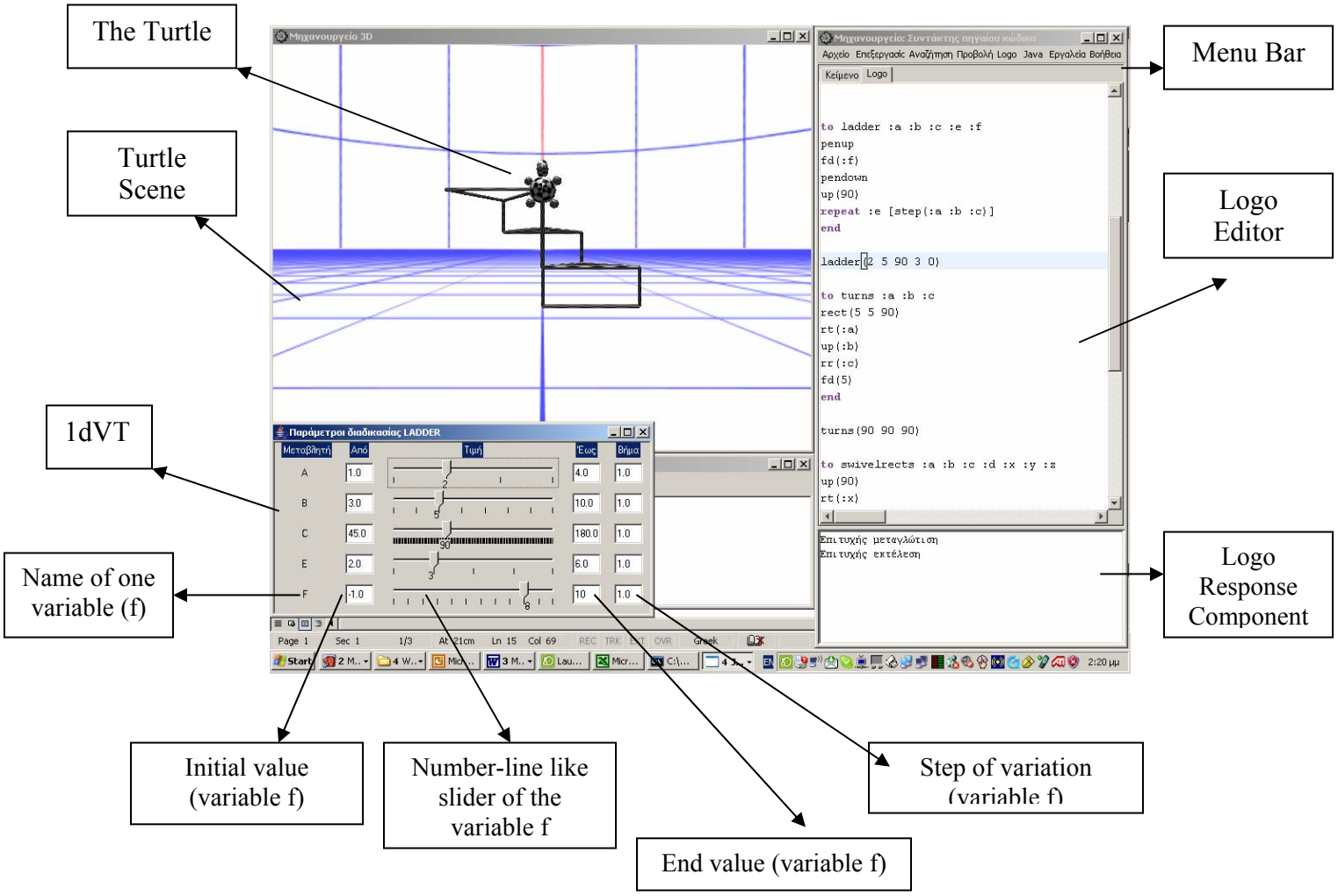


Figure 3: The current components of MaLT.

In the Logo Editor the user is able to write Logo commands (or procedures) and to define them by pressing the "Insert" button. The Logo Response component provides an "answer" to the user's scripting indicating whether a procedure is defined (or redefined) or not. In cases of mistaken procedures the Logo Response component automatically notifies the problem providing comments (e.g., "I don't know how to …" or "You don't say what to do with ..."). The Tool Bar menu offer options like Save, Cut, Paste, Italic, etc.

The Logo Editor thus serves as the linguistic formalisation of mathematical expressions and the symbolic representation of mathematical functions as well as the glue that binds all the representational modes together. There is an extension of Logo commands in 3D space including three kinds of turtle turns instead of one. These are:

LEFT(java.lang.Object a)
Turn the turtle left.

RIGHT(java.lang.Object a)
Turn the turtle right.

FORWARD(java.lang.Object l)
Move the turtle forward.

BACK(java.lang.Object l)
Move the turtle backward.

UPPITCH(java.lang.Object a)
Pitch the turtle upwards.

DOWNPITCH(java.lang.Object a)
Pitch the turtle downwards.

LEFTROLL(java.lang.Object a)
Roll the turtle to the left.

RIGHTROLL(java.lang.Object a)
Roll the turtle to the right.

1dVT

The main part of this component consists of 'number-line'-like sliders, each corresponding to one of the variables used in a Logo procedure (the 1dVT in Figure 3 consists of five sliders). The 1dVT consists of a rectangular window which is activated for a specific procedure including the use of one or more variables. The name of the procedure has to be inserted in the tool. Once activated the tool shows *n* sliders, one for each variable. The user needs to insert two values for each variable representing the range of that variable. The user can change in each slider the Initial Value, the End Value as well as the Step of the Variation (these numbers are shown in Figure 3 in the small boxes beside the sliders). Dynamic dragging of the slider with the mouse has the following effect: the figure constructed by the changes in turtle state, defined through the procedure, will change dynamically as the value of the variable changes.

3.3 Interaction

One of the most important and innovative features of MaLT is that it operates as a hybrid between Turtle Geometry and Dynamic Geometry Environments. As mathematical formalism and graphical representations of objects and relationships are dynamically joint, the user will be able to move to and from between the enactive, the visual and the

symbolic ways of expression. In MaLT the user will be able to interact with the computer environment in two levels:

- Logo programming language and
- Kinesthetic manipulation and visual control

Level 1: Logo programming language

The user will be able to define procedures in the Logo Editor through body-syntonic and at the same time mathematical formal way and to observe the graphical output that results from commands edited in the TS. The user can also use either the available Polar primitives (the absolute `setheading` command and the usual egocentric commands `forward, back, left, right`) or the coordinate Cartesian primitives related to `setpos` command. By programming he/she will also be able to define the turtle's step and thus better control the construction of 3D geometrical objects.

Level 2: Kinesthetic manipulation and visual control

In MaLT vectors are used as one of the means to handle and represent variation of variable Logo procedures. 3D geometrical objects can be designed as results of the execution of these procedures. The user will be able to dynamically manipulate the variable values –and thus the resulting geometrical objects- by using the 1dVT. Once activated the tool the user is able to drag dynamically the slider(s) of the variable(s) of a procedure and to observe the respective changes in the corresponding geometrical objects on the screen. It is noted that the user can manipulate the value(s) of the procedure variable(s) and not the figure itself.

4. The domain of activities in MaLT

4.1 Turtle's movement

Turtle Geometry is characterized as intrinsic because turtle's movement –and thus the construction of any geometrical figure- takes place without reference to any point outside the trace of the turtle. The fact that there is no reference to any distant part of space outside of turtle's path as well as the fact that the turtle's state is uniquely determined by its immediately previous state render turtle's geometry differential. In a sense any turtle heading and motion could be considered as representing a vector.

However, in MaLT non-intrinsic commands belonging to coordinate geometry are also be available. Thus, taking the turtle to specific locations in the coordinate plane is possible through the use of commands of the absolute coordinate and heading system. The

opportunity provided for referring directly to any point of the turtle's path rather than just the adjacent to the turtle's present state adds a non-intrinsic character to turtle's geometry. Consequently through turtle's movement and appropriate activities bridges can be formed between intrinsic, differential, co-ordinate and analytic geometry.

In particular, activities will be formed aiming both at an awareness of turtle's change of state and at an awareness of the 3D co-ordinate space through:

- movement of turtle in 3D space making use of commands related to intrinsic geometry (`FORWARD, BACK, LEFT, RIGHT, LEFTROLL, RIGHTROLL, UPPITCH, DOWNPITCH`);
- movement of turtle in 3D space making use of commands related to extrinsic geometry and of fixed co-ordinate systems, polar and cartesian `(SETHEADINGn, SETPOSITION(x,y,z) or SET(x), SET(y), SET(z))`;
- symbolic control of turtle's movement through variable procedures;
- kinesthetic manipulation and visual control of turtle's movement through the 1dVT.

For instance, students may be asked to drive the turtle to certain points on the screen or to move the turtle to certain points following a certain row, first point A then point B (see Figure 4). *Mystery procedures* may be developed by the researchers where turtle's movement will be defined either by Polar or Cartesian co-ordinates, as follows:

- *Polar*: `LEFT (:θ), UP (:φ) FD (:r)`
- *Cartesian*: `FD (:x), LT (90), FD (:y), UP (90), FD (:z).`

A *mystery procedure* may be a variable procedure where turtle's movement is defined in a certain way. Using the 1dVT and through kinesthetic manipulation and visual control students will be able to explore the way certain variables interrelate and affect turtle's movement in different planes.

4.2 Construction and transformation of geometrical figures in 3D space

In MaLT the user is able to construct geometrical figures in the programming interface of Logo and get immediate visualization of it in the 3D virtual space of TS. MaLT as a Logo environment will thus allow learners to use their body movements to kinesthetically pace out a geometrical construction by speaking a mathematical language that includes vectors embedded in turtle's moves.

The user will then be able to dynamically manipulate the figure by using the 1dVT as well as to use some navigation aids (e.g. cameras) to confirm and experience each figure from different viewpoints. Users will thus be able to experience the construction of a geometrical figure in an authentic 3d situation from:

- their body movements,
- the use of multiple frames of reference,
- the expression of them in a symbolic language,
- the abstraction of it to the geometrical language and
- the 3D visualization from many different viewpoints.

These characteristics:

(a) make MaLT quite different from other 3D environments that allow pupils to handle movement using black-box algorithms that in many cases are hard for students to understand;

(b) foster a more holistic understanding about 3D geometrical objects by improving the chance to develop sound spatial abilities and transfer the meanings of a geometric construction among a variety of different representations.

Example of a scenario

MaLT can be used by students in order to 'build' computer-simulated 3D geometrical objects identified either in their surroundings or from carefully selected 2D representations. At lower levels *Mystery procedures* may be developed by the researchers where turtle's movement will be defined either by Polar or Cartesian co-ordinates. At an advanced level, the user will be able to create complex figures by using the recursive power of Logo that allows the definition of variables and procedures based on previously defined ones. For example, the following two procedures may be developed by students or the researchers as a *Mystery* procedure using either an intrinsic-egocentric frame of reference (`trasformationA` procedure) or a coordinate frame of reference (`transformationB` procedure) to be used as a mechanism for the transformation of a previously defined parallelepiped within the 3D space. By moving the sliders of the 1dVT a vector constructed by the turtle in the TS, which will be the diagonal of a parallepiped, will be transforming in a dynamic way the respective parallelepiped.

```
to transformationA :r :φ :θ
lt (:φ)
uppitch (:θ)
fd (:r)
parallelepiped
end
```

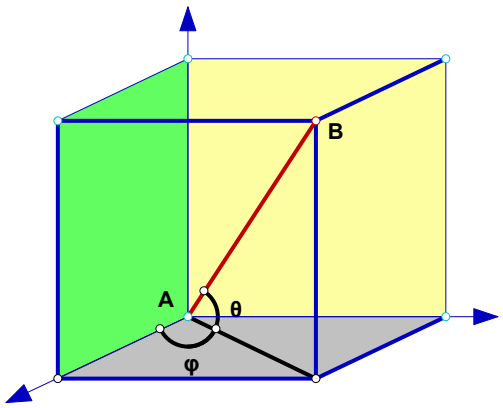


Figure 4: A schematic representation of the vector AB for controlling the construction of a dynamic parallelepiped.

```
to transformationB :x :y :z
setpos (:x)
setpos (:y)
setpos (:z)
parallelepiped
end
```

Using the aforementioned procedures students may be asked to:

- to change the variables so as to construct a cube;
- to explore how a parallelepiped changes when one of the procedures variables changes sequentially;
- to look for relations among the variables so as to construct geometric solids with specific properties;
- to explore the relationship between the facets of a parallepiped and its volume.

While working in MaLT with procedures similar to the above, students will be able to:

- test and develop intuitions and conjectures about the geometrical properties and relations of parallelepipeds;
- explore, describe and argue about the results of subdivision, multiplication and transformation of parallelepipeds;
- use vectorial properties connected with the geometrical elements that they use in school mathematics;
- move back and for between real 3D objects, 2D representations and computer simulated 3D representations.

5. Technical features

During the first year of ReMath, MachineLab has been developed as a stand-alone application developed in Borland Delphi, using the 3Impact game engine (http://www.3impact.com), and runs under Microsoft Windows. In the first version of MachineLab, parts of it have been ported to Java, for easier maintenance, using ActiveX and JNI to communicate with the native parts of the application. Because of the dependency on Windows native code, the application will remain Windows-specific.

MachineLab accepts, as input, scripts written in Logo or PascalScript. When executing a Logo script, it is translated to PascalScript, which is what is actually executed. In the first version, we have replaced PascalScript with Java, which is compiled using either the compiler provided by Sun in the Java Development Kit, or the free compiler Jikes (http://jikes.sourceforge.net/).

For the purposes of communicating with the MathDiLS platform, next versions of MachineLab are planned to import and export parts of their state in a yet to be agreed upon XML format.

DDA 5: Cruislet

Authors: Tryfona, Tsironis, Markopoulos

Talent SA, Educational Technology Lab

Work progress in 2006

Description of work

- Collaboration with ETL researchers to determine the software functional requirements posed by the theoretical framework of WP1.
- Cruislet specification of the domain and the main part of Cruislet. Cruislet is a microworld designed to provide learners with the ability to be involved in exploratory activities focusing on the use of the vectors navigating in 3d large scale spaces. In this framework, we defined
    - The mathematical concepts of references for Cruislet (i.e., Cruislet domain) and
    - The vector representation and functions for vectors for Cuislet (i.e., Cruislet main part).
- Technical specifications (in depth) of the Cruislet main part
- Implementation of the specifications of the Cruislet main part

Synopsis of the 1st version of the Cruislet DDA

1. Introduction

Cruislet environment is a microworld designed to provide learners with the ability to be involved in exploratory activities focusing on the use of vectors navigating in 3d large scale spaces.

Vectors are an essential scientific concept, indispensable for both the physicist and the mathematicians. Despite the obvious importance, there is no clear, universally accepted definition of this term.

In mathematics curricula not much notice is given to the mathematical nature of vector as object, as a set of properties and as a signifier of the measure of entities such as velocity, force, etc. Usually the introduction of the concept involves the visualization of the vectors as arrows. Teachers focusing on the procedure of the drawing of the vector mention that "*…before we can draw a vector, we have to decide where to place the tail of the vector. If we are drawing forces, we usually put the tail of the vector at the place where the force is applied"*. As a result a number of pupils confuse the vector with this arrow, its representation, like a static image regardless its mathematical properties. (Kynigos & Latsi).

In Cruislet DDA vectors represent the dynamic variation of object attributes and students/learners could study the two basic operations defined for vectors, the multiplication of a vector by a number (also called scalar multiplication) and the addition of two vectors.

For example, once they have chosen a starting point for a moving agent (i.e., an origin for space), every point in space corresponds to exactly one vector, namely the vector whose tail is at the origin and whose head is at the given point. Thus, when the agent is defined to start from Athens and fly over Europe learner could follow this movement through the dynamic transformation of the vector representing agent's displacement.

2. Domain

The mathematical concepts of references for Cruislet are:

- Vectors in Cartesian coordinate system and Polar coordinate system
- Vector operations in these coordinate systems
- Geometric transformations (displacement, rotation, symmetry) and their representations in these coordinate system.

The mathematical content of reference for the movement / navigation is the Geometry of Turtle[1]. Within this mathematical framework the movement of the agent represents the displacement of the turtle. A turtle is a mathematical object which is defined by its previous condition (position and direction). The Geometry of the turtle representational system is a compound system involving polar and Cartesian coordinates. In particular the displacement of the turtle is defined in polar coordinate system, while the position setting of the turtle in the Cartesian one. The vector in this system is considered not merely as a mathematical object but more as a representation of an action. This action is the displacement of the turtle. In polar coordinate system, spherical coordinates define a vector in terms of its length (R), an angle between the vector and the z-axis (φ) and the angle between the vectors projection onto the xy plane and the x-axis (θ).

Correspondingly, the displacement of the turtle in a 3d environment is defined by the definition of 3 variables, the measure and the two angles which determine the length of the displacement and its direction.

In this sense, the navigation of the agent within the Cruislet environment corresponds to the displacement of the turtle in Geometry of Turtle framework. The user could actually navigate the agent within Cruislet microworld through the use of Logo editor programmability. The Logo programmability is considered necessary as it provides the users with the option to actually anticipate the result of their action. Nevertheless, according to the theoretical aspects, a learning environment should involve learners in interaction with the learning materials and provide them with the ability to reflect on the actions carried out and consequently make abstractions. (von Glasersfeld, 1996. The main mathematical concepts involved within the Cruislet microworld are the mathematical concept of vectors and the two basic operations (functions) defined for vectors, the multiplication of a vector by a number (also called scalar multiplication) and the addition

[1] "Geometry of Turtle" is a widely-used term, adopted from the Logo community, used to refer to the way to build geometries.

of two vectors could be considered as the repetition of a displacement of the turtle (multiplication) and its total displacement from a given point (addition).

3. Objects – entities

Crusilet is a microworld where the user could actually create new moving agents, define their relative position on the space, adjust the relative position of the camera and then put them in motion. The navigation/displacement of the agent is possible either by the use of the UI of the software or by the use of the Logo editor.

Cruislet microworld is consisted of the above objects:

The terrain scene

At this point, the terrain scene is consisted of a geographic map of Greece which could be seen to an analysis ranged from a satellite view up to 1:50.000. This means that the altitude of the moving agent could vary from 200 m up to 723 km. The terrain scene can be viewed from two map viewers. The first map viewer is in 3d mode and the second is in 2d mode. The terrain scene could be viewed from the camera adjusted to an agent.

Geo-coded information

These 3d terrains, maps and landscapes contain *real geo-coded information*. By navigating through landscapes, users can choose to see descriptive information of the various land features and geographic entities (cities, point of interest, road network, etc). Such information is the geographic coordinates of the center of a city, the population of this city or the relative position in relation to other cities.

The moving agent

The moving agent could be the simulation of any kind of flying machines (aircrafts, helicopters), vehicles moving on the surface (cars, trains, boats) or man kind robots. There is the option to create more than one agent. These agents could be stand together on the terrain on but one of them could be active each time. The creation of an agent presupposes

the adjustment of a camera to a relative position to the agent. The active agent is the one whose camera view is shown on the two map viewer windows. Consequently, both map viewers are centered on the active agent.

The altitude variation tool

The modification of the altitude of the displacement of the agent is possible through the use of the altitude variation tool. By choosing the altitude variation tool he/she can actually adjust the distance between the moving agent and the surface level.

The Logo Editor

This component will be the Logo-like programming interface that engages students in logical procedural thinking and it will be linked to the displacement of the moving agent. The user will be able to write, run and edit Logo programs. In particular, he/she could, as we mentioned above, pre define a particular trip, for example, a fly over from Athens to Alexandroupoli by taking into account the geo-coded information.

A number of Logo commands will be developed which will be provide the ability:

- Create new agents or delete existed ones
- Define an agent as the active one
- Set an agent on a new position on the terrain. The new position could be defined either by its absolute coordinates or by its relative position to a geo coded information, an object of some of the layer of the map. The general form of the setting agent's position Logo commands is:
    - setPosition <lat><log>
    - setPosition<layer><object_id>
- Vary the position of the agent's camera
- Move the agent to a new position. The displacement could be defined with a particular step and number of repetition. The geo coded information that provided in the layer of the map should also be considered. The general form of the displacement Logo commands is:

- setPosition <lat><log><step size><step speed>
- setPosition<layer><object_id><step size><step speed>

The movement of the agent it takes place in a particular altitude (3,5km). If the agent is in lower altitude, then the agent moves up vertically to 3,5 k, it executes the particular movement to the final destination and finally it moves down to the initial altitude. If the initial altitude is above the 3,5km the altitude remains the same.

- Move the agent towards a particular direction and with a particular step. The general form of the movement Logo commands is:
  - move<angle θ><angle φ><step size><step speed>

The trace of the agent

Each time the agent moves there will be an option to leave a trace behind it. The trace has the form of a vector (ending with an arrow). Every step of the agent produces a different vector. The whole displacement of the agent is actually a number of vectors. These vectors are selectable.

4. Interface and Interaction

4.1 Interface

The terrain scene

The scene can be seen from two different windows. The first window contain is the 3d representation of the map whilst the second contains its 2d representation. These windows could be simultaneously displayed by splitting the screen (the first above the second). Figure 1 and 2 present a snapshot from the 2d representation mode of the geographic map of Greece and the 3d representation mode respectively. There is also the option to have only one of the two windows displayed. These windows are the two map viewers. They take up almost the 2/3 of the screen. The remain area is called the Cruislet area and it is divided into two sub areas. The above sub area contains the tabs selection and the below the vector component. There are three tabs:

- The thematic browser containing the configure tree of the layers' visibility
- The Logo editor, where the user edit the programs
- The administrator tab of the attributes of the active agent and its camera. The user sets the camera view in relation with the particular agent and the camera view remain its relative position as the agent moves.

Figure 1: A snapshot of the 2d representation mode

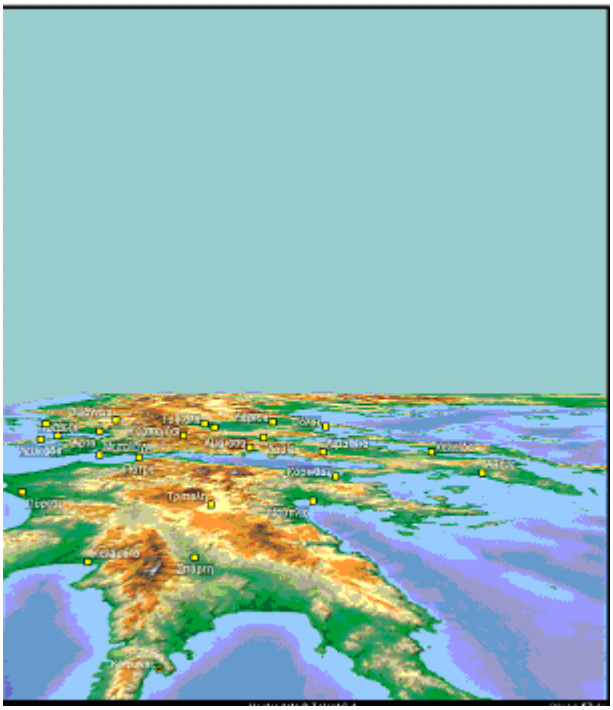


Figure 2: A snapshot of the 3d representation mode

Geo-coded information

The real geo-coded information contained in the terrain scene is descriptive information of the various land features and geographic entities. There is the option to display:

- the boundaries of the nation, the boundaries of prefectures, counties, municipalities, etc.,
- the name of cities, capitals, prefectures, counties, municipalities
- their population
- their geographic coordinates
- the main rivers, lakes, seas, mountains, peaks, etc.
- their altitude above sea level

The user could actually choose which of these attributes will be displayed. By default the names of the major cities are displayed when the altitude of the agent is below 400 km. The display of these geo coded information is possible through the selection of the particular layers in the tabs selection area.

The moving agent

In the administrator tab there is a built-in library of 3d models. The model of the active agent could be changed. There is also a combo box in the administrator tab. This combo box contains all the created agents.

The altitude variation tool

The reduction or the increasing of the altitude affects the actual scale of representation of the area of the terrain and consequently the representation of the displacement. There will be a “+” and a “-” button at the bottom of the active window. By pressing each of the buttons the user could actually increase or decrease the altitude of the agent. This function is also possible by scrolling wheel of the mouse. Scrolling is doing the modification of the

altitude progressively, while the pressing of the buttons alters the altitude instantly. The altitude of the agent is displayed on the bottom right corner of the active window

The Logo Editor

The Logo Editor is a Logo-like programming interface. It is actually one of the tabs on the tabs area. It is not always visible unless it is selected.

The trace of the agent

This trace will be a single, selectable, three- dimensional object (a thin cylinder ending on an arrow). It is actually a 3d vector representation.

4.2 Interaction

The terrain scene

The user could activate each of the two windows, the 3d and the 2d representation mode. He can select each of the geo coded information by clicing on the partiar layer temap on the tabs area

The moving agent

The user could actually administrate the agents. The administration of the agents consisted of :

- The creation and setting of the new agent on the terrain
- The deletion of the agent / agents
- The deletion of the trace of the agent
- The editing of the agent's attributes

The administration tab contains a combo box where all the created agents are displayed, tools for the creation and the deletion of the agents and a form where the attributes of the active agents are included.

These attributes of the active agent are:

- The name of the agent
- The 3d model of the agent
- Its relative position, (altitude, geographical coordinates: lat, long)
- If the agents will leave a trace as it moves or not
- If the agent will be visible or not

The user, through the use of the administrator tab could vary the attributes of the active agent. Each of the agents has a unique name. The 3d model of the agent actually determines its appearance. There is a built-in library of 3d models selectable by the user. The model of the active agent could be changed. Finally there is the option to alter the position of the agent on the space.

Each agent has a camera adjusted. The administrator tab also includes the attributes of the camera. These are:

- The distance from the agent
- The height on or under the agent
- The number of the degrees round the agent
- The vertical alteration of the camera view
- The horizontal alteration of the camera view

The above attributes are active only when the 3d map viewer window is active. In the 2d map viewer window the camera is above the active agent. In thi case only the distance between the came and the agent is defined.

When an agent is active the position of the camera is relative to the position of the agent and the user could not change it. The free navigation of the camera is possible whenever none of the agents is active. The inactivation of an agent is possible through the comboi box. It contains the names of the agents and an option called “none”.

Geo-coded information

The movement/displacement of the agent is related with the including geo-coded information all the time. The navigation of the agent is performed by the user and it could not be possible without the reference in the geo-coded information. For example, the planning of the flying over Athens to Alexandroupoli should take into account the appropriate alteration of the altitude in order to take off from Athens, fly above the major mountains and land in Alexandroupoli. Supposing that the landing and taking off procedures are vertical, the user has to define a safe altitude in order to complete the journey. Thus, when the user is trying to fly the agent through a mountain a warning message appears stating that the displacement is impossible.

The Logo Editor

By the use of the Logo Editor the navigation of the moving agent will be programmable. The user editing a trip will watch the movement of the agent on the two representation windows.

The trace of the agent

The selection of a vector is possible by clicking onto the vector. A second click onto a vector causes the unselection of it. Also the selection of a number of vectors at the same time is provided by pressing the Ctrl key.

By pressing right-click on the terrain a pop-up menu is displayed containing two available functions:

- Multiplication of a vector. This function is available only when one vector is selected on the map viewer. The user can define a number with which the selected vector will be multiplied. The result is a new vector which is displayed on the map. The new vector begins from the point of the map where the right-click was occurred, it has the same direction with the selected vector but its multiplied measure. This option becomes functional when the geo-coded information is considered. As we mentioned above there is a restriction concerning the value of the length of the displacement. For example, the user having the city of Kalamata as starting point plans to move the agent to the city of Thessaloniki. Considering

the geo-coded information referred to these towns he/she defined the initial vector and calculates the times of its repetition.

- Addition of two vectors. This function is applied upon all the selected vectors. The addition of the selected vectors produces a new vector which begins from the point where the user made the right-click.

Finally, through the use of the pop-up menu the user could unselect the selected vectors or delete these vectors.

Additional vector-based operations

The more common operations implemented in the vector-based Cruislet are expressed via:

(i) *range queries*, e.g., which object is in the circle with radius of 10m around point A,

(ii) *window queries*, e.g., which other geometric figures exist in "this" window,

(iii) *nearest neighbor queries*, e.g., which point is the closest to that polygon.

5. Example of Scenarios

A number of scenarios involving the study of the concept of vectors and the two basic operations, multiplication and addition, can be explored. Here, we present three scenarios focusing on the 2d and on the 3d representation of the terrain respectively.

- The user is called to activate the 2d representation mode, create an agent and set its position upon the city of Athens. The user should activate the trace of the agent and navigate it aiming to form, using the trace vectors, an isosceles triangle. The vertices of the triangle shoud be three major city of Greece and inside the area of this triangle a river, lake and a mountain should be included. The user has to calculate the total displacement of the agent and the area of the mental triangle that this displacement formed. One of the restrictions that should be putted is to form the biggest and the smallest triangles that could be possibly formed.

- The user is called to activate the 3d representation mode of the terrain and create an agent in order to organize a trip setting as starting point the city of Kalamata. The initial height of the agent has to be at 200m. The destination of the trip is the city of Ioannina. He/she

has to plan the trip in order to visit all the cities of the mainland with population above 150.000. He/she has to organize the flight and be aware of the mountains that could possible interrupt the flight. Also he/she has to calculate the total displacement of the agent. Additionally, he/she has to reorganise the trip in order to reduce the number of the total steps needed to complete the trip. The user should experiment by creating and editing a Logo program for the planned trip.

- Students/users use Logo Editor to construct a parametric unit vector of their own. The term parametric vector means that we define a procedure that describes the movement of the turtle using variables. For example $\vec{v} = (r, \phi, \theta)$ can be a parametric vector that is produced by specific Logo commands. It might be useful to substitute some of the parameters with constants and thus to concentrate on how the changes of one parameter act on agent's behavior.

  We set a problem. Students must use the parametric unit vector in order to get from a given point or city on the map (we can call it starting point S) to a certain destination D. Assuming that r is constant then φ and θ are the two variables they can be changed. Students have to decide on how to manipulate the two remaining variables and move the agent towards the destination.

  The teacher might ask students to get straight to the destination. In that case students have to set specific φ and θ and repeat $\vec{v} = (r, \phi, \theta)$ (which is not parametric any more) as many times as it's necessary to reach the destination. The activity involves scalar multiplication of a vector that is targeted to the destination.

An example of a scenario of the above is the following: The student is called to activate the 3d representation, create an agent and set its position upon the city of Patras. The objective is to direct the agent upon the city of Thessaloniki by using only one vector and the function of multiplication of this vector. Initially, the student could move the agent up to 50 km from the starting point (city of Patras). Then by choosing this initial vector of displacement he has to calculate / estimate the appropriate number of repetition of this vector of displacement in order to direct the agent to the city of Thessaloniki. The student should take into account the geo-coded information of the two towns and calculate the appropriate a number of repetition of this vector as well as its direction.

DDA 6: Mopix

Authors: N.Winters, K.Kahn, D.Nikolic and C.Morgan

London Knowledge Lab and Institute of Education

Work Progress in 2006

During this first year, we have developed Mopix, a tool for programming games and animations with equations. MoPiX is an early alpha prototype of a system in which students construct models by assembling virtual components (e.g. balls, arms, hands, walls) and giving the components behaviours by dropping equations on them. The equations define functions of time so that the model can be animated by computing attributes at successive time points. Internally, the equations are represented as content MathML. An equation editor will be added to the software soon. Current functionality includes, for example, the ability to create a ball bouncing subject to gravity.

The overall goal of our learning environment is for learners to iteratively construct their own game environment in which characters and events are determined by the use of applied mathematics. To do this they will progress through a number of steps and share the constructions they create. The games are built by following a construct-animate-investigate-collaborate cycle.

In the current version, students can undertake the construct-animate-investigate part of the cycle but cannot as yet save or share their constructions. Animation is achieved by dragging a component onto the stage, dropping equations on it, thus specifying behaviours and then animating these by pressing play. The current version has been tested by students in two 2.5-hour sessions and is available here: http://www.lkl.ac.uk/mopix

We have met all the goals we set ourselves. However, we would like to add an equation editor (currently under development) and fix a few more bugs. We expect to have the editor integrated into the next version by November 20th.

We will have a prototype ready for experimentation at then end of month 12, but only in the lab. As we are developing ‘from zero’, parts of the tool are buggy and as such MoPiX not suitable for use in schools, unless part of the iterative development cycle. However, development is continually on-going.

MoPiX: Programming Games and Animations with Equations (DDA from the LKL)

Niall Winters, Ken Kahn, Dusanka Nikolic and Candia Morgan
London Knowledge Lab and Institute of Education

Introduction

Console games and animated movies are an everyday part of many students' lives. However, while they enjoy playing (or watching) they do not often understand the mathematics that underpins the development of these media.

We have developed MoPiX, a tool for programming games and animations with equations. MoPiX is an early alpha prototype of a system in which students construct models by assembling virtual components (e.g. balls, arms, hands, walls) and giving the components behaviours by dropping equations on them. It allows learners to build, inspect, edit, execute, monitor, and share dynamic visual models. These models can be executed to produce animations, interactive simulations, or small physics-based games.

In MoPiX, equations define functions of time so that the model can be animated by computing attributes at successive time points. Internally, the equations are represented as content MathML. They are stored in an equation library and can be created by the learner using the in-built equation editor. Current functionality includes, for example, the ability to create a ball bouncing subject to gravity, the ability to combine different objects and manipulate their properties.

The overall goal of our learning environment is for learners to iteratively construct their own game environment in which characters and events are determined by the use of applied mathematics. To do this they will progress through a number of steps and share the constructions they create. The games are built by following a *construct-animate-investigate-collaborate* cycle.

While it is possible for students to use MoPiX individually, a particular feature of the DDA will be its capacity to facilitate student-student interaction and collaborative problem solving among small groups of students. *Note: this will be implemented at the next stage of development.* Student constructions will be sharable between mobile devices (such as ultra-mobile tablet PCs) and inter-group communication will be facilitated by the open nature of knowledge sharing. In addition, different students within a group can work on different aspects of a scene, which can then be combined.

Domain of the DDA

The mathematical domain may be described as applied mathematics in that mathematical equations are used to define and control the properties and behaviours of visual components. The initial set of equations for horizontal and vertical motion provided in the equation library for users to choose from allows behaviours consistent with Newtonian laws of motion, though it may be possible for users to construct equations that define other types of behaviours.

Equations in MoPiX are playing the same role as program code in other modeling systems. While some behaviours are not readily expressible as functions of time, initial testing has revealed that a surprisingly rich and varied set of behaviours can be defined and combined to make a wide range of simulations and animations (and soon also games). We expect that students will see algebraic equations in MoPiX as a source of empowerment and creativity. In addition, we wanted to provide students with a way of modelling in which equations are ‘tools to think with’. Their association with behaviours allows the learner to have a direct and immediate sense of how their application changes the state of objects in a dynamical way.

The current library of equations in MoPiX covers these areas:

- Horizontal position, velocity, and acceleration
- Vertical position, velocity, and acceleration
- Horizontal bouncing
- Vertical bouncing
- Constant and dynamic transformations of width and height
- Constant and dynamic transformations of colour
- Constant and dynamic transformations of transparency
- Constant and dynamic transformations of rotation
- Change of shape
- Constraining a point on the perimeter (or centre) of one object to be attached to a point on the perimeter (or centre) of another
- Constraining one object to be at the horizontal or vertical average of the position of two other objects
- Controlling whether an object leaves a trail as it moves and if so the width, colour, and transparency of the trail.

Design

Theoretical underpinnings of design

The design of MoPiX is informed by constructionist and socio-cultural theories of learning. Within these general frameworks, we make use of the following ideas: construction toolkits, the role of game making, semiotic mediation, collaboration through communication in context and Learner-Centred Design.

Construction toolkits The constructionist approach to learning (Papert, 1980; Harel & Papert, 1991; Kafai & Resnick, 1996) 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 very much based on these principles. They are dynamic visual environments that support building activities in social contexts. Learners build constructs with fundamental elements and then activate these constructions as a means of investigating their hypotheses. Because the kits are so strongly visual, they support the creation of structure and function (Strohecker and Slaughter, 2000).

MoPiX is conceived as a constructionist toolkit in this sense. The fundamental elements are mathematical equations that, when combined and assigned to objects, create animations and graphs. Learners build components and may immediately observe the behaviours of their constructions. The visual feedback allows them to form and test hypotheses about: the meanings of the equations; the behaviours of moving objects in interaction; the relationships between variables such as velocity and acceleration of objects within systems in motion; the effects of changing parameters (such as the value of g).

The role of game making A growing body of research indicates that playing computer games can lead to serious learning (e.g., Shaffer et al 2005) and that learning can be deeper and more richly interconnected if game playing is combined with game making (Kafai 1995). MoPiX supports the creation of animations and interactive visual simulations. A game component can simply be an agreed upon goal or challenge or, more ambitiously, can be embedded in the software.
Semiotic mediation According to a Vygotskian framework, meanings are appropriated from others in a dialectic of prior experiences and identities and new experiences and identities, within social practices (Vygotsky, 1978). In this process, language and other cultural tools, including other forms of semiotic representation, are critical forms of mediation, The interaction between the individual and the real world is regulated and transformed by the use of symbolic material, which offers culturally specific systems for representing reality and hence a universe of meanings for interpreting experience. The design of MoPiX aims to provide a multi-semiotic set of representations that will offer a rich system of potential meanings. These range from visual simulations of motion that should connect to everyday physical experience to more-or-less conventional mathematical representations using algebraic and graphical forms. The linking of mathematical to visual representations is intended to support learners in using mathematics and the principles of laws of motion as means of making sense of the world.

Collaboration through communication in context Socio-cultural approaches emphasise the interpersonal nature of learning as well as the importance of the tools or learning materials. Consideration of the proposed context of use plays an important part in the design process (Winters and Price, 2005). MoPiX is intended to be used in a collaborative context in which learners are able to share their constructions and investigate, modify and play each other's constructions in a game-like manner. Learners may work with the

microworld individually to construct and investigate. However, the use of tablet PCs will allow both face-to-face communication among a group of learners and sharing of constructions by direct communication between the devices, while the development of a web-based library will enable sharing of construction components among a wider community of users. This environment aims to encourage collaboration and 'exploratory talk', enabling students to exchange ideas and to engage critically with each other's contributions - a form of talk that, it is argued, supports learning through group interaction (Barnes, 1976; Barnes & Todd, 1995; Edwards, 2005; Mercer, 1995).
Learner-Centred Design (LCD) Learner-centred design (Soloway, Guzdial and Hay, 1994) is built on socio-cultural and constructivist theories of learning and the user-centred approach to interaction design. Learner-centred design is based on the premises that a user of technology constantly changes, through learning, and that their needs from the technology change in the process. In particular, *the user learns through using the technology, and the design of the technology needs to account for that learning*. This leads to the question of how can environments (including the interface) support learners and learning? LCD suggests that students learn through an active, social process of meaning construction (Vygostky, 1962). Critically, understanding is built up through the acts of conversing with others, constructing artefacts, and reflecting on those conversations and artefacts. Soloway, Guzdial and Hay (1994) see scaffolding as the main role of teachers in constructivist learning, and propose that this should be the role of the interface in technology-rich environments. While we keep this as an option, we see a more critical role of the teacher in mediating the communication between the group of learners.

Quintana et al (2005) propose a framework for designing scaffolding structures. Position this framework in the context of inquiry-based learning. Consequently, organize the framework around three processes: 'sense making', which involves the basic operations of testing hypotheses and interpreting data; 'process management', which involves the strategic decisions involved in controlling the inquiry process; and articulation and 'reflection', which is the process of constructing, evaluating, and articulating what has been learned. From these principles, they derive a framework that includes several elements:

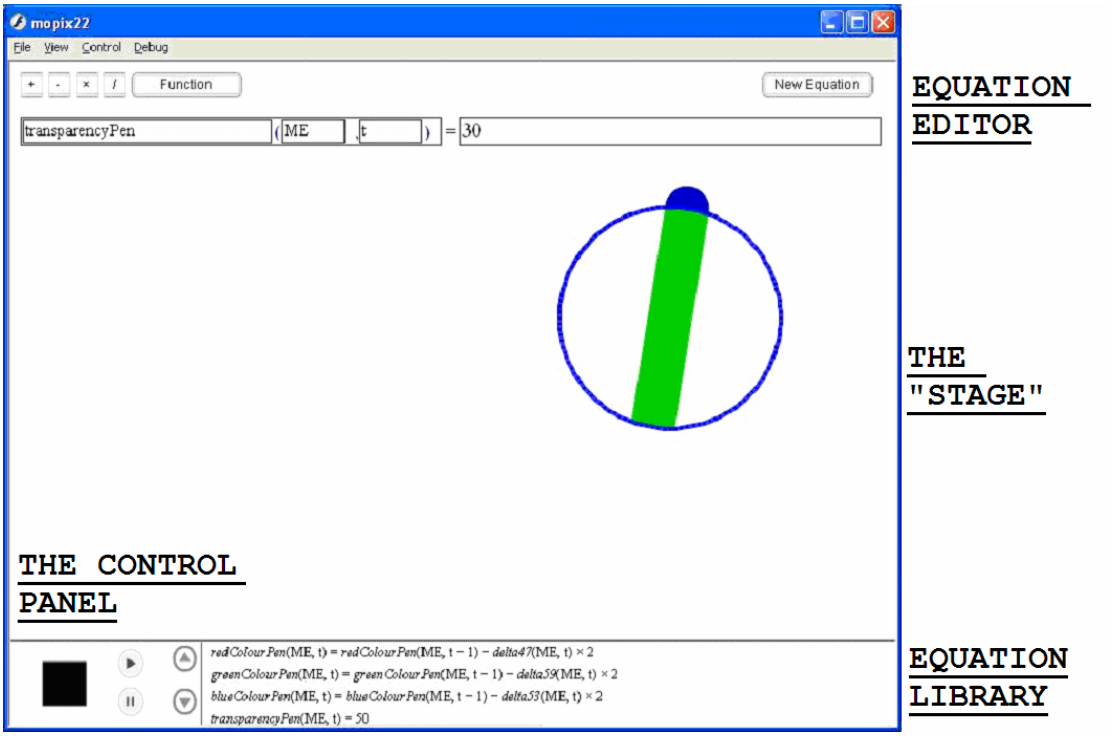
- The task model, the constituents of activity derived from the inquiry based learning literature.
- Obstacles encountered by learners.
- Scaffolding guidelines provide principles for designing scaffolds to help learners overcome the obstacles.
- Scaffolding strategies, more specific implementation approaches
- Examples

This requires further investigation to investigate the benefits for learning design by integrating LCD with the constructionist toolkits approach.

Description of the interface and the interaction

The current version of the MoPiX interface consists of:

1. The Stage where simulations and games are executed
2. The Equation Editor where new equations are created
3. The Equation Library where pre-defined equations can be found
4. The Control Panel with play/pause buttons and a source of new components



To create a model, drag a black square from the Control Panel to The Stage. Drag and drop any of the equations from the Equation Library or Equation Editor on top of the black square. At any time you may press the Run Button to run the model. While it is running you can add more equations or click the Pause Button. There are equations for change the size, colour, transparency, orientation, appearance, and position of components.

Equations always define functions of two arguments: the object and the time. The object is always the template variable *ME* on the left-hand side of the equation. When dropped on an object *ME* is replaced by the object's name. Some equations refer to other objects with template variables *OTHER1*, *OTHER2*, …, *OTHER9*. These are also replaced by object names when dropped on components. To use an equation with an OTHER template first drop it on the ME object and then the required OTHER objects and finally drop it again on the ME object to add it to that object.

Equations associated with objects can define the temporal values of attributes such as position, appearance, width, height, rotation, transparency, and colour.

Evolution of MoPiX

Programming on MoPiX began in May 2006, based on initial design specifications and theoretical underpinnings developed over the previous six months. The evolution of the current prototype can be views through two distinct phases:

Phase 1: May –September 2006

- initial design considerations: multiple screens (separate ones for construction and play mode) *versus* one screen for everything (couple of versions of these)
- arrangement of the objects on the screen (construction bits, graphing, equations, buttons) (another couple of versions)

- on the internal side, MathML adopted as a way of communicating math's equations; equations get their notation
- several versions of the moving single object on the screen (bouncing)

- equations added on the screen: scrolling, choosing and dropping them on an object becomes possible
- graphing (choosing a function from a menu) introduced
- more equations (colours, oscillating equations, shape equations)

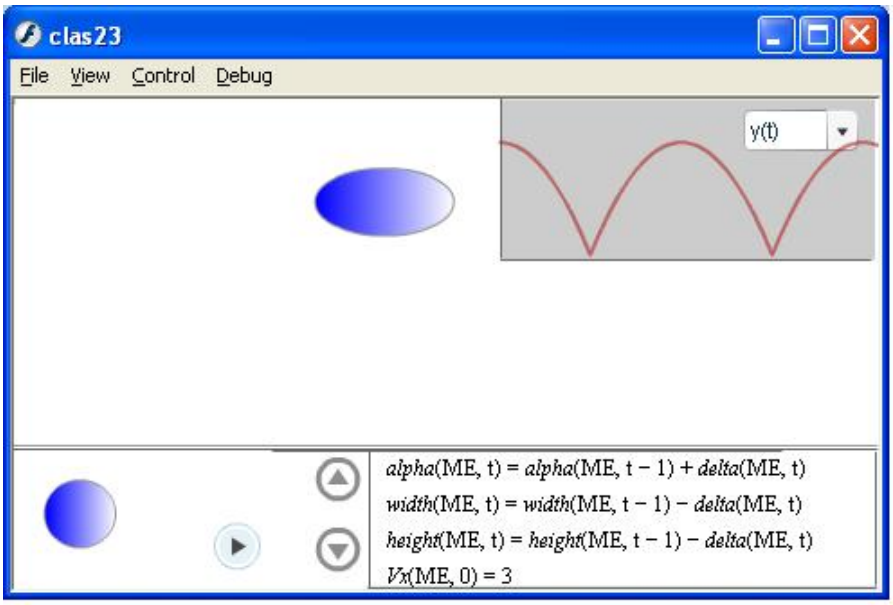


Figure 1: MoPiX (Phase 1 – v1)

Next, as part of this phase we added:

- pause and rewind buttons added

- improved graphical interface
- more equations added
- testing MoPiX on different platforms: PocketPC, Sony PSP and PC

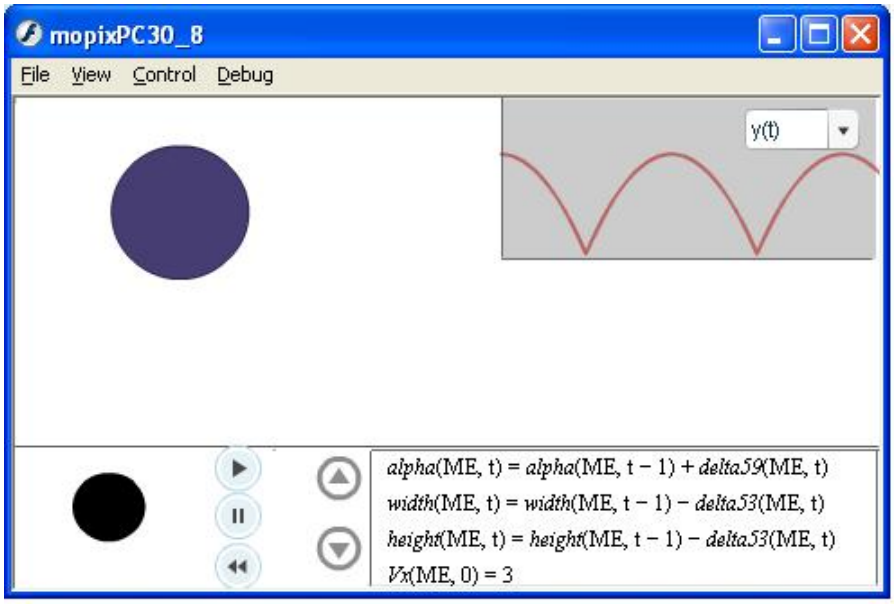


Figure 2: MoPiX (Phase 1 - v2)

Phase 2: September - November 2006

- design of Equation Editor starts (version with open source Java Script editor considered, couple of versions of partially integrated JSEditor and MoPiX made)
- two different approaches considered: parser versus getting the user enter an equation in a tree form, second adopted
- Editor and MoPiX being developed in parallel: Editor being written in Action Script with XML /MathML as main characteristics; MoPiX being improved in terms of number of equations, graphing by using equations, equations that reference multiple objects
- testing with the students and fixing a number of bugs

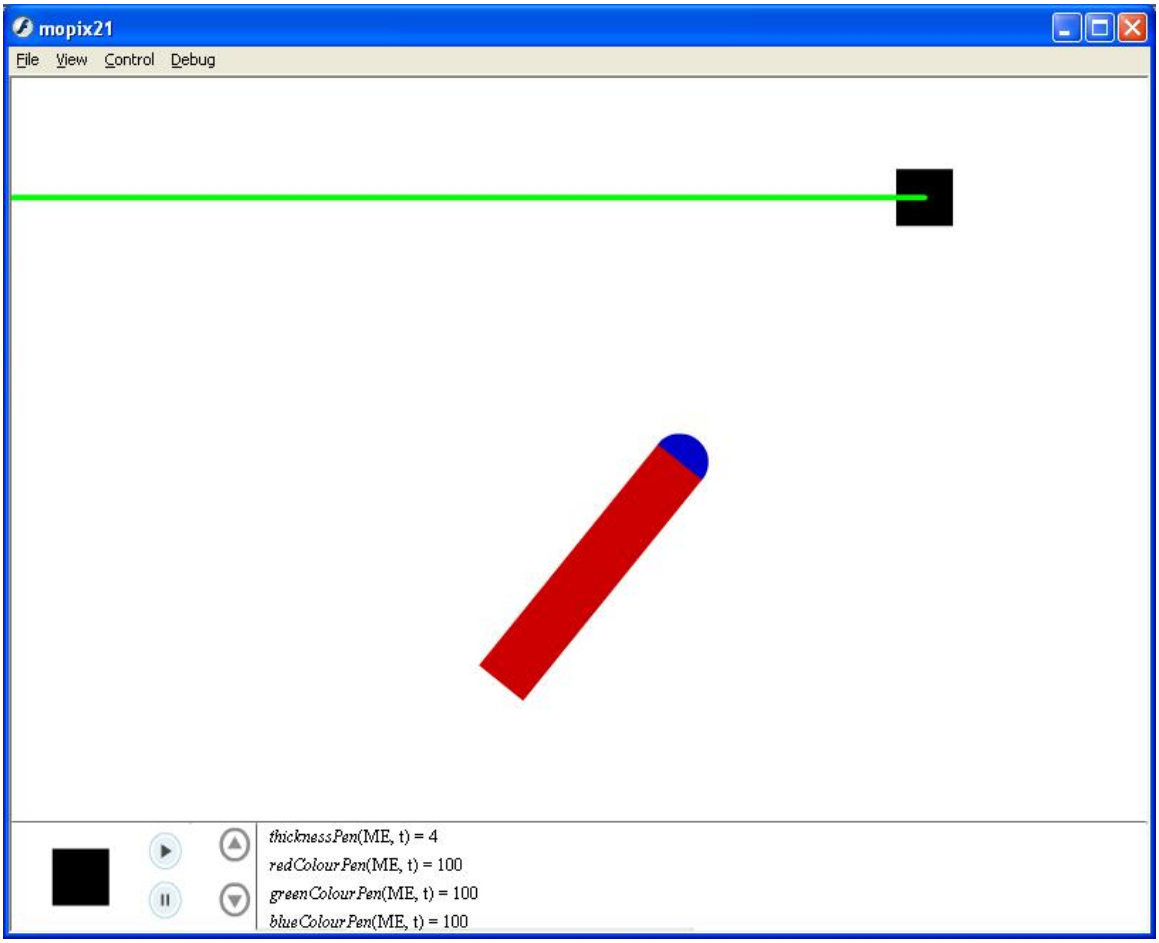


Figure 3: MoPiX (Phase 2 - v1)

In the next stage of this phase we added:

- first version of *Equation Editor* finished (accepting user's tree-structured input and generating MathML, displaying equation on the screen)
- improved Equation Editor integrated with MoPiX with the possibility to create, drag and drop the user's equation on an object

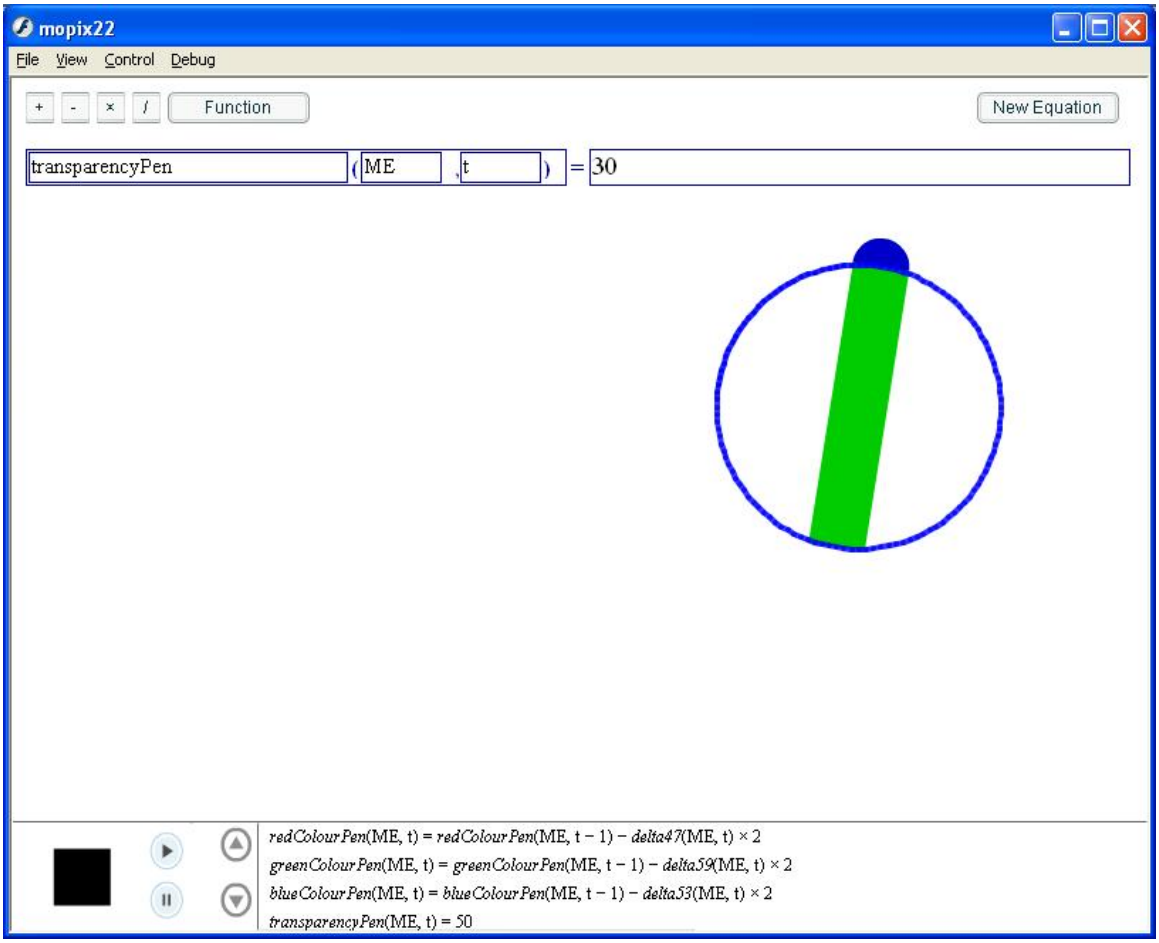


Figure 4: MoPiX (Phase 2 v2), the current stable version, as of November 2006

The current version has been tested by students in two 2.5-hour sessions and is available here: http://www.lkl.ac.uk/mopix. We made several changes to MoPiX in response to feedback from our testers. Five bugs were uncovered by the testers and were fixed before the next session. Testers found the names of equations elements confusing. In response we changed OBJECTn to be OTHERn, hitWall and hitGround were renamed amIHittingAWall and amIHittingGround and alpha(ME,t) became transparency(ME,t). The testers were confused by an artefact of the graphing equations that caused a horizontal line to appear when the graph reached the right hand side. This was fixed soon after the session. We also quickly implemented the testers suggestion that we change the sample equations so the ball bounced off the line separating the stage and equation library.

Planned enhancements to MoPiX

In the coming months we plan to enhance MoPiX with the following functionality:

- to save models and equations using MathDiLS

- to browse and load previously saved models and equations using MathDiLS
- to support collisions between model entities
- to support user input for creating interactive simulations and games
- to support the grouping of sets of equations that accomplish a higher level behaviour (e.g. a group defining position, velocity, and acceleration).

We also plan to move the library of pre-defined equations to HTML pages that can be created and browsed using standard tools. Each page will provide additional information about an equation to provide guidance, context, and help in making variant equations. This will also enable the library to be scaled to a large number of equations.s

References

Barnes, D. (1976). *From Communication to Curriculum*. Harmondsworth: Penguin.
Barnes, D., & Todd, F. (1995). *Communication and Learning Revisited*. Portsmouth, NH: Heinemann.
Edwards, J.-A. (2005). *Exploratory talk in peer groups - exploring the zone of proximal development.* Paper presented at the CERME4 Working Group 8, Sant Feliu de Guíxols, Spain.
Harel, I., & Papert, S., eds. (1991). *Constructionism*. Norwood, NJ: Ablex.
Kafai, Y. (1995), *Minds in Play*, Lawrence Erlbaum Associates.
Kafai, Y., & Resnick, M., eds. (1996) *Constructionism in Practice*. Mahwah, NJ: Lawrence Erlbaum.
Mercer, N. (1995). *The Guided Construction of Knowledge: Talk amongst Teachers and Learners*. Clevedon: Multilingual Matters.
Papert, S. (1980). *Mindstorms*. New York: Basic Books.
Quintana, C., Reiser, B. J., Davis, E. A., Krajcik, J., Fretz, E., Duncan, R. G., Kyza, E., Edelson, D. and Soloway, E. (2004) A Scaffolding Design Framework for Software to Support Science Inquiry *Journal of the Learning Sciences*, **13,** 337-386. http://www.leaonline.com/doi/abs/10.1207/s15327809jls1303_4
D. Shaffer, K. Squire, R. Halverson, J. Gee (2005), Video games and the future of learning, *Phi Delta Kappan, 87*(2), 104-111
Slaughter, A. and Strohecker, C. (2001) *Framework for Microworld-style Construction Kits*, Proceedings of EdMedia, World Conference on Educational Multimedia.
Soloway, E., Guzdial, M. and Hay, K. (1994) Learner-centered design: the challenge for HCI in the 21st century *Interactions*, **1,** 36-48. http://dx.doi.org/10.1145/174809.174813
Strohecker, C., & Slaughter, A. (2000). Kits for learning and a kit for kitmaking. Extended Abstracts, CHI'2000.
Tufte, E. R. (1983). *The Visual Display of Quantitative Information*. Cheshire, CT: Graphics Press.
Tufte, E. R. (1997). *Visual Explanations*. Cheshire, CT: Graphics Press.
Vygotsky, L. (1978) *Mind in society: The development of higher psychological processes*, Harvard.

Winters, N. and Price, S. (2005). Mobile HCI and the learning context: an exploration**.** *Context in Mobile HCI Workshop at MobileHCI05*, Salzburg, Austria. http://www.lkl.ac.uk/niall/context-mobilehci-crc.pdf