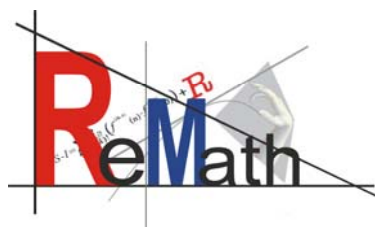


**Information Society Technologies (IST)
Programme**



Project Number: IST4-26751

**ReMath
Representing Mathematics with Digital Media**



Refined Version of the Dynamic Digital Artefacts

Deliverable Number: D16
Due date of deliverable: 30.11.2008 (month 36).....
Actual submission date: 19.12.2008.....
Start Date of Project: 01.12.2005
Duration: 42 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 Ricerche, Piazzale Aldo Moro 7, 00185, Roma, Italy,
4. Università degli Studi di Siena, Banchi di Sotto 55 , 53100, Siena, Italy,
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:

*The page numbers refer to the global page numbers (at the top of the pages).
The page numbers of the bottom the pages are local page numbers.*

Introduction.....4

DDA 1 Casyopee.....5

DDA 2 Aplusix.....23

DDA 3 Alnuset.....44

DDA 4 MachineLab.....65

DDA 5 Cruislet.....90

DDA 6 Mopix.....106

Annex I Internal Review.....129

Introduction

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

Year three has been devoted to the stabilization of many existing components (bugs fixing, small evolutions of the interfaces and the functionalities). Certain evolutions were suggested by students or teachers, others were ideas of the design teams.

For all DDAs, major evolutions of a few modules were made: some of them were planned from the beginning, others appeared as necessary during year 2.

The major evolutions are listed below:

Casyopée: important improvement of the geometric module and the measure panel; link between the symbolic and geometrical windows; change of the underlying Computer Algebra System to Maxima.

Aplusix: graphical representation of the solution sets of equations, inequations and systems of linear equation of two unknowns; representation of two steps in the second view of the tree module.

AlNuSet: Addition of a bi-dimensional editor, of a function component, and of communication system with the MathDiLS platform.

MaLT: change of the 3d game engine to the Java based JMonkey which speeds up the drawing. Inclusion of a set of ready made stereo-metrical objects, a special GUI to dynamically manipulate important features of these objects and dynamic controllers for camera movement.

Cruislet: immediate visualization of multiple linked representations; introduction of two options: geographical coordinates and spherical coordinates; development of a 3D controller representation of the spherical coordinates; development of new Logo programming primitives.

Mopix: re-implementation of MoPiX using the Google Web Toolkit; important modification of the interface and the functionalities.

Current versions of the DDAs are stabilized prototypes which have been tested with students.

DDA 1: Casyopee

Authors: JM Gélis, JB Lagrange

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

Table of content

Introduction	2
1. Last evolutions of the DDA.....	4
1.1. Enhancing capabilities of the geometric module	4
1.2. Enhancing the design of the measure panel	5
1.3. Linking symbolic and DG windows of Casyopée.....	7
.....	8
1.4. Changing the underlying Computer Algebra System	8
1.5. Unifying interface	8
2. History of the design and development of the DDA.....	9
3. Description of the final form of the DDA.....	9
3.1. The symbolic window of Casyopée	9
3.2. The geometrical window of Casyopée	12
4. Perspectives.....	16

Introduction

Casyopée is a Learning Environment that deals with algebraic functions and geometrical dependencies. Modelling geometrical dependencies by way of algebraic functions, and studying the geometrical properties of curves representing functions are essential mathematical activities that Casyopée makes very accessible for students. They allow students a better link between syntactical and semantic points of view in algebra and an easy introduction into calculus.

Casyopée has two main windows. The first one, (called the symbolic window) provides students with symbolic computing and representation capabilities as well as facilities for proving. The second one consists in a Dynamic Geometry (DG) window. Note that Casyopée's two windows are closely linked. This is a difference with other systems like TI_inspire that have also two windows, but juxtapose an existing numerical DG system (Cabri geometre) and an existing symbolic system (Derive) with weak links between.

Real functions of one real variable are the central objects of Casyopée. A function is defined by a formula involving a function variable and a domain. As most other symbolic systems related to functions and numerical graphers define functions over the whole set of real numbers, without regard to the existence of formulas, this definition is a distinctive feature in Casyopée. It allows being consistent with the mathematical definition as well as providing realistic modelling: when designing a function as a model of a situation, often the function is not defined on the whole set of real numbers and often not on the whole set of existence of the formula. Casyopée provides means for creating sets of ordered real numbers, possibly including parameters, in order to define domains. These parameters can be treated both formally and numerically by way of animation. Constraints can be set on parameters in order to adapt to all situations: for instance if the parameter is intended to model a measure, it can be defined as positive. Functions can depend on parameters. Expressions (that is to say formulas not involving a function variable but possibly involving parameters) can also be defined and treated. Thus Casyopée treats in a consistent way the algebraic objects generally included in upper secondary curricula about functions.

A wide range of construction capabilities is available within the DG window to build a figure including free points. Curves of functions can be drawn using the algebraic definition of functions (domain and formula). Because Casyopée is a DG system based on an underlying symbolic kernel (the free software Maxima in the refined version), it offers constructions, like the intersection of a line and a curve, and the facility for exporting geometrical functions or expressions that are not provided by existing DG systems based on numerical calculations. Measures can be defined as “geometrical calculations” possibly including symbolic objects (parameters, functions, expressions...) created in the symbolic window. Casyopée can compute a domain and a formula for “geometrical” expressions or functions related to measures, providing a capability to express algebraically geometrical dependencies. This “export” capability that will be explained below is of a great help for students when modelling algebraically geometrical functional dependencies or expressions.

The kernel is now Maxima that can be freely installed on Windows based systems (XP or Vista) together with Casyopée. The installation process requires only to download and execute a 8 Mbytes installer. Like with most software based upon the communication of two modules (here Casyopée and the Maxima kernel), some difficulties can be experienced when installing on computers for use without administrator rights, or with Data Execution Prevention, due to protections inside Windows. We are preparing a list these possible difficulties and remedies. A download page is available at <http://casyopee.eu>

We recapitulate the main capabilities available in Casyopée's symbolic window:

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

We recapitulate the main capabilities available in Casyopée's DG window:

- (1) geometrical construction (allowing to build figures, including free points on objects such as segments, circles, straight lines...); Casyopée's algebraic objects –functions, expressions and parameters- can be used. For example, parameters and functions can be involved in geometrical objects definitions (eg a coordinate points defined by its two coordinates, expressed as numerical values or values including parameters);
- (2) creation of geometrical calculations (well-formed formulas involving measures and symbolic objects);
- (3) numerical explorations of measures;
- (4) capabilities to determinate and link independent and dependant measures that will be used to define “geometrical” functions;
- (5) “exportation” of these geometrical functions into the symbolic window of Casyopée;
- (6) “exportation” of geometrical expressions of measure depending on no free point into the symbolic window of Casyopée.

Figure 0 recapitulates Casyopée's architecture.

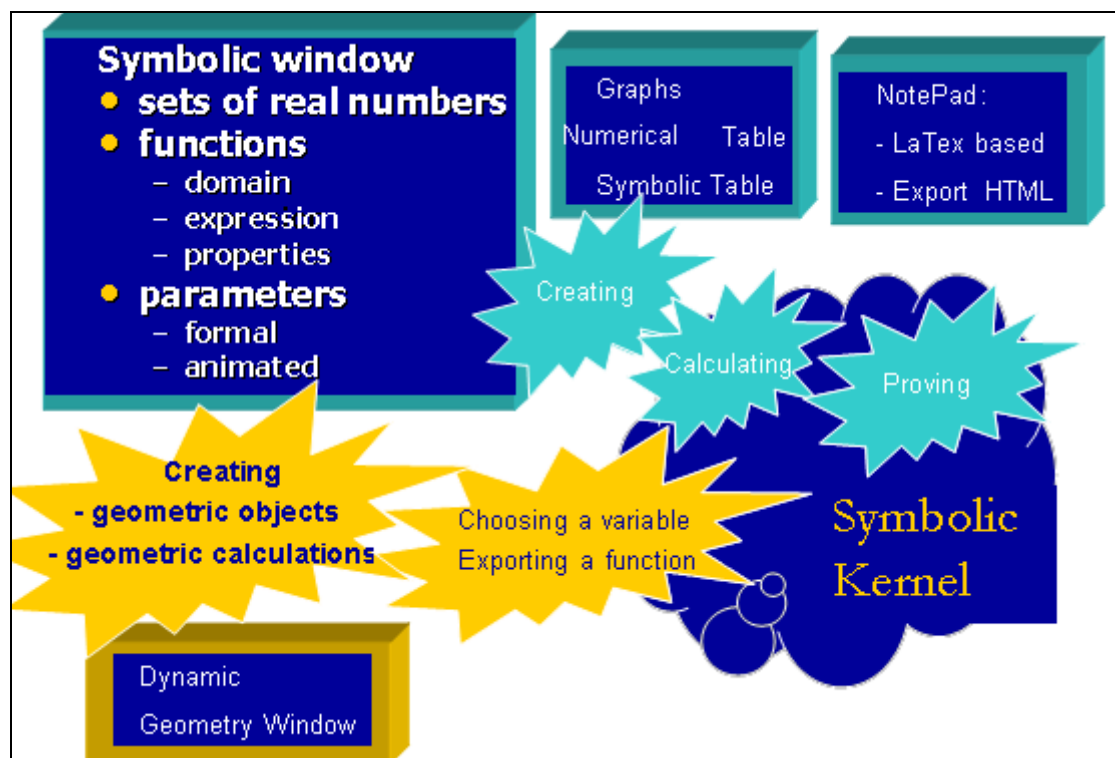


Figure 0: Casyopée's architecture

The following paragraphs are illustrated with examples of sessions. The current version is used, that includes the latest implemented capabilities. The language of this version is still French, all messages being not translated in English at the moment (see perspectives).

1. Last evolutions of the DDA

We describe below the main recent evolutions of Casyopée.

1.1. Enhancing capabilities of the geometric module

The first version of the geometric module developed in the first year of the project was basic and could only partially be used to solve the problems that teachers want usually to give their students. Consequently, geometric construction was strongly extended to cover a wide range of objects. For example, circles, curves and all associated facilities have been implemented.

These new extensions had to be consistent with the geometrical objects previously defined. It is now possible to specify free points on circles and curves and to define intersection points between two circles or between straight lines and circles or curves. This development does not consist only in drawing capabilities, but includes an important part of underlying formal calculation necessary to allow defining measures and geometric functions, and also to handle geometric objects like for instance the intersection between a curve and a line.

The figure 1 illustrates these new geometrical possibilities. It is now possible, with the geometric module of Casyopée, to build a figure involving circles. The situation includes a free point M on a segment [oA]. Two equilateral triangles (oMP and AMR) are defined and Casyopée offers unique means to study the variation of the distance PR. User can follow its numerical variations in the measure panel and remarks that the minimum value is reached when M is at the middle of [oA] (and when (PR) is parallel to (oA)).

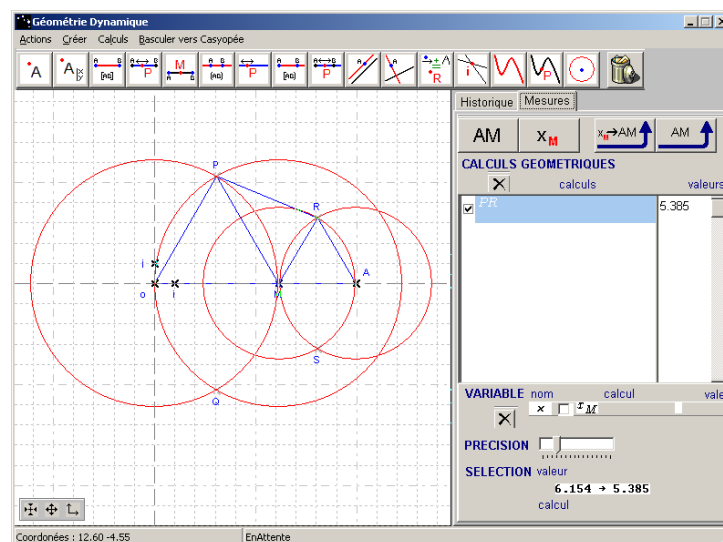


Figure 1 : Example of a situation involving circles.

Figure 2 involves 2 functions (whose expressions are $x \rightarrow \exp(x)$ and $x \rightarrow \exp(-x)$) and their curves. A free point M is defined on the first curve, a second one (N) is constructed as the intersection of the second curve and of the parallel to the axis (oj) passing through M. The respective tangent lines to each curve in M and N can be proved to be perpendicular. A

conjecture can be done simply by creating the geometric calculations MN^2 and $NP^2 + PM^2$ and remark that the numerical values of these two expressions are equal when dragging M all along the first curve. An algebraic proof can be made by choosing a variable related to M and exporting the functions corresponding to the two calculations into the symbolic window. The next paragraph gives an idea of how is it possible within Casyopée.

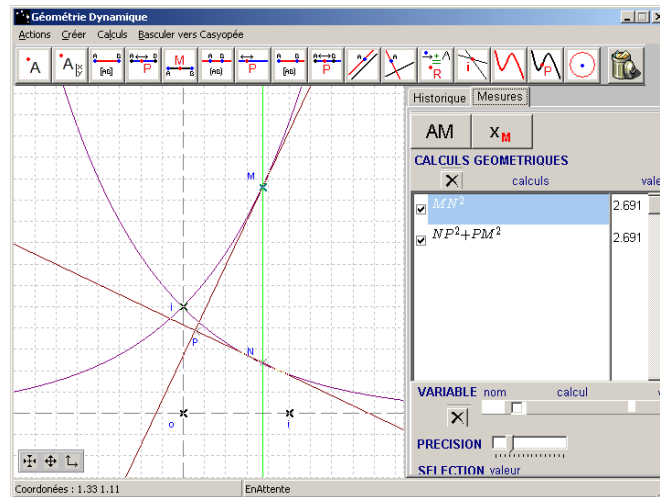


Figure 2: Situation including 2 curves (of functions $\exp(x)$ and $\exp(-x)$) and 2 tangent lines it is possible to prove that they are perpendicular.

1.2. Enhancing the design of the measure panel

After the first year of the project, the measure panel has been modified in order to be easier to understand and to activate. The main evolutions consist in new dispositions (based on combo-boxes) that aim at displaying all available possibilities (eg when setting up a variable). A new button has also been added, allowing exporting expressions. The following paragraphs describe the final result.

This panel displays four main buttons. The first one allows creating geometric calculations, the second one choosing a variable (that is to say a distance, an abscissa or an angle depending univocally of a free point), the third one exporting a geometric function (ie a function whose dependant variable is the selected calculation and the independent variable the chosen variable) and the last one exporting a geometric expression (ie an formula not depending on any free point).

Figure 3 displays a session including measure definitions. In this situation, A and B are coordinate points respectively on the y and x -axis, M is a free point on the segment $[oB]$ and (AM) and (MC) are perpendicular. The problem consists in studying the dependency of the area of AMC and of the distance AC with respect to the position of M .

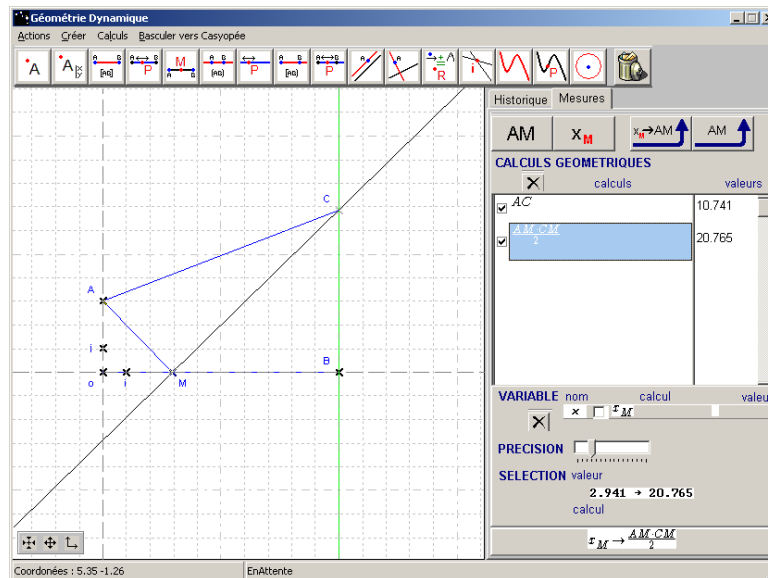


Figure 3: Session with measures definitions
(M is a free point on the segment [oB] and (AM) and (MC) are perpendicular).

Figure 4 gives an idea of the manipulation that allows defining a variable. The first proposed measure (distance between o and A) is not accepted by Casyopée because it depends on no free point. The second one (distance between M and A) is also refused because it does not depend univocally on the point M. The third one (abscissa of M) is adequate, but other measures like the distance between o and M or between M and B would also have been.

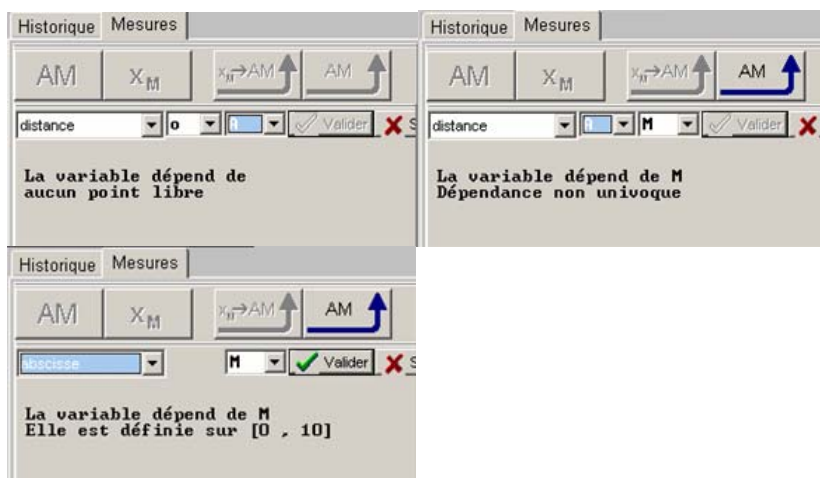


Figure 4: Different attempts to define a variable, the last one being successful.

The two latest buttons are called “export buttons”, because they allow exporting a geometrical function (if the chosen calculation is depending univocally on the chosen variable) or expression (if it depends only on fixed objects) towards the algebraic window. A special window (see Figure 5) shows all the relevant specifications when a function is going to be exported. Casyopée computes the formulas and the domain, that user can modify, if necessary.

As said above, this export capability is of a great help for students when modelling algebraically geometrical functional dependencies or expressions.

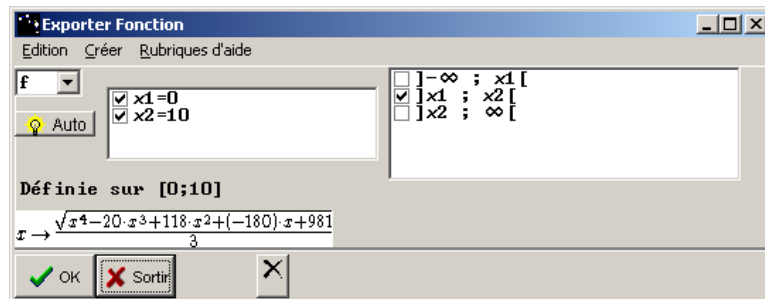


Figure 5: Special window for exporting a geometrical function.

1.3. Linking symbolic and DG windows of Casyopée

The symbolic and dynamic geometry windows of Casyopée are strongly linked. After exporting a geometrical function, this function can be studied in the symbolic window (for example in order to determinate its variation and its extrema) but it remains linked to the dynamic geometry window, especially by a dynamic link between this window and the graph of the function as explained by the example below. This feature is a recent evolution.

Figure 6 displays a session involving the two windows. The successive steps of this session are: (1) constructing the figure in the DG window, M being the only free point and the lines (AM) and (MC) being perpendicular; (2) defining measures of the hypotenuse and of the area of the right-angled rectangle AMC; (3) defining the independent measure variable x_M ; (4) specifying the Geometrical functions related to the previous hypotenuse and area (note that formal expressions are displayed in Casyopée's symbolic window); (5) working on these functions to determinate their extrema and variations, by differentiating and computing zeros of the derivative.

The link between the symbolic window and the DG window is enhanced by enactive and dynamic relations between on the one side the graph of the functions (within the symbolic window) and on the other one the initial figure (within the DG window). More precisely, a cross can be marked on a graph of a geometrical function. Dragging this cross all along the graph makes the concerned free point of the figure consequently moving, ensuring that the place of the precise location of the cross on the graphical representation marks the exact configuration of the Geometrical figure. The reverse is also working. If one moves a free point on the figure, then the cross on the graph of a geometric function is adapting itself to be in coherence with the precise geometric configuration. This dynamic reciprocal link between the graph of a geometrical function and the geometrical figure greatly helps students make sense of the algebraic modelisation of geometrical functional dependencies.

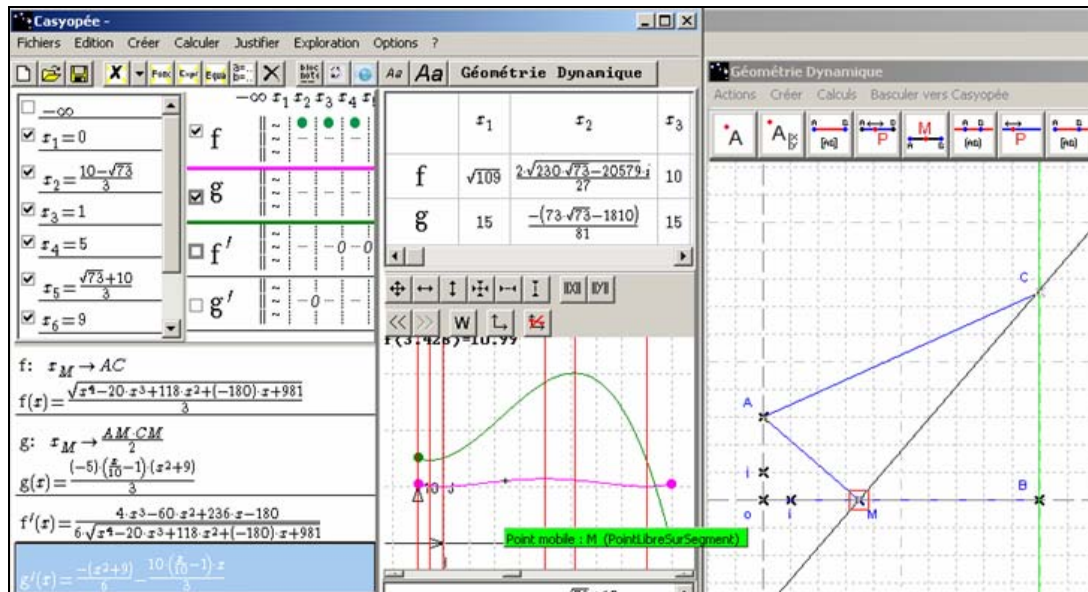


Figure 6: Example of link between symbolic and DG windows of Casyopée.

The work begins by constructing a figure (including a single free point M). Then, two geometrical functions are defined and exported into the symbolic window. Graphical representations in the symbolic window and figure in the DG window are dynamically linked. When the cross moves along one of the graphs, M moves accordingly in the DG window and reciprocally.

1.4. Changing the underlying Computer Algebra System

In the version for the cross experiment, Casyopée used MuPad as computer algebra kernel. This symbolic kernel is essential to Casyopée work. All transformations on formulas, explicitly requested by the user, or required by the functioning of the software are computed by the kernel.

The kernel is also involved in the geometrical module. All geometrical objects (eg points, straight lines, circles, curves...) have symbolic definitions that allow defining formal expressions of measures. These formal expressions are quite essential to define geometrical functions and their expressions (all computations are carried out by the kernel).

In view of a large dissemination and for licence reasons it was not possible to use MuPad any longer. Thus the Casyopée team choose another CAS kernel, Maxima mainly because it is free and, in consequence, will allow an easy dissemination. Substituting Maxima to MuPad was a considerable work because all calculus steps in Casyopée are strongly depending on the kernel's results that have to be interpreted to be useful for students or internal functioning.

1.5. Unifying interface

A particular attention was paid to Casyopée's interface. Users have to recognize easily all the working steps when using Casyopée and have to quickly understand what the environment is expecting. All entry dialog boxes (in the symbolic window as well as in the geometrical one) have the same presentation and offer users with the same buttons (numerical ones, constants, parameters, eventually points already defined...). Figure 7 illustrates this point.

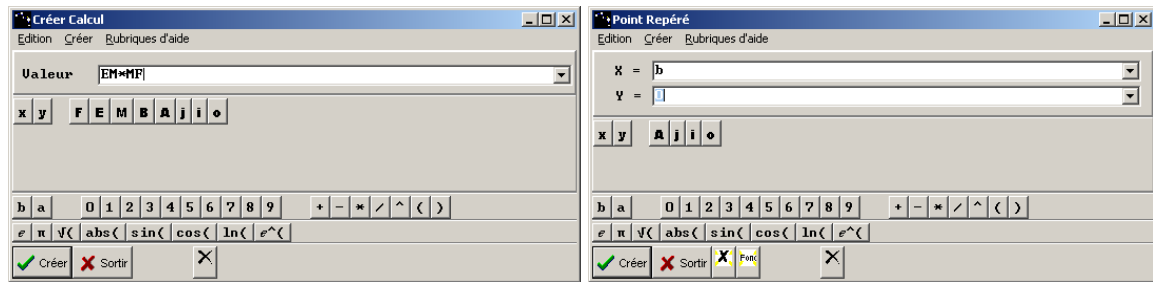


Figure 7: Dialog boxes sharing the same presentations

Parameters have been considered in the same way all over the environment. In some parts of Casyopée, parameters have to be instanced (for geometrical construction, for graphing functions). In others ones, parameters are formal and no particular value is considered, in order to allow students to obtain general formulas.

2. History of the design and development of the DDA

Two goals explain the evolutions of the design during these last two years. The first one is related to technical constraints, mainly the change of the underlying symbolic computer system that had serious consequences all over the DDA. The second goal was to enhance Casyopée's ergonomics, using feedbacks both from researchers and teachers.

The main steps of Casyopée development in years 2 and 3 of the project were:

- the change of the initial kernel (MuPad) to a new one, Maxima, as explained above;
- multilingualism (note that this point is not quite finished, last evolutions have not still be taken into consideration)
- the enhancement of the geometrical module, with capabilities for tackling all problems at upper secondary levels;
- the rewriting of some panels, such as the measures' panel, in order to be more efficient et clear;
- the constant preoccupation of having all over Casyopée a sufficient robustness and coherent behaviour;
- working on the dynamic organisation of links between objects, especially symbolic and geometrical; symbolic functions' and expression's definition can be modified by the user in order to allow correcting mistakes or exploring several forms, and parameters can be animated; the geometrical objects are dynamically updated to reflect these changes.

Note that each new version was regularly tested by our teachers' team and their students. Their remarks were taken into account and integrated if possible in Casyopée.

3. Description of the final form of the DDA

3.1. The symbolic window of Casyopée

Figure 8 displays the symbolic window. Its interface is made of different panels, each of them dealing with particular objects.

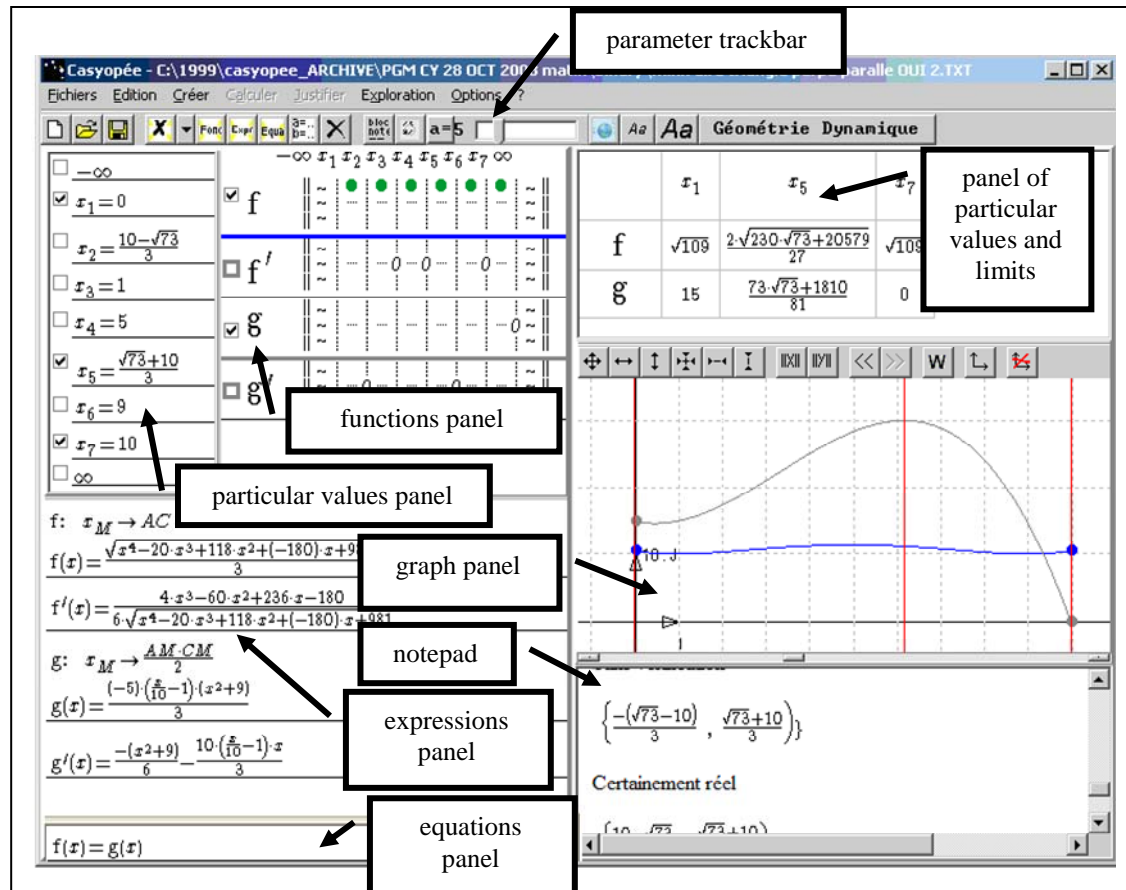


Figure 8: the main Casyopée interface.

Let us give some more precise explanations:

- **particular values panel**: this part of the window displays an ordered set of particular values of a function variable, entered by the user or computed by the environment (for example when automatically specifying geometrical functions); nine variable identifiers (x , y , t , ...) are available, that allows different sets of values.
- **expressions panel**: this part includes the formulas for all functions and expressions, geometrical ones exported from the DG window as well as functions directly created by the user; note that mere expressions can also be specified (expressions are considered by Casyopée as functions without function's variable); Figure 9 displays an example of a window that appears when creating a function: the user enters a formula and sets the domain set is by (un)checking items in the list (real values or open intervals);
- **equation panel**: equations of four types can be created and solved. 1) $f(x)=k$ 2) $f(x)=g(x)$ 3) $f(x)=k \cdot g(x)$ 4) $k=0$, where f and g are functions and k is a formula possibly depending on parameters. Casyopée solves the first three types in x , and the fourth in a parameter. The existence and consistency with the domain of the functions and parameters are checked by the kernel.
- **functions panel**: a table including displays the properties (sign, variation) of the functions after they have been proved or admitted.;
- **particular values and limits** of functions for chosen values are also available on demand in the right part of the window;

- **graph panel:** this panel contains graphs of selected functions; a lot of functionalities help the user to obtain relevant representations: it is possible to zoom, translate, modify the axis scale, resize the representation of the window with the mouse...
- **notepad:** all the actions activated by the user are recorded in this part using Latex. Users have consequently to their disposition a draft including arguments of their action. At the end of their work, users can reorganise this NotePad in order to obtain a well written report and/or exporting this as an html file;
- **parameters' trackbars:** they are used to animate parameters, that is to say to change the value of the parameters in non symbolic calculations. Constraints on parameters (intervals) can be modified by a special menu entry.

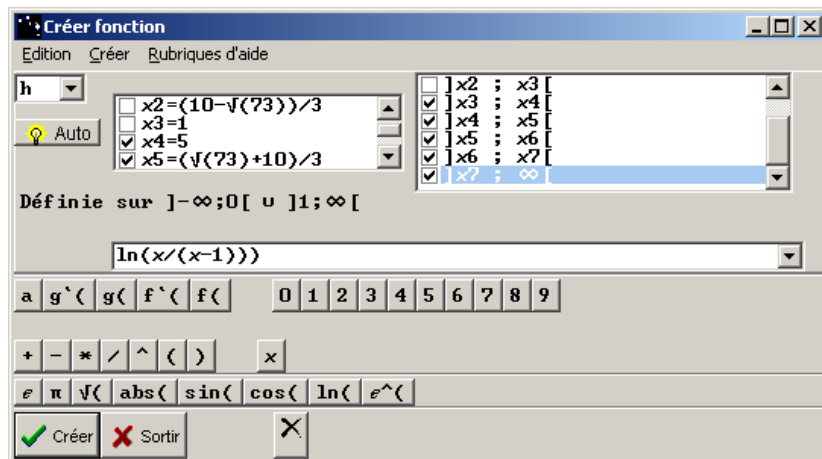


Figure 9: Example of creating a function.

Casyopée includes facilities to prove or admit properties of functions with the help of theorems (Figure 10). Students have to express the conditions for application of one of these theorems before applying them. Figure 10 displays the steps required when applying properties in order to prove the sign of a quotient. The property and the theorem have to be chosen in the corresponding menu. Successive entry boxes allow then specifying the conditions for applying the theorem.

Figure 10 shows also the four types of equations that can be created and possibly solved by Casyopée.

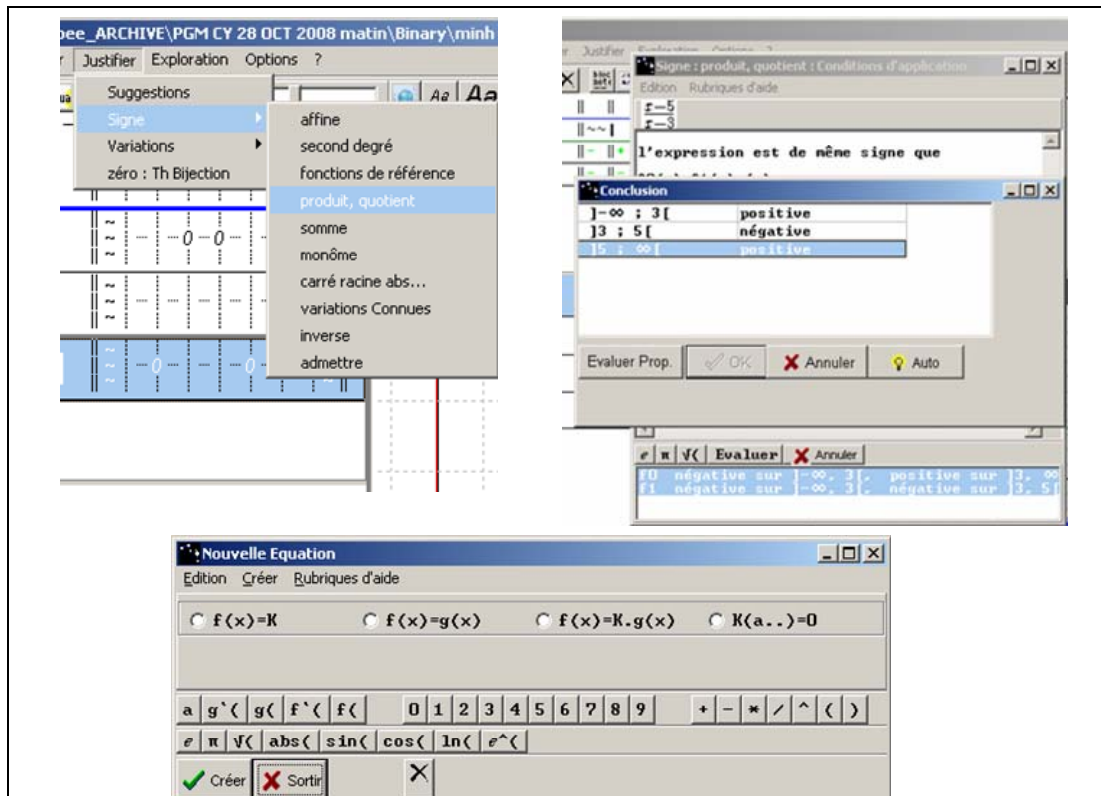


Figure 10: Others functionalities of Casyopée: (1) selection a theorem; (2) applying a theorem; (3) defining an equation.

3.2. The geometrical window of Casyopée

As mentioned above, Casyopée provides students with dynamic geometry capabilities like in other software, but also specific features helping students make sense of the algebraic modelisation that was the objective of Casyopée in ReMath. A figure constructed within Casyopée may for example include:

- **different types of points:** free points (belonging to circles, curves, straight lines.. or free in the plan), points defined by their coordinates, various intersection points (between straight lines, or circles, or curves). Figure 11 displays different possibilities visible in the menu. Note especially the possibility of drawing curves of function defined on a domain (not necessarily the whole set of real numbers) and of creating an intersection point between a curve and a line.

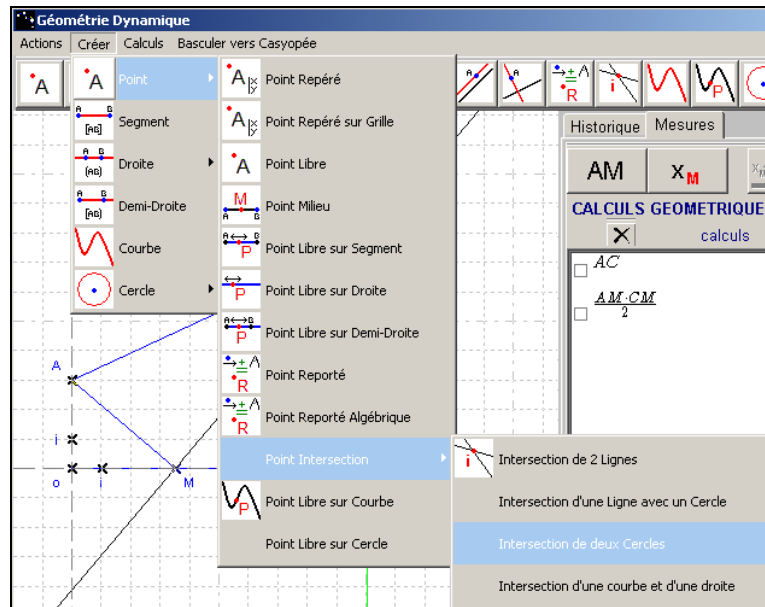


Figure 11: Different possibilities for creating points.

- **various geometrical objects** (see Figure 12), such as circle (defined by different means), curves, segments, straight lines including parallel and perpendicular ones or tangents to a curve. Thanks to the symbolic definition of geometrical objects, Casyopée can easily know whether a point belongs to a line, a circle or a curve, or test the equality of objects.

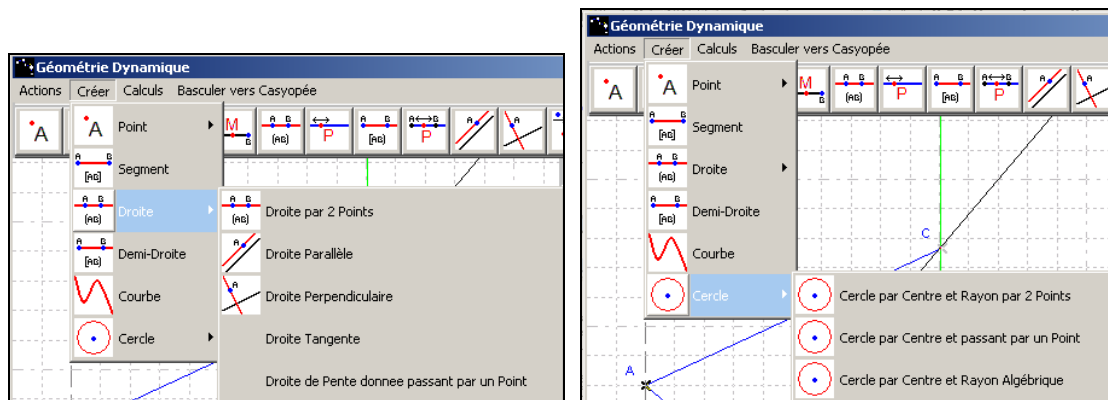


Figure 12: Others geometrical objects.

All the previous constructors allow users to build figures that can be moved or transformed by dragging any free point. Figure 13 is redisplaying a previous example (see Figure 3) where M is a free point on the segment [oB] and (AM) and (MC) are perpendicular. Note that an history of the figure construction is always available.

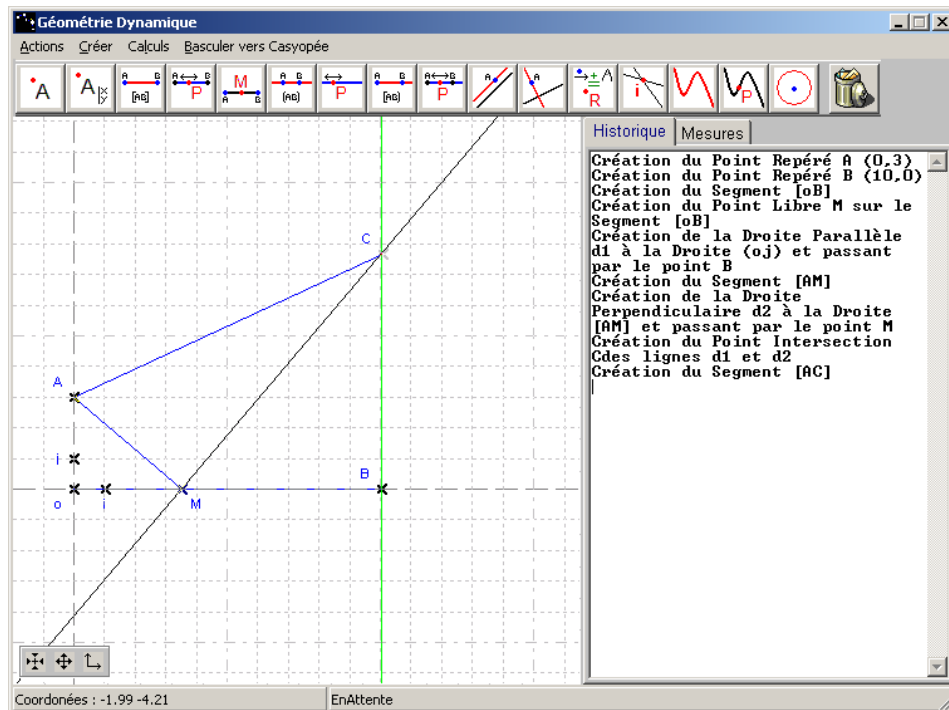


Figure 13: A dynamic figure example
(M is a free point on the segment [OB] and (AM) and (MC) are perpendicular).

The geometrical window of Casyopée includes also the measures panel that allows creating geometrical calculations, choosing a variable, exploring (co-)variations and exporting geometrical expressions and functions.

A geometrical calculation is a well-formed formula that contains distances or coordinates of points and possibly algebraic objects (parameters, expressions, functions...)

As explained above, if a geometrical calculation depends on no free point, then it can be exported as a geometrical expression. Casyopée checks the non-dependency and computes an algebraic expression, possibly involving parameters and functions.

A variable is a particular measure, more precisely a distance between 2 points or any coordinate of a free point, or a coordinate of a vector, or an angle in case of a free point on a circle. A valid variable depends univocally on a free point. Internally, this is checked by Casyopée by verifying that the variable can be expressed as a reversible function of a geometrical hidden parameter - emanating from a free point.

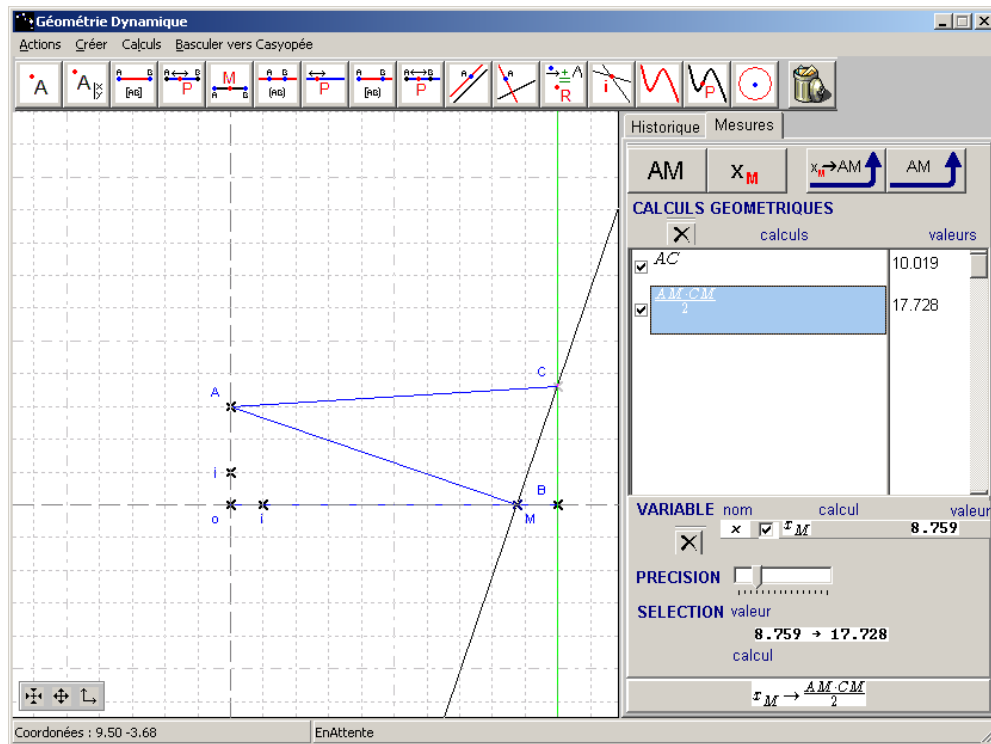


Figure 14: Example of measures definition and selection.

As explained above, when a valid variable is chosen and a geometrical calculation is selected, then it defines a geometrical co-variation that can be explored numerically. If it is a consistent dependency, (that is to say if the dependant measure is only depending one the geometrical parameter specified in the variable measure) it can be exported as a geometrical function towards the symbolic window. Casyopée verifies the consistency of the dependency between the 2 measures and if consistent, computes and exports the relevant function.

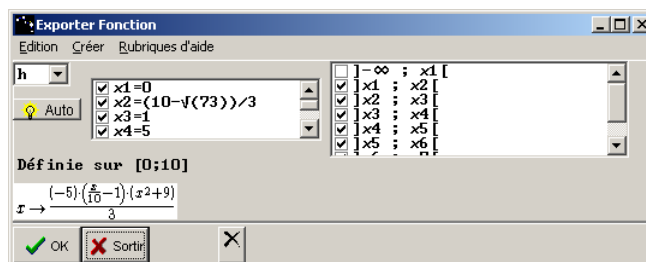


Figure 15: Example of exportation of a geometrical function.

This exportation of a geometrical function or expression is designed to help students during the modelling process. It allows them to concentrate on strategic decisions, like choosing a variable or creating a calculation even when they have not great ability in algebraic manipulations. In ordinary settings (without Casyopée, or with another software), the students cannot operate themselves this part of the modelling process.

4. Perspectives

Casyopée has now reached its general objectives in ReMath with facilities to help students in the process of algebraic modelling and handling of associated representations. In relationship with these objectives, we recapitulate here the main choices and features that make Casyopée different from other existing software more or less dedicated to functions or from DG systems, and justify the efforts made in the ReMath project.

The first choice has been to create a DG window deeply linked to the existing symbolic window. A most noticeable consequence is the facility for exporting geometrical expressions and functions (i.e. creating an algebraic object from a geometrical formula or dependency). It allows a student to concentrate upon crucial decisions when modelling a geometrical dependency (choice of a geometrical calculation as a dependant variable and of a measure as an independent variable) without being distracted by algebraic manipulation, and to explore or calculate concurrently the dependency in the DG window and on the exported corresponding function in the symbolic window. No other software provides this exportation of a geometrical function or expression and this dynamic link between the two windows. It greatly helps students make sense of the algebraic modelisation that was the objective of Casyopée in ReMath.

Another important consequence is that all algebraic objects can be used when creating geometrical objects. Thus, for instance, generality in a geometrical figure can be expressed by using Casyopée parameters. As another example, curves of functions have been implemented as geometrical objects totally consistent with the algebraic definition of the functions. For instance, it allows creating intersection points between a curve and a line, a facility that numerical dynamic geometry environments do not provide and which is important when studying geometrical properties of curves, an important topic at upper secondary level.

The second choice has been to redesign Casyopée around the free Maxima kernel. The kernel's symbolic power serves to offer facilities to the student (for instance, checking consistency of definitions, exploring algebraic forms without being distracted by symbolic manipulation, building an algebraic proof...). Choosing Maxima allows easy installation of Casyopée free of licence, and then wide dissemination of the associated ReMath results especially among teachers.

At month 36, Casyopée is fully operational except for the complementation of the English version and documentation. While the Mupad version for the ReMath experiment provided both English and French, French has been privileged in a first stage for the Maxima version in order to perform classroom tests. Creating versions in other languages (localisation) is easy and our team will complete the English version in the next two months. Considering the special links that our team developed with two universities in Vietnam, we also plan to make a Vietnamese version. We will study the possibility of developing a tutorial environment based on Casyopée's principles that could be more adapted for average teachers. Another work to do is to enhance the notepad in order that it could be more easily edited and exported by the students. MathML will be used as a standard to represent expressions, instead of LaTeX.

Disseminating Casyopée and Remath results among teachers in connection with the other Remath work packages is also an important perspective. We selected a region of France (Brittany) where local authorities are specially interested by Casyopée's features. A group of teachers has been set up with the help of the French INRP (National Institute for Pedagogical Research) and the IREM (Institute for Research in Mathematics Teaching) of Rennes. Although already interested by digital technology, these teachers were not involved before in

Casyopée's design and experimentation. We expect them to bring a fresh vision of Casyopée's features and to prepare scenarios of use that will be disseminated to other mathematics teachers together with Casyopée on the professional digital workspace for teachers in Brittany. We plan to set up a regional community of users, and later to extend then community to France.

DDA 2: Aplusix

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

UNIVERSITE JOSEPH FOURIER, Laboratoire Leibniz INPG-UJF-CNRS

Table of content

1)Introduction.....	2
2)Last evolutions of the DDA	4
2.1 Graph module.....	5
2.2 Tree Module	10
3)History of the design and development of the DDA.....	11
3.1 Tree-Representation	11
3.2 MathDiLS features	11
3.3 Graph module.....	12
4)Description of the final form of the DDA.....	12
4.1 Tree module.....	12
4.2 Graph module.....	16
4.3 MathDiLS module.....	18
5)Perspectives	19

1) Introduction

Aplusix is a software application that helps students to learn algebra.

With Aplusix, students can easily type in algebraic expressions, thanks to an advanced two-dimensions maths editor, and make calculations on these expressions. They can make as many calculation steps as they want in a way that is very close to their usual work with paper and pencil. Aplusix gives them an immediate feedback about the equality or equivalence of their calculation steps (Figure 1). Aplusix can validate the end of an exercise.

Aplusix has also lots of other features for students (map of exercises, test and self-correction activities...) and teachers (authoring tool, statistics...).

Aplusix has been designed by researchers in computer sciences and mathematic education.

The theoretical framework of the computer sciences side is made of:

- Mathematic and computer science theories (polynomial, functions, equations, etc. and their representations – rewriting rule theory),
- Human-computer interaction ideas leading to a two-dimensions editor of the manipulated objects and easy tools,
- Basic math education ideas leading to have parameters allowing to customize the application to the level of the students.

The computer sciences side does not include a particular learning theory, the application is supposed to be usable according to several learning theories used by teachers and researchers. The theoretical framework of the mathematic education side is made of the theory of Semiotic representation (Duval, 1993, 1995, 2006) and the anthropomorphic theory (Chevallard, 1992, 2006). The application has been developed to be particularly suitable for these theories.

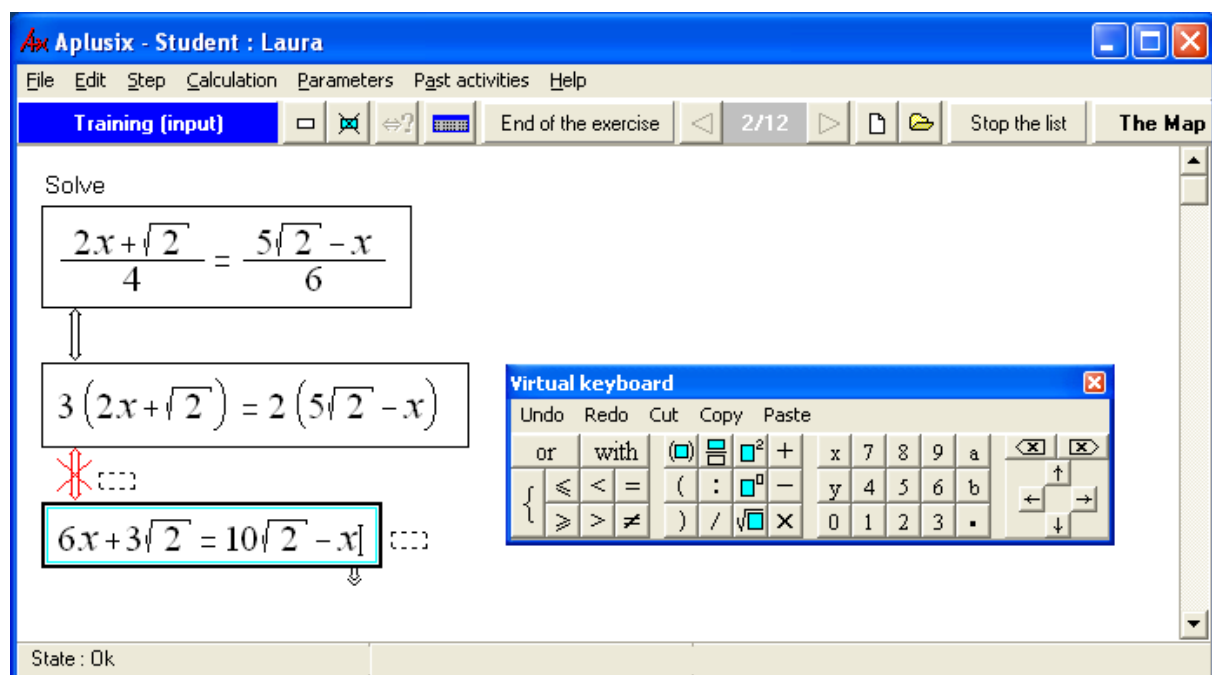


Figure 1: Solving an exercise with Aplusix: the first and the second step are equivalent, the third is not so Aplusix display a red cross as feedback

Lots of experiments in different countries in the world showed that Aplusix is very effective in learning algebraic skills because of its high level of interactivity.

Nevertheless, before the Remath project, it was using a single representation of the mathematical objects: algebraic expressions in their “natural representation”. This natural representation, which is the most used in classrooms, is not that easy for young students.

So the main aim of Aplusix in the framework of the ReMath project was to give students new means to understand the algebraic objects.

Two different ways were planned:

- one focusing on the algebraic structure of algebraic expressions: the tree module,
- one dealing with the meaning of algebraic expressions and their equivalence, by representing their denotations (the objects they represent) in a 2D space: the graph module.

The first one, the tree module, is dedicated to the tree representation of algebraic expressions. We can represent an algebraic expression with a tree¹ having operators as internal nodes and arguments of these operators as children, as in the following example (Figure 2).

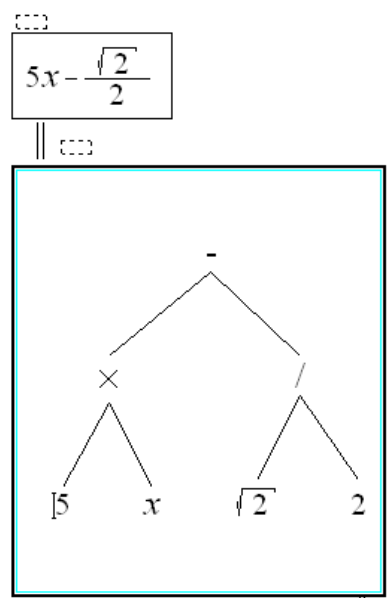


Figure 2: Example of display obtained with the Tree module

The second one, the Graph module, enables graphical representations in a 2D or 1D space of mathematical objects associated to algebraic expressions, i.e., their denotation. In the case of polynomial or rational expressions of one variable, the denotation is a function which is displayed in a 2D space.

In the graph module, each expression of one variable can be drawn as a curve or a straight line. Below the graphs, a legend indicates the equation corresponding to the curve (for

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.

example, if one decides to represent the expression $\frac{1}{2}x^2$ the legend will have an item indicating $y = \frac{1}{2}x^2$)

In the case of equations or inequations of one unknown, the denotation is a set of solutions which is displayed in a 1D space; an intermediary drawing of each equation of the form $f=g$ is represented by the graphs of f and g in a 2D space

Figure 3).

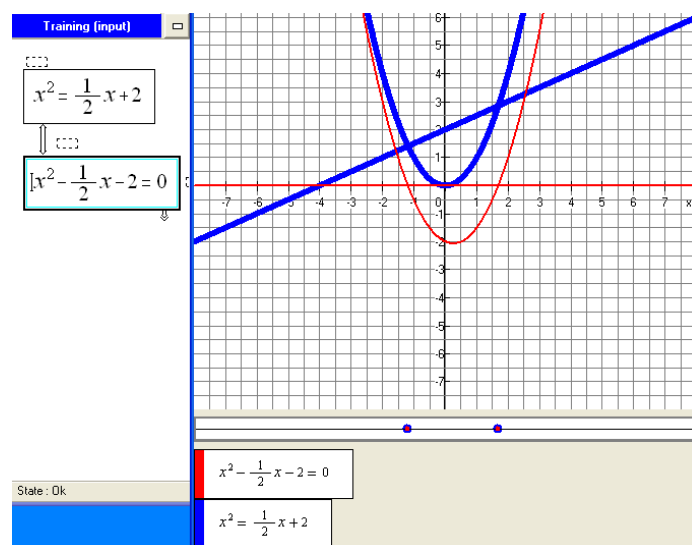


Figure 3: Example of display obtained with the Graph module. The sets of solution of the two equations are drawn on the real line (bottom of the figure). As the equations are equivalents in this example, the two sets are identical (two points of the real line). The above 2D figure is an intermediary drawing of the equations where each equation $f=g$ is represented by the graphs of f and g .

The graph module allows to select expressions listed in the legend and to change the colour, thickness and display order.

In addition a module for communication with webservices and specially MathDiLS has been developed. This communication mainly consists in browsing resources, open or save a file with the webservice.

2) Last evolutions of the DDA

The two main parts of the development made for the Aplusix DDA were the introduction of tree representations of algebraic expressions (display and editing) and of graphical representations of the mathematical objects represented by the expressions for several classes of expressions (display only).

Details on the history of the different developments are presented below in the section “History of the design and development of the DDA”.

2.1 Graph module

The part that has more evolved during the last months is the graph module. Indeed we added the support of domain of definition of functions, the graphical representation of equation and inequations of one unknown, and the graphical representation of systems of linear equations with two unknowns.

Rational expressions

Semantics of rational expressions have been changed to functions instead of rational fractions. This is more adequate for secondary schools. This means that conditions are sometime necessary to preserve equivalence, see Figure 4.

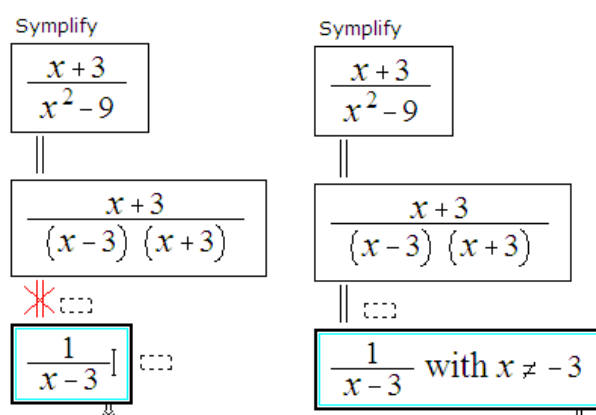


Figure 4: On the left, there is no equivalence because the expression of the third box is defined for $x = -3$ while the expression of the second box is not. On the right, there is equivalence thanks to the condition “with $x \neq -3$ ”.

With operator

The “with” operator has been added with the management of rational expressions of one variable.

It can be also used to define functions in a restricted domain.

The “with” operator has been added on the virtual keyboard (Figure 5).

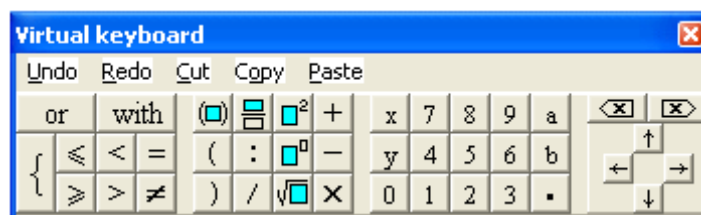


Figure 5: The new virtual keyboard of Aplusix

Management of discontinuity

We used the calculation of the domain of definition to draw discontinuous functions.

like $y = \frac{1}{x}$ for infinite limits.

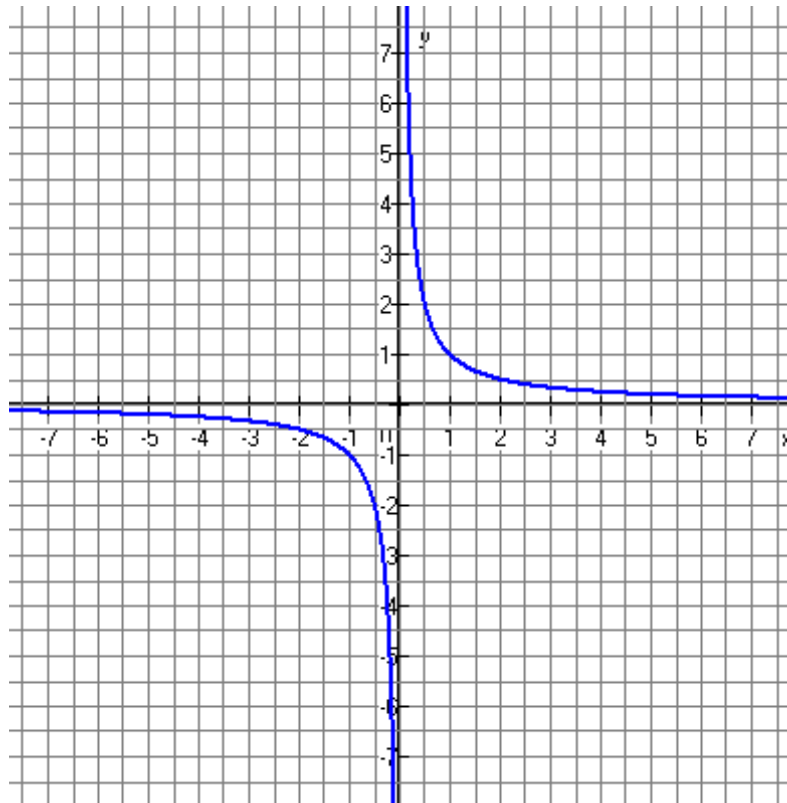


Figure 6 : Graph of $y = \frac{1}{x}$

In the case of finite limits, we implemented didactical representations used in France: a filled circle for a reached limit (see Figure 7), and an arc centred on the excluded point for an unreached limit (see Figure 8).

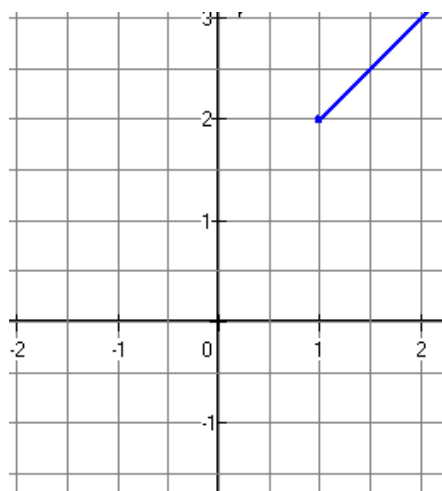


Figure 7: Inclusion

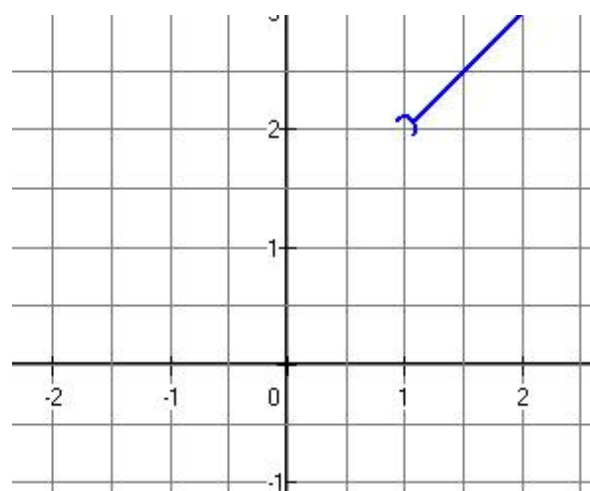


Figure 8: Exclusion

Equations, inequations and their sets of solutions.

We have added a graphical representation of the set of solutions for a given equation or inequation.

First of all the introduction of an equal operator change the way the curves are drawn and the legend.

When we introduce at the end of a polynomial or rational expression of one variable an equal sign and another expression (for example $\frac{1}{2}x^2$ becomes $\frac{1}{2}x^2 = -4x + 1$) the two curves are drawn, with the same colour and the same thickness. The legend indicates the equation $\frac{1}{2}x^2 = -4x + 1$ and all the modifications applied to the curves are applied for both curves of the equation.

In addition of the 2D-space representation, the new graphical module of Aplusix displays a representation of the set of solutions.

The set of solutions is represented on a straight line just below the grid used for the 2D drawing of curves.

The set of solutions is displayed but with no explicit message like “the set of solutions is ...” It should be seen rather as a way to compare two steps by comparing their sets of solutions than a tool that give an instant solution of the equation or inequation to solve.

Thus, one major issue of the graphics development is dealing with overlapping of graphical objects whether they are curves or solution points (Figure 9). We made the choice to change the thickness and the colour of the drawing as we have already done for the curves. Sure the reading of the value of the solution is less accurate but our main goal is that students see the superposition in the solution space as sign of the equivalence of the equations.

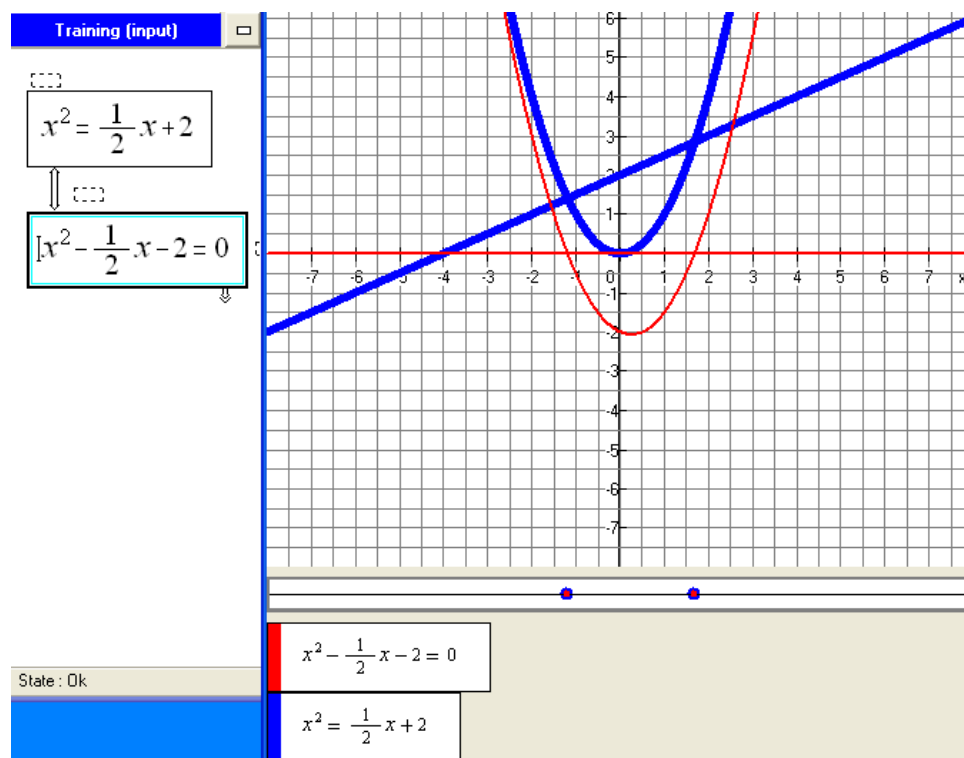


Figure 9: Display of curves and sets of solutions for two equivalent steps.

For the drawing of the set of solutions, there are two non exclusive cases:

- A finite number of solutions: in this case, we draw a point for each solution (Figure 10),

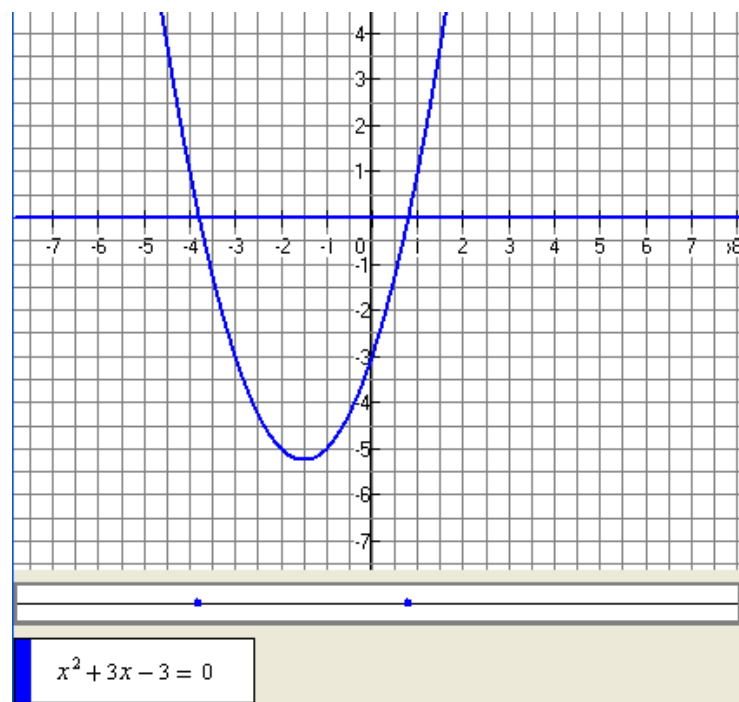


Figure 10

- An infinite number of solutions: in this case, we draw a part of the intervals with the same conventions for included and excluded extremities we use for curves. (Figure 11)

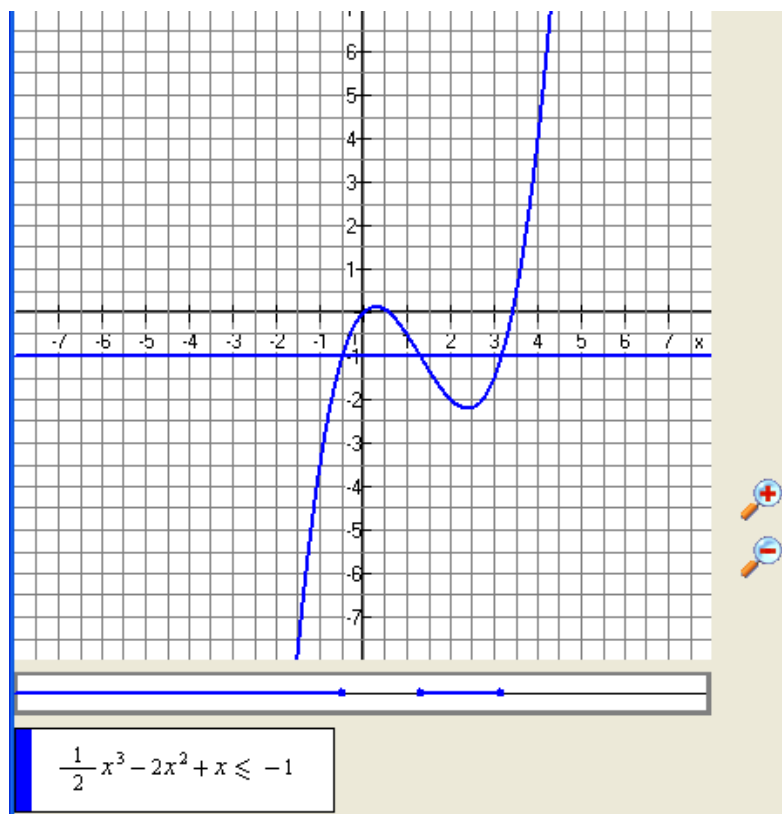


Figure 11

The curves and the solution drawing order can be changed by the user. The user can also change the thickness and the colour of the curve in order to better see what happens.

Linear Systems of equations

Linear systems of equations with 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.

When two equivalent systems with one solution are drawn on the graph, their solution is a same point. To make the superposition visible, the point is filled with the two colours of the straight lines representing each system.

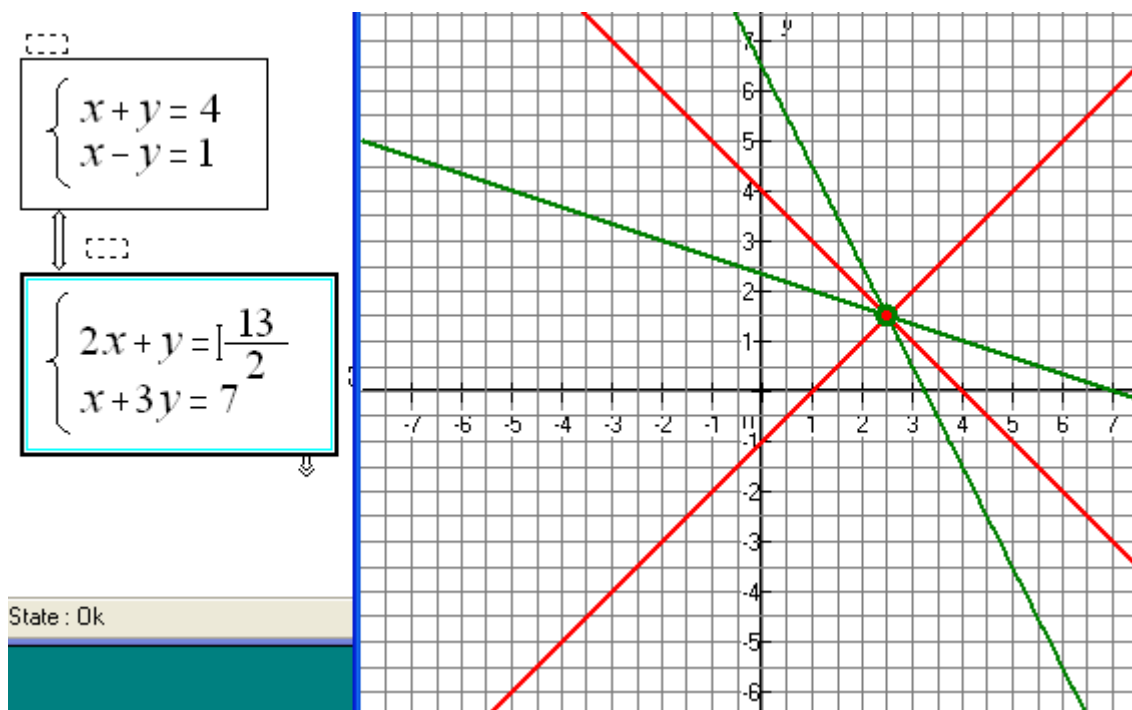


Figure 12 : Representation of two equivalent systems

Smooth transitions

As the graph and the steps it represents are synchronized, one important part of the development has been to ensure the transitions between expressions and equations or inequations. So the user can directly see the result of the changes in the graph. Aplusix adapts the graphical display to the nature of the represented expressions (whether it is an equation, inequation, a system or not). The expressions of the steps may be of different natures, although this has no real meaning, it is up to the user to understand what happens in such situation.

2.2 Tree Module

We describe here the latest development of the tree module. A detailed description can be found in the previous deliverables DEL4 and DEL8 of the WP2. An overview of the tree module is available on the third part of this document (Description of the final form of the DDA).

Operator drawing

Times and square roots operators have been redrawn to be the same as in natural representation (Figure 13). The old drawing was using the symbols “*” for the times and “sqrt” for the square root.

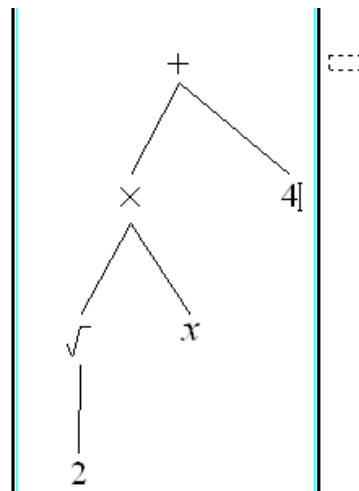


Figure 13

Second view

The second view has been extended to display both the current and the previous step. This has been done to allow the student to use the second view to compare the evolution they perform between the two steps, i.e., to see the evolution in two representations of their choice (Figure 14).

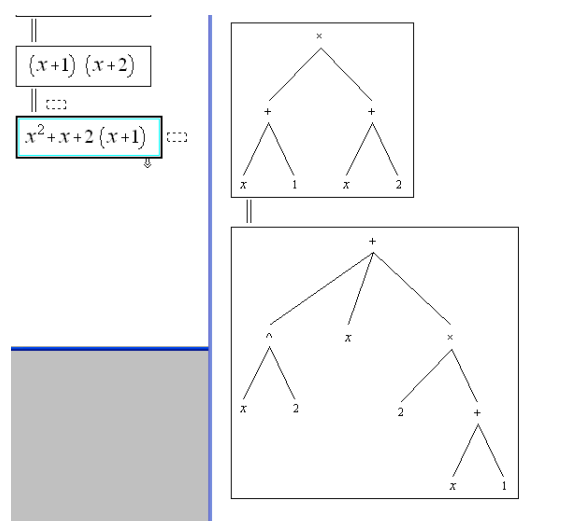


Figure 14

We have also fixed some bugs (replay system, parameters...).

3) History of the design and development of the DDA

3.1 Tree-Representation February-2006-May 2008

At the beginning of the project, we only defined two modes of interaction for the tree module. In these first specifications, tree representations were only available with correct operators and a correct arity.

We developed the tree representation of algebraic expressions, that is to say the way algebraic expressions were displayed as trees on the screen.

Students were not able to freely build trees. The didactician colleagues were very sensitive to this point, because in the expected building activities, students should be able to make such mistakes as erroneous operators or erroneous number of arguments.

So we introduced two other possible modes for editing tree representations: a free and a controlled mode. 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. That was probably the most difficult part of the work on the tree module, because we had to create a new way of building and handling trees. For that purpose, we made the choice of a direct manipulation editing for the tree representations. So the editing has been based on insertion drawings and direct addition by clicking when these drawings were available. Some additional features such as the insertion of a parent node with one child node made the building of trees easier and more natural.

As we implemented the 4 modes (usual, free tree, controlled tree and mixed) and the transitions between these modes, didactician colleagues of our team pointed out that it will be necessary to explain to students why they cannot change the representation of a given expression from the free tree representation to the controlled tree or usual representation. In order to clarify this explanation, we added a highlighting of the part of the tree that was syntactically incorrect.

We also added more editing and interaction abilities (like the second view of the current and previous steps) to improve the feedback to the user.

3.2 MathDiLS features January 2007-April 2007

Aplusix was a standalone application that could run on a local network and had no web communication abilities. So one of the first work was to learn more about the webservices technologies and to introduce in the project new components that could handle such communications.

We designed a general frame for webservices in order that Aplusix would be able to connect to different webservices and in this frame we developed a specific module for MathDiLS.

First we have implemented the connection, authentication and simple data retrieval like the list of DDA.

Then we implemented the functions that enable authoring and use by students: browsing categories, opening and saving features and creating categories.

Finally, we have implemented import and export of content MathML.

3.3 Graph module January 2008-November 2008

First we designed a graphical representation of steps on continuous functions.

During this phase, we developed such features as pan, zoom and resize of the graphical part of the window.

Then we considered discontinuous functions, so we worked on drawing of a function by intervals with included or excluded bound values.

Finally, we worked on the representation of equations, inequations and systems. We implemented the drawing of the set of solutions.

We have also a lot of work on transitions between the representation of a function and the representation of an equation, as the graph module and the main editing window of Aplusix are synchronized.

4) Description of the final form of the DDA

This description of the final form of Aplusix deals with the modules developed during the Remath project.

4.1 Tree module

We present here an overview of the main features of the Tree module.

For a more complete description of the module see the “Aplusix-Tree: User manual” document which is a part of the DEL8 of the Remath project.

Objectives

The starting point of the tree module was the hypothesis 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.

So the objectives of the tree module in order to test this hypothesis were:

- 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.

To reach these objectives we defined three representations systems and four interaction modes.

The tree module manages the following three representations systems:

Usual representation (also called **natural representation**) system: the one which is used by teachers and students on paper and board.

Example: Figure 15

$$2\left(-\frac{3}{4}x + 5.21\right) = 5x - \sqrt{2}$$

Figure 15 An equation in usual representation system

Tree representation system: In this system, operators (+, −, *, /, ^, sqrt, =, ≠, <, ≤, >, ≥, and, or, not) are placed in the internal nodes of a tree while variables and numbers are placed in the leaves.

Example: Figure 16

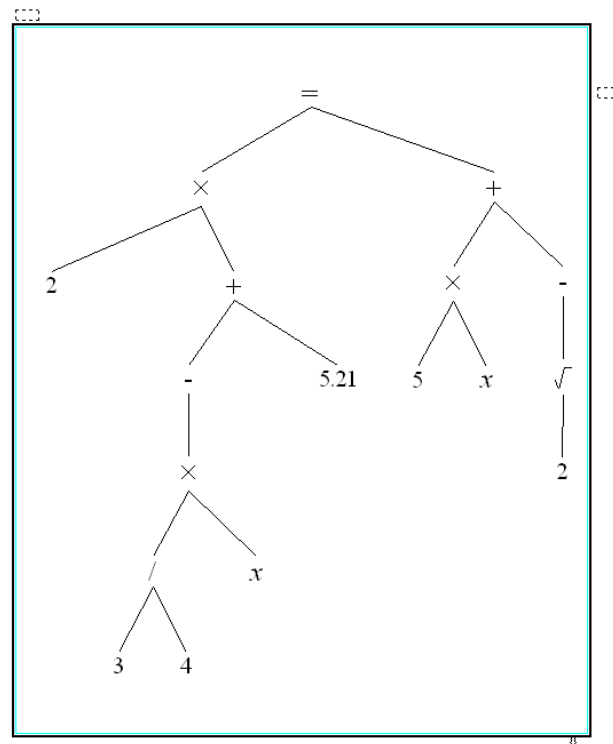


Figure 16 The same equation in Tree representation system

Mixed representation system: it is a tree representation in which leaves can contain usual representations. Each leaf that contains a usual representation which is not a variable or a number can be expanded in order to obtain its tree. Reciprocally each leaf that contains a tree representation can be collapsed in a usual representation.

Example:

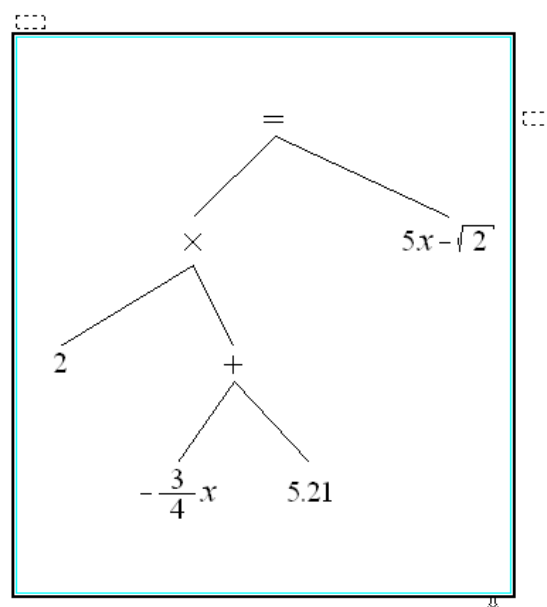


Figure 17: The same equation in mixed representation system

Four interaction modes have been implemented in the module:

To understand the differences between these modes it is necessary to define some terms.

We consider three notions linked to algebraic expressions:

- well-formed expressions,
- quasi-well-formed expressions,
- ill-formed expressions.

The notion of well-formed expression comes from the definition: it is an expression in which operators have expressions as operands, with the right number and the right types. Ill-formed expressions are objects that look like expressions, but do not respect the definition on some points. It is necessary to represent and manipulate ill-formed expressions because: (1) one cannot build a well-formed expression without intermediate ill-formed stages; (2) one cannot expect that students will always produce well-formed expressions.

We call quasi-well-formed expressions, expressions in which places for operators contain correct operators and the operators have a correct number of operands, including empty operands represented as empty boxes or question marks or something else.

From the beginning, Aplusix uses the “usual representation” of algebraic expressions and quasi-well-formed expressions: when an operator is recognized, a correct number of operands is assigned to this operator by adding, when necessary, empty operands (question marks). This is a design choice. It implements a scaffolding mechanism: the student cannot build any sort of ill-formed expressions.

For each representation system, one can choose to implement an editor which allows any sort of ill-formed expressions or which limits its field to quasi-well-formed expressions.

Usual representation = Usual representation + Quasi-well-formed expressions.

Free tree representation = Tree representation + Ill-formed expressions (any sort): one can put any string of characters in the nodes of the tree; one can give any number of successors to a node of the tree. (Figure 18)

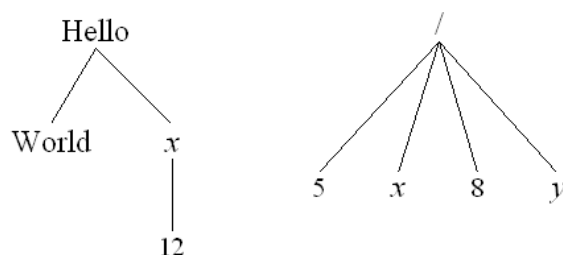


Figure 18: Example of ill-formed trees which are not quasi-well-formed: on the left, internal nodes do not contain known operators; on the right, “/” has 4 operands.

Controlled tree representation = Tree representation + Quasi-well-formed expressions: one can put only known operators in the internal nodes of the tree; one can put only variables and basic numbers (sequences of digits with possibly a minus sign before, with possibly a decimal separator inside) in the leaves of the tree; the correct number of successors of a node is maintained by the software.

Mixed representation = Mixed representation + Quasi-well-formed expressions. This is similar to the Controlled tree representation except that:

- leaves may contain any sort of quasi-well-formed expressions in the usual representation system,
- “+” and “-” buttons allow expanding and collapsing the tree (Figure 19).

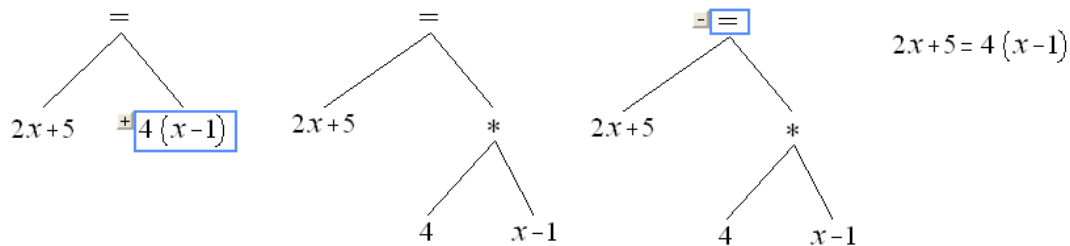


Figure 19: A click on the “+” button of the 1st tree expands the node, providing the 2nd tree.
 A click on the “-” button of the 3rd tree collapses the node, providing the 4th representation (usual one).

Each mode has its own editing properties and the mode of a given step can be changed with a popup menu item.

The tree representation module offers also additional control features.

Syntax checking

Aplusix can verify the syntax of a chosen step and says “The expression is correct” or opens a window with error messages if there is a syntax error. The sub-tree where the error is located is highlighted.

Second view

It consists of a window with another representation of the chosen step and the previous step if an option is enabled and if there is a previous step.

This representation can be:

- a usual representation,
- a tree representation,
- a tree of monomials (a mixed representation in which the leaves contain monomials)

Advanced editing

All modes of interaction benefit of the same advanced editing features: Cut, copy, paste, drag & drop, selection replacement.

Exercises

There are two new standard types of exercises which can be used only in sections of problems:

- Transform a usual representation into a tree representation
- Transform a tree representation into a usual representation

4.2 Graph module

Objectives:

- 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.
- 4) To show that equivalent equations, inequations or systems lead to the same set of solutions.

Graph module general features:

- pan: the user can drag the grid and move the center as he/she wishes
- zoom in/out
- grid: the grid is marked with little grey lines representing a unit, the half of a unit, or a tenth of a unit depending on the zoom level.(Figure 20)

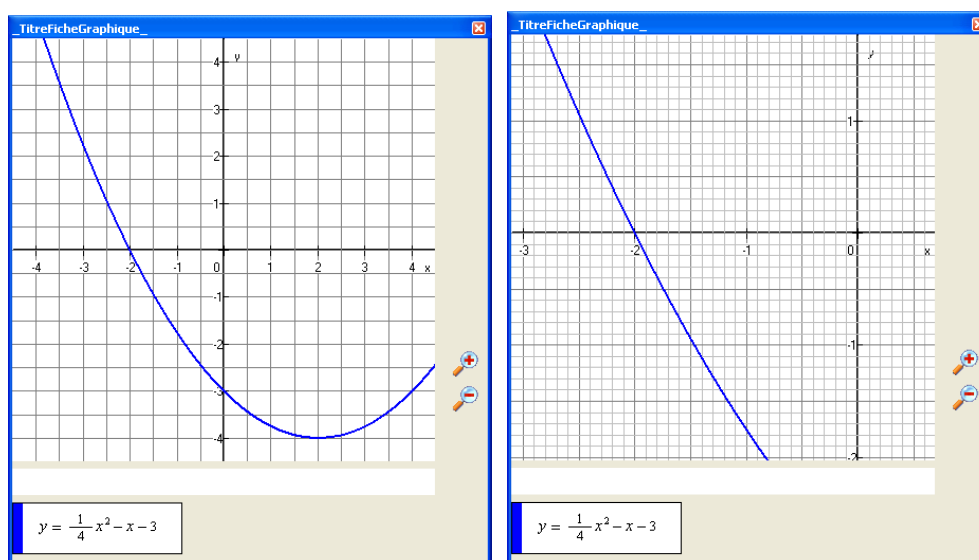


Figure 20

Curves for expressions

The graphical module of Aplusix displays polynomial and rational expressions of one variable represented as curves in the 2D-space.

This allows checking the equivalence of two steps by watching if the curves are overlapping or not.

The user should choose a step and select in the step menu to add this step to the graphical representation.

In order to enhance this verification, we have added a default behaviour for the curve style according to their drawing order. Thus, we have four basic colours and if a colour is given to a first curve the second one will have another. The same thing is applied with the curves thickness. The first curve is thinner and the next curves are getting thicker and thicker.

(Figure 21)

Curves and set of solutions for equations, inequations and systems

For the user the command to draw curves for equations is the same as simple expression.

The display is different because the user gets two curves for one step. These two curves have the same style (colour, width). In addition of the curve drawing, the set of solutions is also displayed below the grid. As it has been said in the first part of this document, it is composed of point and intervals.

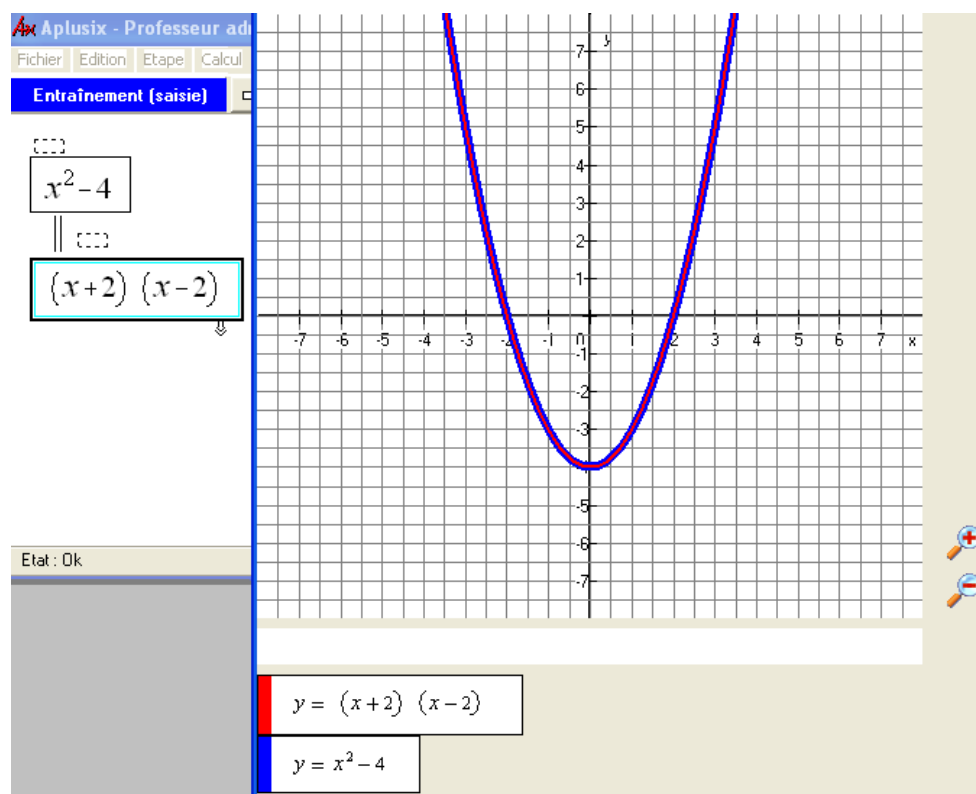


Figure 21

Curves pallet (Figure 22)

By selecting the legend of a given curve, one makes a curve tool pallet appear. When selected, the expression is highlighted by a light blue border.

One can choose to change the thickness, the colour and the order of the curve. The legends of the curves are displayed in the same order that the curves are displayed and no more on the order of the reasoning steps.



Figure 22

One can also delete the curve from representation by clicking on the red cross.

4.3 MathDiLS module

Objectives:

- 1) To allow teachers to create categories or files hosted by MathDILS.
- 2) To allow students to have an access to Aplusix files hosted by MathDILS.

Authoring tool:

From the authoring tool of Aplusix, a teacher can create his own exercise, and connect to MathDiLS, select a category and save his exercise in the category. A teacher can also edit and modify existing exercises hosted on MathDILS (Figure 23).

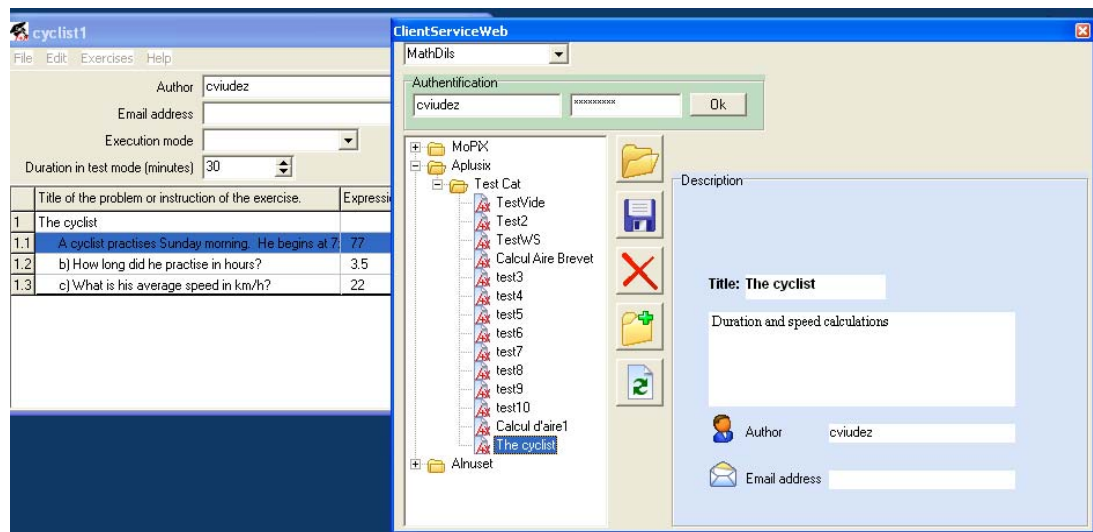


Figure 23 : Editing an exercise from MathDiLS in the Aplusix authoring tool

Aplusix:

A student can connect to MathDiLS and browse categories to find an exercise. He/she can open it in Aplusix and start working as if he/she as opened an exercise file on his computer or on the local network (Figure 24).

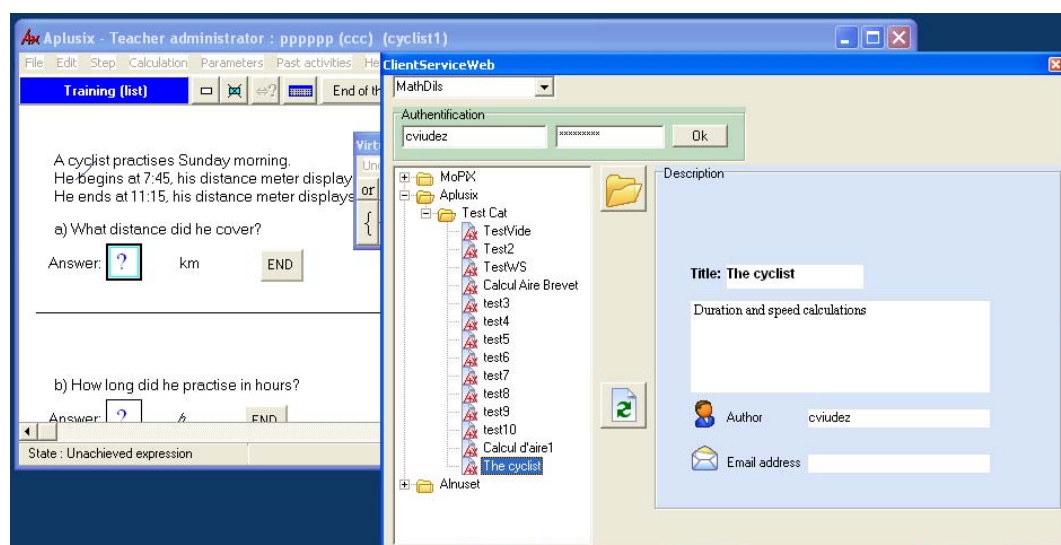


Figure 24: Launching an exercise from MathDiLS

5) Perspectives

From July 2005 to now, Aplusix has been commercialized in France, Italy, UK and Benelux through publishers who signed a contract with the UJF University. The commercialized version does not include trees and graphs.

A company, called ARISTOD, has been recently created (October, 17, 2008) in order to develop and commercialize Aplusix. This company is a start-up of the UJF University and is registered 508 577 491 RCS Grenoble. The existing contracts are currently being modified in order to have a contract between UJF and ARISTOD on the one hand, and contacts between ARISTOD and publishers on the other hand.

The first main problem of the company is to earn enough money to live, which cannot be obtained with the current “local network” version of Aplusix, because we don’t have enough customers (customers are schools). So we will develop a web version of Aplusix in order to reach individual customers (students at home who would like to improve their skills in arithmetic and algebra). This web version will also be better for schools because students of schools subscribing to Aplusix will be able to use it at school and at home. Additional features including teacher annotation of students’ works will favour teaching at distance. New exercises will be created to have a better covering of the curricula of many countries; they will be indexed by the school levels of the different countries.

The new modules developed during the Remath project will be added to the distributed version in a second phase.

a) Perspectives in terms of development

1. Moving to a web version with a specific application (AplusixWeb) using web services.
2. Developing text and math input and display in a given area to allow problem descriptions, students’ comments and teachers’ annotations in a smart environment.
3. Developing an advanced authoring tool for creating exercises and problems or simply for editing documents with texts and maths.
4. Adding the tree representations to AplusixWeb.
5. Adding the graph representations to AplusixWeb.
6. Extending the current algebraic domain (e.g., non linear systems of equations).
7. Extending the domain to trigonometry and logarithm.
8. Introducing calculus exercises (e.g., limits and derivatives).
9. Developing a companion which will be able to give hint or to perform calculation steps on demand.
10. Re-developing Aplusix as a Java applet in order to run in a browser.

b) Perspectives in terms of deployment

Firstly, a French Aplusix Web Service will be developed and launched. In a second phase, similar operations will be done in other countries with our current publishers as responsables of the services. In a third phase, we will look for new publishers for other countries.

Chevallard, Y. (1992), Concepts fondamentaux de la didactique : perspectives apportées par une approche anthropologique. *Recherches en Didactique des Mathématiques* 12/1, 77-111.

Chevallard, Y. (2006), Conférence plénière d'ouverture du 4e congrès de la */European Society for Research in Mathematics Education/* (CERME 4), Sant Feliu de Guíxols, 17-21 février 2005. In *Proceedings of the Fourth Congress of the European Society for Research in Mathematics Education*, Universitat Ramon Llull, Barcelone, 2006, 21-30.

Duval, R. (1993), Registres de représentation sémiotique et fonctionnement cognitif de la pensée, *Annales de Didactique et de Sciences Cognitives* 5, IREM de Strasbourg.

Duval R. (1995), *Sémiosis et pensée humaine*. Ginevra: Peter Lang.

Duval R. (2006), A cognitive analysis of problems of comprehension in a learning of mathematics. *Educ. Studies in Mathematics* 61(1-2), 103-131.

DDA 3: Alnuset

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

Istituto Tecnologie Didattiche - CNR

Table of content

Introduction	2
1.The latest evolutions of the DDA.....	4
2.History of the design and development of the DDA	11
3. Description of the final form of the DDA	13
4. Perspectives.....	20

Introduction

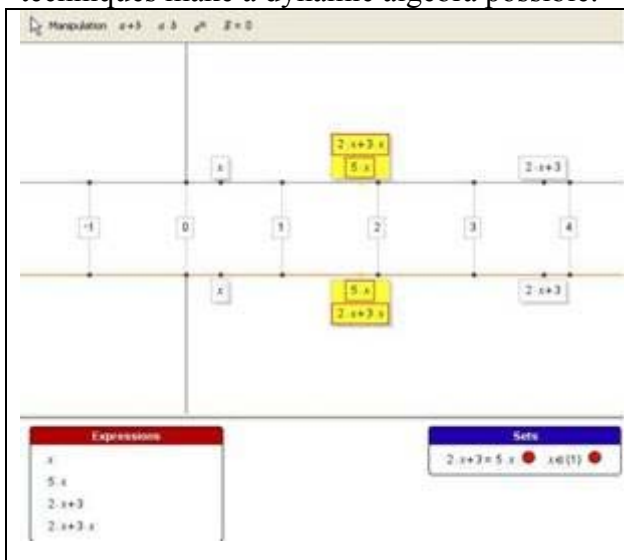
AlNuSet has been realized to favour the comprehension of fundamental mathematical concepts, to develop the mastering of algebraic language, to improve the didactical practices of algebra, of functions and of numerical sets at the level of lower and upper secondary school.

AlNuSet has been completely designed and implemented within the ReMath Project

AlNuSet is constituted of three components: the Algebraic Line component, the Algebraic Manipulator component and the Functions component. They make techniques of quantitative, symbolic and functional nature available as to operate with algebraic expressions and propositions. These techniques are deeply different from those instrumented into CAS. This diversity is motivated by the different teachers and students needs with respect to those of the CAS professional users

Algebraic Line

Its main characteristic is the representation of the algebraic variable as a mobile point on the numerical line, namely as a point which can be dragged with the mouse along the line. This feature transforms the number line into an algebraic line where it is possible to operate with algebraic expressions and propositions through techniques of quantitative and dynamic nature. These techniques make a dynamic algebra possible.



Algebraic Line. Representation of variable and expressions on the line. The drag of x allows to verify the equivalence between expressions and to explore the solution for the equation

Algebraic Manipulator

Its main characteristic is the possibility to transform algebraic expressions and propositions through a set of particular commands. These commands correspond to the basic properties of operations, to the properties of equality and inequality, to the logic operations among propositions, to the operations among sets. In this Manipulator it is possible to create a new transformation rule (user rule) once this rule has been proved. These characteristics support the development of skills in the algebraic transformation and they contribute to assign the meaning of proof to it.

1. The latest evolutions of the DDA

The latest evolution of AlNuSet here described highlights modifications and new implementations with respect to deliverables 4 and 8. They concern:

- a) The general menu bar of AlNuSet (partially modified).
- b) The algebraic bi-dimensional editor (new)
- c) The algebraic Line component (partially modified)
- d) The symbolic manipulator component (partially modified)
- e) The functions component (new)
- f) The communication between AlNuSet and MathDiLS platform (new).

a) The general menu bar of AlNuSet

Some modifications have interested the menu bar of AlNuSet with respect to the previous versions of the DDA.

At present, the main menu of AlNuSet (see Fig. 3) is constituted by:

- The “File” menu containing the standard functions to manage files (new, open, save, save as, quit) and some functions to export/import files from/to the MathDiLS platform (MathDiLS open, MathDiLS save, MathDiLS save as). The latter functions will be described in point f) of this section
- The “Domain” menu to choose the numerical set to operate on.
- The “Help” menu.

Moreover, the new command “Edit...” allows the user to access the bi-dimensional algebraic editor (see next point) while the “Quick editor” field of the interface makes available a standard linear algebraic editor.

b) The main features of the bi-dimensional editor

The bi-dimensional algebraic editor, not included in the first version of AlNuSet, has been implemented to simplify the editing of expressions, propositions and numerical sets to operate with. Apart from bi-dimensionality, the main characteristic of this algebraic editor is that its use is based both on icons of its interface and on keyboard commands.

The icons of the interface refer to three kinds of commands (see Fig. 1):

- commands to edit expressions (the first line in the figure),
- commands to edit algebraic propositions and systems of propositions (the second line in the figure)
- commands to edit numerical Sets (the third line in the figure).

Interaction with the editor is quite easy.

The user can insert an expression (or a proposition or a Set) exploiting the icons of the interface or directly through the keyboard. Numbers and mathematical symbols are automatically inserted in a specific blue square (in the Fig. 1).

Two cursors, respectively red and yellow, are involved in the editing activity.

The red cursor indicates the position where the typed character (number, letter or mathematical symbol) will be inserted. The red cursor is always present in the writing at hand. Instead, the yellow cursor appears in the structure of the algebraic writing at hand when the mouse points at a position that is different from the position of the insertion. The yellow cursor is useful in the selection of parts of an expression - it indicates the starting point for the selection in relation to the actual position of the mouse (Fig. 2).

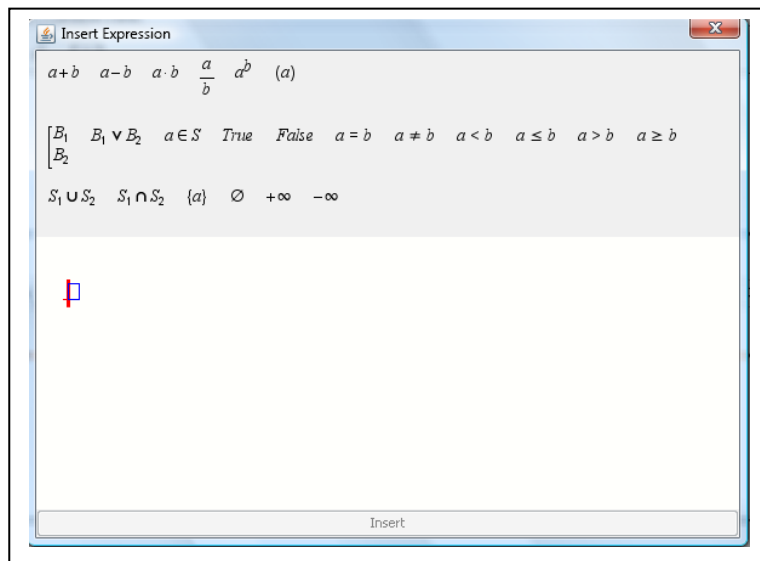


Fig. 1: The interface of the bi-dimensional editor

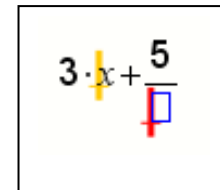


Fig. 2: Example of representation during the insertion of an expression in the bi-dimensional editor (the red cursor and the blue square show where the next number will be inserted; the yellow cursor indicates the starting point for the selection)

Once the user has finished editing, the insertion of the edited writing in the actual component of AlNuSet occurs through the “Insert” button of the interface.

c) Modifications of the Algebraic Line Component

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 expressions that are displayed on two parallel marked axes.

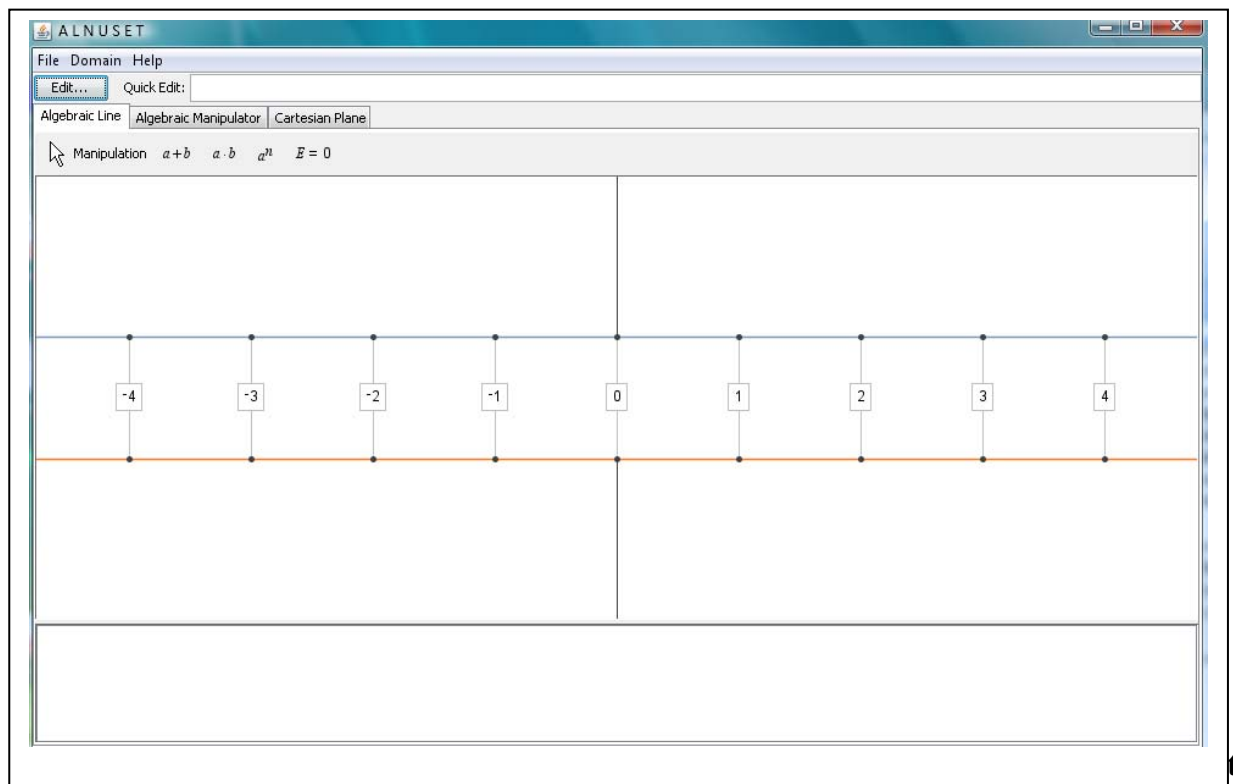
By dragging letters on the line, all expressions depending on them are refreshed accordingly.

An algebraic expression can be either edited by a linear or by a bi-dimensional editor or it can be constructed through a graphical editor, based on three geometrical models of the algebraic operations (addition/subtraction; multiplication/division, integer power/rational power).

Algebraic propositions can be edited by means of the linear or the bi-dimensional editors. With the implementation of the bi-dimensional editor, the command “Comparison” to construct simple propositions, described in a previous deliverable, was eliminated. Once the proposition is edited it is automatically reported in the specific window named “Sets”.

In this version of AlNuSet the instrumented technique to construct and validate the truth set of algebraic propositions is unvaried while the instrumented technique to find roots of polynomials has been improved. At the moment there is no limit to the degree of the polynomial on which to operate while formerly the degree ranged from greater than 1 to less than or equal to 4.

The latter version of the interface of the Algebraic Line Component of AlNuSet is the following (Fig. 3).



The interface is constituted by the main menu bar of AlNuSet and by the following specific commands:

- The manipulation command to drag points on the line.
- The three commands of the graphical editors to access the corresponding models of the operations (addition/subtraction; multiplication/division, integer power/rational power).
- The command to find the root(s) of a polynomial

In this version of AlNuSet the specific functions concerning the setting, the state and the modes of representation of expressions on the Algebraic Line have been improved from the point of view of their interface language.

Many functions can be applied using the right button of the mouse when the cursor points at the expression or the proposition to work on.

Below (Fig. 4) the possibilities made available for expressions:

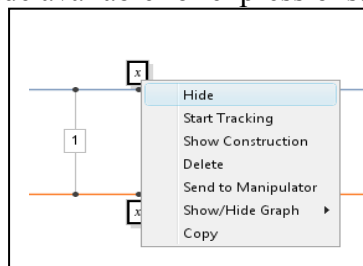


Fig. 4: The popup menu available through the right button of the mouse when the cursor points on the expression

Given an expression on the line or in the specific space “Expressions”, clicking on the right button of the mouse it is possible: to hide the expression, to start the tracking with it, to show its construction, to delete it, to send it to the manipulator, to construct its graph in the Cartesian plan.

The figure below (Fig. 5) shows the possibilities made available for propositions, once they are edited in the Algebraic line component. They are automatically inserted in the specific window named “Sets” where specific commands can be applied on it.

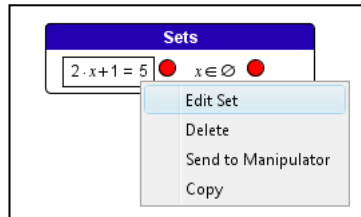


Fig. 5: The popup menu available through the right button of the mouse when the cursor points on the proposition

Given a proposition the user can edit its truth set, delete it, send it to the manipulator.

d) Main new features of the Algebraic Manipulator Component

The Algebraic Manipulator component is a structured symbolic calculation environment for the manipulation of algebraic expressions and for the solution of equations and inequations.

With respect to previous versions, some important modifications were made.

1. The solution of equations and inequations is not limited to a degree less than or equal to 4. Moreover, the solution can be written both in logic format and in set format.
2. There is a specific space in which commands corresponding to new transformational rules are inserted, namely “user rules”, created by the user once that they have been tested using the basic commands available in the interface or new, previously tested rules. These rules can be now saved, imported and used in further algebraic manipulations.
3. The basic rules of the interface were modified as shown in Fig. 6 and Fig. 7.

The interface of the Algebraic Manipulator component provides a set of commands for symbolic manipulation that is specific to the numeric domain chosen by the user (Natural integers, Relative integers, Rational numbers, Full range domain). These commands can be applied on a part of expression or proposition to transform it. In this way, the user can control each step of the algebraic transformation solution process.

In the following figures, the rules that can be applied for expressions and propositions are reported. Fig. 6 represents a table containing the rules for Natural numbers and the modifications (with respect to the rules of this table) to obtain rules for Integer Numbers. Fig. 7 represents a table containing the rules for rational numbers and the rules modified in this table to obtain the set of rules for full range domain.

Non-negative integers rules		Integers rules																																												
<table><tr><th>Addition</th></tr><tr><td>$A+B \Leftrightarrow B+A$</td></tr><tr><td>$A+(B+C) \Leftrightarrow (A+B)+C$</td></tr><tr><td>$A \Leftrightarrow A+0$</td></tr><tr><td>$A-A \Leftrightarrow 0$</td></tr><tr><td>$a_1+a_2+\dots \Rightarrow n$</td></tr><tr><td>$n \Rightarrow a+b$</td></tr><tr><th>Powers</th></tr><tr><td>$A^n \Leftrightarrow A \cdot A \dots$</td></tr><tr><td>$A^{n_1+n_2+\dots} \Leftrightarrow A^{n_1} \cdot A^{n_2} \dots$</td></tr><tr><td>$(A_1 \cdot A_2 \dots)^n \Leftrightarrow A_1^n \cdot A_2^n \dots$</td></tr><tr><td>$(A^n)^m \Leftrightarrow A^{n \cdot m}$</td></tr><tr><th>Computation</th></tr><tr><td>$A \Rightarrow (A)$</td></tr><tr><td>Remove extra ()</td></tr><tr><td>Simplify numerical expression</td></tr><tr><th>Logic and Set</th></tr><tr><td>Simplify boolean expression</td></tr><tr><td>Simplify set</td></tr><tr><td>$L \Leftrightarrow x \in S$</td></tr><tr><td>$x \in S_1 \vee x \in S_2 \vee \dots \Rightarrow x \in S_1 \cup S_2 \cup \dots$</td></tr><tr><td>$\begin{cases} x \in S_1 \\ x \in S_2 \\ \dots \end{cases} \Rightarrow x \in S_1 \cap S_2 \cap \dots$</td></tr></table>	Addition	$A+B \Leftrightarrow B+A$	$A+(B+C) \Leftrightarrow (A+B)+C$	$A \Leftrightarrow A+0$	$A-A \Leftrightarrow 0$	$a_1+a_2+\dots \Rightarrow n$	$n \Rightarrow a+b$	Powers	$A^n \Leftrightarrow A \cdot A \dots$	$A^{n_1+n_2+\dots} \Leftrightarrow A^{n_1} \cdot A^{n_2} \dots$	$(A_1 \cdot A_2 \dots)^n \Leftrightarrow A_1^n \cdot A_2^n \dots$	$(A^n)^m \Leftrightarrow A^{n \cdot m}$	Computation	$A \Rightarrow (A)$	Remove extra ()	Simplify numerical expression	Logic and Set	Simplify boolean expression	Simplify set	$L \Leftrightarrow x \in S$	$x \in S_1 \vee x \in S_2 \vee \dots \Rightarrow x \in S_1 \cup S_2 \cup \dots$	$\begin{cases} x \in S_1 \\ x \in S_2 \\ \dots \end{cases} \Rightarrow x \in S_1 \cap S_2 \cap \dots$	<table><tr><th>Multiplication</th></tr><tr><td>$A \cdot B \Leftrightarrow B \cdot A$</td></tr><tr><td>$A \cdot (B \cdot C) \Leftrightarrow (A \cdot B) \cdot C$</td></tr><tr><td>$A \Leftrightarrow A \cdot 1$</td></tr><tr><td>$A \cdot 0 \Leftrightarrow 0$</td></tr><tr><td>$(A_1 \cdot A_2 \dots \cdot B) \div B \Rightarrow A_1 \cdot A_2 \dots$</td></tr><tr><td>$a_1 \cdot a_2 \dots \Rightarrow n$</td></tr><tr><td>$n \Rightarrow p_1 \cdot p_2 \dots$</td></tr><tr><th>Distribute / Factor</th></tr><tr><td>$A \cdot (B_1+B_2+\dots) \Leftrightarrow A \cdot B_1+A \cdot B_2+\dots$</td></tr><tr><th>Solving</th></tr><tr><td>$A \leq B \Rightarrow B \leq A$</td></tr><tr><td>$A \leq B \Rightarrow A-B \leq 0$</td></tr><tr><td>$A \leq B+T \Rightarrow A-T \leq B$</td></tr><tr><td>$A+T \leq B \Rightarrow A \leq B-T$</td></tr><tr><td>$T \cdot A \leq B \Rightarrow A \leq \frac{B}{T}$</td></tr><tr><td>$A^2 \leq B \Rightarrow A \leq \sqrt{B}$</td></tr><tr><td>$T \cdot A \leq 0 \Rightarrow T \leq 0 \vee A \leq 0$</td></tr><tr><th>Insert from Algebraic Line</th></tr><tr><td>Factor rational roots</td></tr><tr><td>Insert solution set</td></tr><tr><td>Instantiate variable</td></tr></table>	Multiplication	$A \cdot B \Leftrightarrow B \cdot A$	$A \cdot (B \cdot C) \Leftrightarrow (A \cdot B) \cdot C$	$A \Leftrightarrow A \cdot 1$	$A \cdot 0 \Leftrightarrow 0$	$(A_1 \cdot A_2 \dots \cdot B) \div B \Rightarrow A_1 \cdot A_2 \dots$	$a_1 \cdot a_2 \dots \Rightarrow n$	$n \Rightarrow p_1 \cdot p_2 \dots$	Distribute / Factor	$A \cdot (B_1+B_2+\dots) \Leftrightarrow A \cdot B_1+A \cdot B_2+\dots$	Solving	$A \leq B \Rightarrow B \leq A$	$A \leq B \Rightarrow A-B \leq 0$	$A \leq B+T \Rightarrow A-T \leq B$	$A+T \leq B \Rightarrow A \leq B-T$	$T \cdot A \leq B \Rightarrow A \leq \frac{B}{T}$	$A^2 \leq B \Rightarrow A \leq \sqrt{B}$	$T \cdot A \leq 0 \Rightarrow T \leq 0 \vee A \leq 0$	Insert from Algebraic Line	Factor rational roots	Insert solution set	Instantiate variable	<p>With respect to the table containing non-negative integers rules (left) two rules are modified:</p> <ul style="list-style-type: none">The addition rule $A-A \Leftrightarrow 0$ is replaced by the two rules $A+-A \Leftrightarrow 0$ $A-B \Leftrightarrow A+-B$The multiplication rule: $(A_1 \cdot A_2 \dots \cdot B) \div B \Rightarrow A_1 \cdot A_2 \dots$ is replaced by the two rules $-A \Leftrightarrow -1 \cdot A$ $1 \Leftrightarrow -1 \cdot -1$
Addition																																														
$A+B \Leftrightarrow B+A$																																														
$A+(B+C) \Leftrightarrow (A+B)+C$																																														
$A \Leftrightarrow A+0$																																														
$A-A \Leftrightarrow 0$																																														
$a_1+a_2+\dots \Rightarrow n$																																														
$n \Rightarrow a+b$																																														
Powers																																														
$A^n \Leftrightarrow A \cdot A \dots$																																														
$A^{n_1+n_2+\dots} \Leftrightarrow A^{n_1} \cdot A^{n_2} \dots$																																														
$(A_1 \cdot A_2 \dots)^n \Leftrightarrow A_1^n \cdot A_2^n \dots$																																														
$(A^n)^m \Leftrightarrow A^{n \cdot m}$																																														
Computation																																														
$A \Rightarrow (A)$																																														
Remove extra ()																																														
Simplify numerical expression																																														
Logic and Set																																														
Simplify boolean expression																																														
Simplify set																																														
$L \Leftrightarrow x \in S$																																														
$x \in S_1 \vee x \in S_2 \vee \dots \Rightarrow x \in S_1 \cup S_2 \cup \dots$																																														
$\begin{cases} x \in S_1 \\ x \in S_2 \\ \dots \end{cases} \Rightarrow x \in S_1 \cap S_2 \cap \dots$																																														
Multiplication																																														
$A \cdot B \Leftrightarrow B \cdot A$																																														
$A \cdot (B \cdot C) \Leftrightarrow (A \cdot B) \cdot C$																																														
$A \Leftrightarrow A \cdot 1$																																														
$A \cdot 0 \Leftrightarrow 0$																																														
$(A_1 \cdot A_2 \dots \cdot B) \div B \Rightarrow A_1 \cdot A_2 \dots$																																														
$a_1 \cdot a_2 \dots \Rightarrow n$																																														
$n \Rightarrow p_1 \cdot p_2 \dots$																																														
Distribute / Factor																																														
$A \cdot (B_1+B_2+\dots) \Leftrightarrow A \cdot B_1+A \cdot B_2+\dots$																																														
Solving																																														
$A \leq B \Rightarrow B \leq A$																																														
$A \leq B \Rightarrow A-B \leq 0$																																														
$A \leq B+T \Rightarrow A-T \leq B$																																														
$A+T \leq B \Rightarrow A \leq B-T$																																														
$T \cdot A \leq B \Rightarrow A \leq \frac{B}{T}$																																														
$A^2 \leq B \Rightarrow A \leq \sqrt{B}$																																														
$T \cdot A \leq 0 \Rightarrow T \leq 0 \vee A \leq 0$																																														
Insert from Algebraic Line																																														
Factor rational roots																																														
Insert solution set																																														
Instantiate variable																																														

Fig. 6: The interface of the Algebraic Manipulator component when natural numbers domain or Integers domain is selected

Rational rules		Full range rules																																																						
<table><tr><th>Addition</th></tr><tr><td>$A+B \Rightarrow B+A$</td></tr><tr><td>$A+(B+C) \Rightarrow (A+B)+C$</td></tr><tr><td>$A \Rightarrow A+0$</td></tr><tr><td>$A+-A \Rightarrow 0$</td></tr><tr><td>$A-B \Rightarrow A+-B$</td></tr><tr><td>$a_1+a_2+\dots \Rightarrow x$</td></tr><tr><td>$n \Rightarrow a+b$</td></tr><tr><th>Powers</th></tr><tr><td>$A^n \Rightarrow A \cdot A \dots$</td></tr><tr><td>$A^{n_1+n_2+\dots} \Rightarrow A^{n_1} \cdot A^{n_2} \dots$</td></tr><tr><td>$(A_1 \cdot A_2 \dots)^n \Rightarrow A_1^n \cdot A_2^n \dots$</td></tr><tr><td>$(A^n)^m \Rightarrow A^{n \cdot m}$</td></tr><tr><td>$A^{-n} \Rightarrow \frac{1}{A^n}$</td></tr><tr><th>Computation</th></tr><tr><td>$A \Rightarrow (A)$</td></tr><tr><td>Remove extra ()</td></tr><tr><td>Simplify numerical expression</td></tr><tr><td>Expand</td></tr><tr><td>Collect</td></tr><tr><td>Eliminate variable</td></tr><tr><th>Logic and Set</th></tr><tr><td>Simplify boolean expression</td></tr><tr><td>Simplify set</td></tr><tr><td>$L \Rightarrow x \in S$</td></tr><tr><td>$x \in S_1 \vee x \in S_2 \vee \dots \Rightarrow x \in S_1 \cup S_2 \cup \dots$</td></tr><tr><td>$\begin{cases} x \in S_1 \\ x \in S_2 \end{cases} \Rightarrow x \in S_1 \cap S_2 \cap \dots$</td></tr></table>	Addition	$A+B \Rightarrow B+A$	$A+(B+C) \Rightarrow (A+B)+C$	$A \Rightarrow A+0$	$A+-A \Rightarrow 0$	$A-B \Rightarrow A+-B$	$a_1+a_2+\dots \Rightarrow x$	$n \Rightarrow a+b$	Powers	$A^n \Rightarrow A \cdot A \dots$	$A^{n_1+n_2+\dots} \Rightarrow A^{n_1} \cdot A^{n_2} \dots$	$(A_1 \cdot A_2 \dots)^n \Rightarrow A_1^n \cdot A_2^n \dots$	$(A^n)^m \Rightarrow A^{n \cdot m}$	$A^{-n} \Rightarrow \frac{1}{A^n}$	Computation	$A \Rightarrow (A)$	Remove extra ()	Simplify numerical expression	Expand	Collect	Eliminate variable	Logic and Set	Simplify boolean expression	Simplify set	$L \Rightarrow x \in S$	$x \in S_1 \vee x \in S_2 \vee \dots \Rightarrow x \in S_1 \cup S_2 \cup \dots$	$\begin{cases} x \in S_1 \\ x \in S_2 \end{cases} \Rightarrow x \in S_1 \cap S_2 \cap \dots$	<table><tr><th>Multiplication</th></tr><tr><td>$A \cdot B \Rightarrow B \cdot A$</td></tr><tr><td>$A \cdot (B \cdot C) \Rightarrow (A \cdot B) \cdot C$</td></tr><tr><td>$A \Rightarrow A \cdot 1$</td></tr><tr><td>$A \cdot 0 \Rightarrow 0$</td></tr><tr><td>$-A \Rightarrow -1 \cdot A$</td></tr><tr><td>$1 \Rightarrow -1 \cdot -1$</td></tr><tr><td>$A \cdot \frac{1}{A} \Rightarrow 1$</td></tr><tr><td>$\frac{A}{B} \Rightarrow A \cdot \frac{1}{B}$</td></tr><tr><td>$\frac{1}{A_1 \cdot A_2 \dots} \Rightarrow \frac{1}{A_1} \cdot \frac{1}{A_2} \dots$</td></tr><tr><td>$a_1 \cdot a_2 \dots \Rightarrow x$</td></tr><tr><td>$n \Rightarrow p_1 \cdot p_2 \dots$</td></tr><tr><th>Distribute / Factor</th></tr><tr><td>$A \cdot (B_1+B_2+\dots) \Rightarrow A \cdot B_1+A \cdot B_2+\dots$</td></tr><tr><th>Solving</th></tr><tr><td>$A \leq B \Rightarrow B \leq A$</td></tr><tr><td>$A \leq B \Rightarrow A-B \leq 0$</td></tr><tr><td>$A \leq B+T \Rightarrow A-T \leq B$</td></tr><tr><td>$A+T \leq B \Rightarrow A \leq B-T$</td></tr><tr><td>$T \cdot A \leq B \Rightarrow A \leq \frac{B}{T}$</td></tr><tr><td>$A^2 \leq B \Rightarrow A \leq \sqrt{B}$</td></tr><tr><td>$T \cdot A \leq 0 \Rightarrow T \leq 0 \vee A \leq 0$</td></tr><tr><td>$\frac{A}{B} \leq 0 \Rightarrow \begin{cases} A \leq 0 \\ B \leq 0 \end{cases}$</td></tr><tr><th>Insert from Algebraic Line</th></tr><tr><td>Factor rational roots</td></tr><tr><td>Insert solution set</td></tr><tr><td>Instantiate variable</td></tr></table>	Multiplication	$A \cdot B \Rightarrow B \cdot A$	$A \cdot (B \cdot C) \Rightarrow (A \cdot B) \cdot C$	$A \Rightarrow A \cdot 1$	$A \cdot 0 \Rightarrow 0$	$-A \Rightarrow -1 \cdot A$	$1 \Rightarrow -1 \cdot -1$	$A \cdot \frac{1}{A} \Rightarrow 1$	$\frac{A}{B} \Rightarrow A \cdot \frac{1}{B}$	$\frac{1}{A_1 \cdot A_2 \dots} \Rightarrow \frac{1}{A_1} \cdot \frac{1}{A_2} \dots$	$a_1 \cdot a_2 \dots \Rightarrow x$	$n \Rightarrow p_1 \cdot p_2 \dots$	Distribute / Factor	$A \cdot (B_1+B_2+\dots) \Rightarrow A \cdot B_1+A \cdot B_2+\dots$	Solving	$A \leq B \Rightarrow B \leq A$	$A \leq B \Rightarrow A-B \leq 0$	$A \leq B+T \Rightarrow A-T \leq B$	$A+T \leq B \Rightarrow A \leq B-T$	$T \cdot A \leq B \Rightarrow A \leq \frac{B}{T}$	$A^2 \leq B \Rightarrow A \leq \sqrt{B}$	$T \cdot A \leq 0 \Rightarrow T \leq 0 \vee A \leq 0$	$\frac{A}{B} \leq 0 \Rightarrow \begin{cases} A \leq 0 \\ B \leq 0 \end{cases}$	Insert from Algebraic Line	Factor rational roots	Insert solution set	Instantiate variable	<p>With respect to the table containing the Rational number rules (left), two new rules are added:</p> <ul style="list-style-type: none">A new rule has been added to the Powers rules: $A^{\frac{1}{2}} \Leftrightarrow \sqrt{A}$A new rule has added to the Solving rules: $A^{\frac{p}{q}} \leq B \Rightarrow A^p \leq B^q$
Addition																																																								
$A+B \Rightarrow B+A$																																																								
$A+(B+C) \Rightarrow (A+B)+C$																																																								
$A \Rightarrow A+0$																																																								
$A+-A \Rightarrow 0$																																																								
$A-B \Rightarrow A+-B$																																																								
$a_1+a_2+\dots \Rightarrow x$																																																								
$n \Rightarrow a+b$																																																								
Powers																																																								
$A^n \Rightarrow A \cdot A \dots$																																																								
$A^{n_1+n_2+\dots} \Rightarrow A^{n_1} \cdot A^{n_2} \dots$																																																								
$(A_1 \cdot A_2 \dots)^n \Rightarrow A_1^n \cdot A_2^n \dots$																																																								
$(A^n)^m \Rightarrow A^{n \cdot m}$																																																								
$A^{-n} \Rightarrow \frac{1}{A^n}$																																																								
Computation																																																								
$A \Rightarrow (A)$																																																								
Remove extra ()																																																								
Simplify numerical expression																																																								
Expand																																																								
Collect																																																								
Eliminate variable																																																								
Logic and Set																																																								
Simplify boolean expression																																																								
Simplify set																																																								
$L \Rightarrow x \in S$																																																								
$x \in S_1 \vee x \in S_2 \vee \dots \Rightarrow x \in S_1 \cup S_2 \cup \dots$																																																								
$\begin{cases} x \in S_1 \\ x \in S_2 \end{cases} \Rightarrow x \in S_1 \cap S_2 \cap \dots$																																																								
Multiplication																																																								
$A \cdot B \Rightarrow B \cdot A$																																																								
$A \cdot (B \cdot C) \Rightarrow (A \cdot B) \cdot C$																																																								
$A \Rightarrow A \cdot 1$																																																								
$A \cdot 0 \Rightarrow 0$																																																								
$-A \Rightarrow -1 \cdot A$																																																								
$1 \Rightarrow -1 \cdot -1$																																																								
$A \cdot \frac{1}{A} \Rightarrow 1$																																																								
$\frac{A}{B} \Rightarrow A \cdot \frac{1}{B}$																																																								
$\frac{1}{A_1 \cdot A_2 \dots} \Rightarrow \frac{1}{A_1} \cdot \frac{1}{A_2} \dots$																																																								
$a_1 \cdot a_2 \dots \Rightarrow x$																																																								
$n \Rightarrow p_1 \cdot p_2 \dots$																																																								
Distribute / Factor																																																								
$A \cdot (B_1+B_2+\dots) \Rightarrow A \cdot B_1+A \cdot B_2+\dots$																																																								
Solving																																																								
$A \leq B \Rightarrow B \leq A$																																																								
$A \leq B \Rightarrow A-B \leq 0$																																																								
$A \leq B+T \Rightarrow A-T \leq B$																																																								
$A+T \leq B \Rightarrow A \leq B-T$																																																								
$T \cdot A \leq B \Rightarrow A \leq \frac{B}{T}$																																																								
$A^2 \leq B \Rightarrow A \leq \sqrt{B}$																																																								
$T \cdot A \leq 0 \Rightarrow T \leq 0 \vee A \leq 0$																																																								
$\frac{A}{B} \leq 0 \Rightarrow \begin{cases} A \leq 0 \\ B \leq 0 \end{cases}$																																																								
Insert from Algebraic Line																																																								
Factor rational roots																																																								
Insert solution set																																																								
Instantiate variable																																																								

Fig. 7: The interface of the Algebraic Manipulator component when rational numbers domain or full range domain is selected

It should be noted that commands are divided into the following fields:

- Addition rules
- Multiplication rules
- Powers rules
- Distribute/factor rules
- Computation rules. In this field, two specific rules, also present in the CAS systems were inserted. They are “Expand” and “Collect”. They determine the result of a numerical expression and the result of a computation with polynomials respectively. These commands simplify the manipulation. In particular, they can be used once students master the transformation of an expression into another one. Moreover, two specific rules in this field allow the user to insert or eliminate parentheses. Finally, the rule “Eliminate variable” allows the user to apply the substitution rule in a system.
- Logic and Set rules. This newly inserted field provides logic and sets commands to manipulate sets. These rules are particularly useful to write the solution set of a proposition in a set format.
- Solving rules.
- Insert from algebraic Line. These commands correspond to three rules that allow the user to import results from the Algebraic line component to the manipulator, namely the root of a polynomial to factor it, the truth set of a proposition and the value assumed by a variable on the algebraic line in order to replace it in the expression to be manipulated.

e) Main features of the Function Component

AlNuSet was enriched with a Function component. Its interface is composed by the Algebraic line and by a Cartesian plane. Its main characteristic is the possibility to represent an expression visualized on the algebraic line through the graph of its associated function in the Cartesian Plan. This occurs through the command “Show graph” applied to the expression on the line and the selection of a letter contained in the expression as the independent variable of the function.

It is important to note that the implementation of this component was not planned at the beginning of the project. It was designed since the relations emerging during the drag of a variable point on the line between the representation of an expression on the Algebraic line and its graph on the Cartesian plan, could be very interesting and effective from a didactical point of view.

By dragging a point corresponding to an algebraic variable, both the expression involving that variable on the algebraic line and the point on the graph of the Cartesian plan move accordingly.

With regard to features described in deliverable 8, several improvements of graphic rendering were realized.

f) Main features of the relationship between AlNuSet and MathDiLS platform

AlNuSet has been equipped with three commands (MathDiLS open, MathDiLS save, MathDiLS save as..) that allow the user to interact with the MathDiLS server and to use it as repository to save and store files of the activity performed with AlNuSet. Moreover a new application, named AlNuSet MathDiLS Manager, has been implemented to construct and delete categories (folders) on the MathDiLS server and to delete files stored within these categories.

When one of these three commands is used to interact with the MathDiLS server, a specific window for the interaction between AlNuSet and the MathDiLS platform opens (See Figs. 8 and 9). In particular, the command operates as follows:

- “MathDiLS Open” allows the user to access the list of AlNuSet files stored in a specific category (folder) on the MathDiLS server, to select one of them and to open it.

- “MathDiLS Save...” or “MathDiLS Save as...” allows the user to save the actual state of the application as file within a category (folder) of the MathDiLS server.

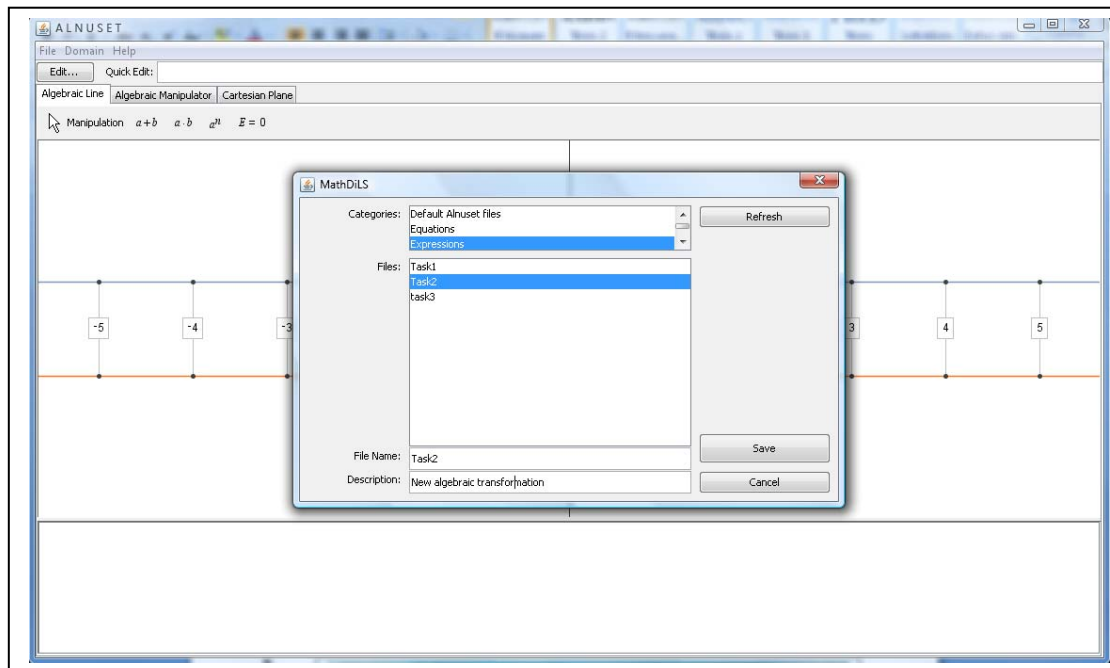


Fig. 8: The interface of the “MathDiLS Open” window

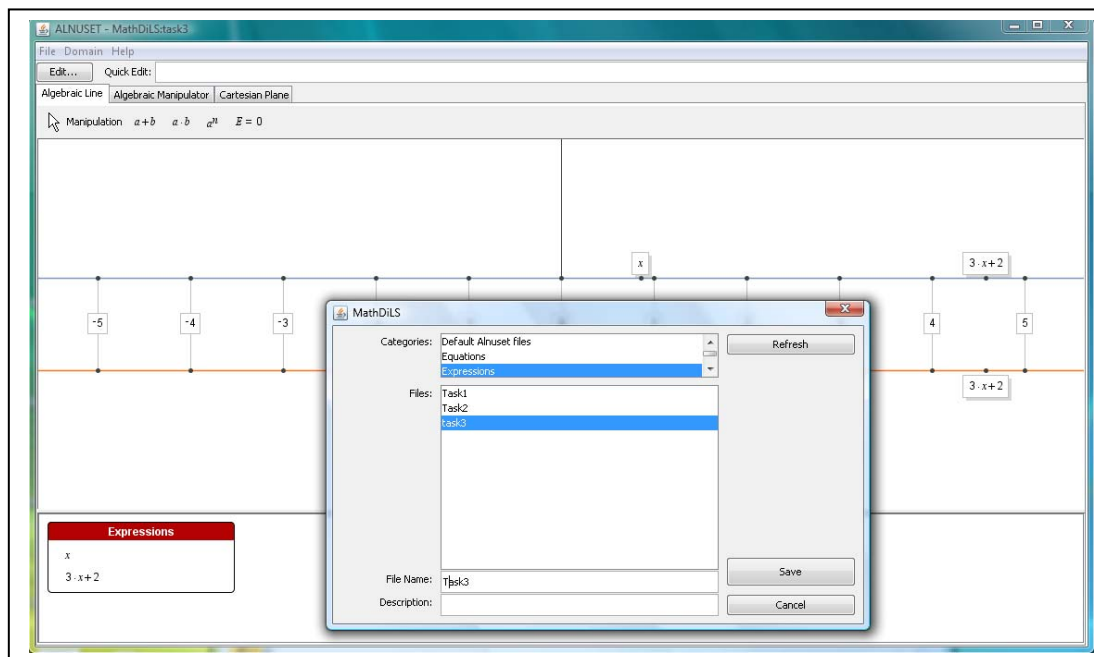


Fig. 9: The interface of the “MathDiLS save as...” window

A specific application, namely “AlNuSet MathDiLS Manager” (See Fig. 10) can be used to manage categories (or folders) in AlNuSet. Through this application it is possible to create a category, delete a category and delete a file within a selected category.

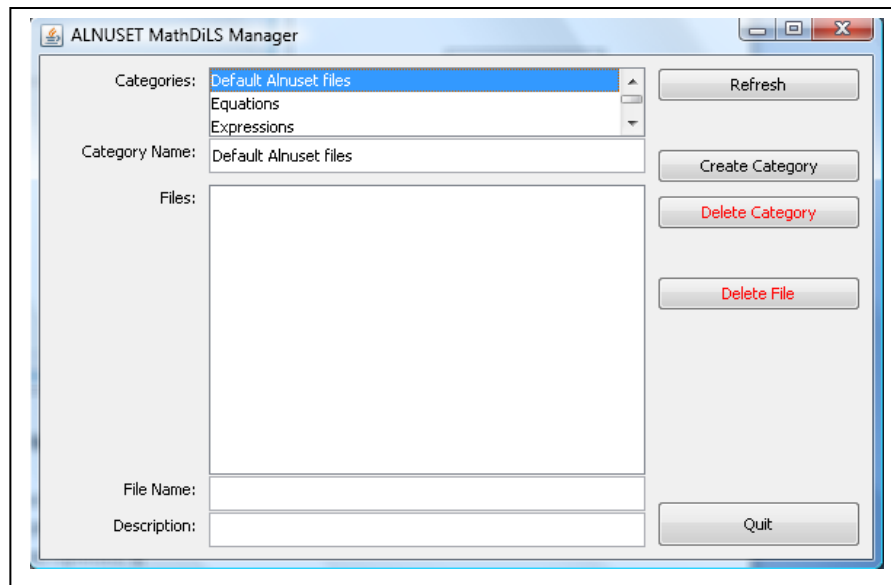


Fig. 10: The interface of the AlNuSet MathDiLS Manager

2. History of the design and development of the DDA

At the beginning of the Remath project, the idea of DDA development by ITD-CNR was rooted on the two microworlds embedded into Ari-Lab 2 system that the team had previously realized, namely the Fraction microworld and the Symbolic manipulator microworld. We noted that the Fraction microworld made available a specific operative and representative environment to explore properties of positive rational numbers. Moreover, the symbolic manipulator microworld makes available an environment to perform the transformation of numerical expressions that is centered on the use of a set of re-writing commands making reference to properties and rules of addition and multiplication.

In realizing DDA, the initial idea was to enrich the functionalities of the two described microworlds with new operative and representative possibilities. For example, the team thought of transforming the operative and representative device of the Euclidean line of fraction microworld in order to include negative numbers. Moreover, the team thought of inserting a function to create user rules into the symbolic manipulator and new commands to solve equations.

The design of the DDA marked a turning point when in the mind of a team member emerged the idea to represent the algebraic variable on the number line. This idea is at the basis of the successive developments of AlNuSet within the Remath project. According to this idea, the algebraic variable is a mobile point on the line associated to a letter that can be dragged along the line with the mouse. The importance of this idea is strictly connected to the possibility to represent algebraic expressions as points on the line whose positions depend on the value of the variable that can change through the drag.

This idea has deeply changed the initial setting of the project. The knowledge domain of reference for the DDA has been extended immediately: not only the domain of properties of numeric sets, but also the algebra domain and successively, with the design of a newly devised environment, the function domain too. Even the terminology used by the team changes - the numeric line (or Euclidean line) becomes algebraic line and a new conceptualization of the algebraic activity for

teaching aims is being devised. From the standpoint of technological and educational innovation what is the added value of this idea?

Generally speaking, we observe that the advent of the dynamic geometrical artefacts or of spreadsheets has evidenced that even a single creative idea may give birth to a new kind of innovative artefact, i.e., when it allows to instrument techniques with a new operative and representative dimension. This is the case of the drag of the variable point of a geometrical construction, as in dynamic geometrical software, or in the automatic re-computation of formulas of the table, as in the case of spreadsheet.

The idea of representing an algebraic variable as mobile point on the line allowed to implement new quantitative techniques to operate with algebraic expressions and propositions, to instrument already existing techniques in a new way, to integrate operatively techniques of different nature. This occurred with the solution of different design problems regarding the distribution of tasks and responsibilities between user and computer in executing a specific technique. Other problems concerned the management of interactivity during the execution of the technique, namely, the operative modalities of input by the user, the representation of the result by the computer (output), the visualisation of specific feedback to support the user action or to accompany the presentation of the result. Moreover, design problems have regarded also instrumented techniques interconnections. Generally speaking, the way these problems are solved affects techniques accessibility by user, their usefulness for the tasks to be solved, the meaning instrumented techniques evidence in the interaction, the discourse that can be developed about them. The following paragraph describes the main steps in the development of AlNuSet in the last three years from an operative standpoint while next section describes AlNuSet from the point of view of the technological and educational innovation

The phases of the AlNuSet development

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 focus was 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 ($\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}$), the variable and its type (integer, rational, real) and the variable drag were implemented. Finally, polynomial root tool, auto-scrolling and zoom function, a displayed full post-it for operands were set up.

As to the Algebraic manipulator component, the activity 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 were implemented. Moreover, the selection of a part of expression, the activation of available rules, and the execution of rules were devised.

At the end of the first year a prototype of ALNUSET constituted by two separate Java applications, the Algebraic Line component and the Algebraic Manipulator component was available.

During the second and third year of ReMath project, the development of these two components was refined and a new component, namely the Function component was developed. Moreover, considerable efforts were devoted to the integration of these three components.

As to the Algebraic Line component some functions were improved and new ones were developed. In particular, a lot of functions were made available through the right button of the mouse and the interface was simplified. Furthermore, the polynomial roots can be determined for a generic polynomial without the degree limit, and the “comparison function” replaced by the possibility to

insert propositions by the available editors. Some new functions were developed such as the “Edit Set” to create the Truth Set of a proposition. Moreover, some specific functions to integrate the Algebraic Line component with the other two components were added. In particular, a specific function to send to and to receive from the Algebraic Manipulator component expressions and propositions; another function, namely the “View Graph” function, was developed and inserted in the context menu, to send the expression to the Function component and visualize its graph on the Cartesian plane.

Concerning the Algebraic Manipulator component, the ITD team work was mainly devoted to make the rules and properties of the interface complete and consistent according to the chosen numeric domain. Moreover, some new rules concerning sets were added and rules to solve propositions were refined. Two specific rules, namely the “Collect” rule and the “Expand” rule, also present in CAS system were implemented. Three specific functions were added to integrate the Algebraic Manipulator component with the Algebraic Line: “factor roots function”, that can be applied only if the root of the function is determined in the Algebraic Line; “insert solution set” that inserts the solution of a proposition once it is determined in the Algebraic Line and “Instantiate variable” through which the variable of an expression in the Manipulator can be replaced by the value that such variable assumes in the Algebraic line. Moreover, through specific implemented functions expressions and propositions can be sent to and received from the Algebraic Line component.

Concerning the Function component, the work focused on the internal and external representation of this component. In particular, the graph of any expression with respect to all variables used in AL was implemented. Moreover, different graphs can be represented and distinguished by colors on the Cartesian plane, and a specific update is made when unit size is changed by dragging in the line display.

The bi-dimensional editor was developed and refined.

In the last months ITD team was mainly involved in fixing bugs in the system and making available the integration between AlNuSet and the Mathdils platform.

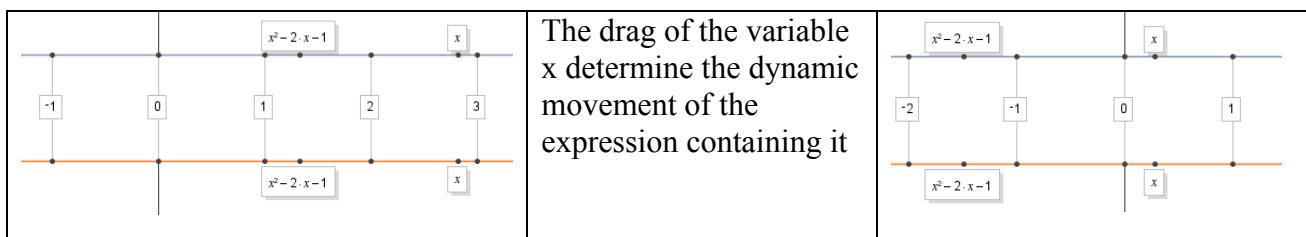
3) Description of the final form of the DDA

ALNUSET is constituted of three components: the Algebraic Line component, the Algebraic Manipulator component, and the Functions component. These three components make available techniques of quantitative, symbolic and functional nature to operate with algebraic expressions and propositions. The techniques of quantitative nature focus on numerical quantities indicated in an indeterminate way by a variable or by a literal expression, or on the numerical quantities that condition the equality or inequality of a proposition. The techniques of symbolic nature focus on the rules system to manipulate algebraic expressions and propositions symbolically preserving the equivalence through the transformation. The techniques of functional nature focus on the link between a variable and an expression containing it and on the different representations of such link to construct algebraic concepts. These techniques were designed to mediate the solution of algebraic tasks by students and to support the development of a discourse in the classroom among students and teachers on the algebraic objects, processes and relations involved in the activity.

For editing expressions and propositions to operate with, the three components make available a linear algebraic editor and a bi-dimensional algebraic editor. The editing of expressions and propositions with the linear editor occurs with the sequential insertion of characters in a field named “Quick edit” on the interface of AlNuSet.

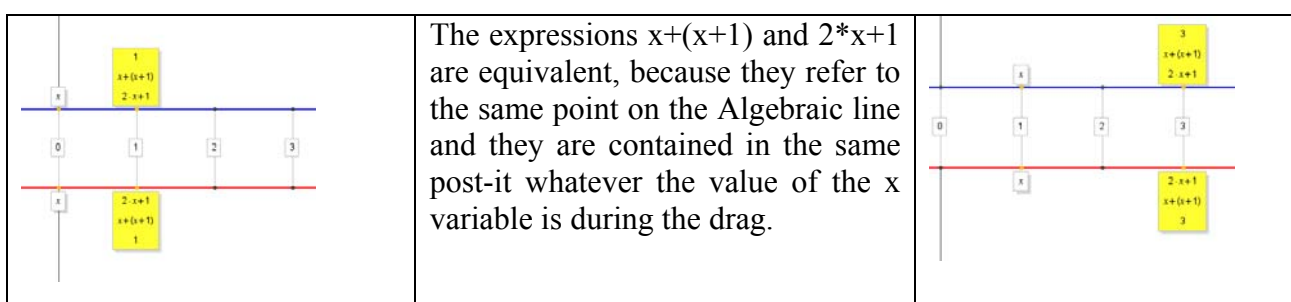
The bi-dimensional editor is available through the command “Edit” of the interface. Expressions and propositions can be edited in a bi-dimensional way either by means of commands associated to

Note that through successive applications of these geometrical models it is possible to construct complex algebraic expressions and to represent them on the line. If the constructed or edited expression contains a variable, the computer automatically computes the value of the expression on the basis of the value of the variable on the line. Once an expression containing a variable has been constructed, when the user drags the mobile point of the variable, the computer refreshes the positions of the points corresponding to the expressions containing such a variable in a automatic and dynamic manner. In this way, through the exploitation of the digital technology, the traditional number line has been transformed into an algebraic line. The following two figures report the representation of a variable and of an algebraic expression on the lines of this component. Note that the presence of two lines is motivated by operative necessities connected with the use of the geometrical models of the algebraic operations .

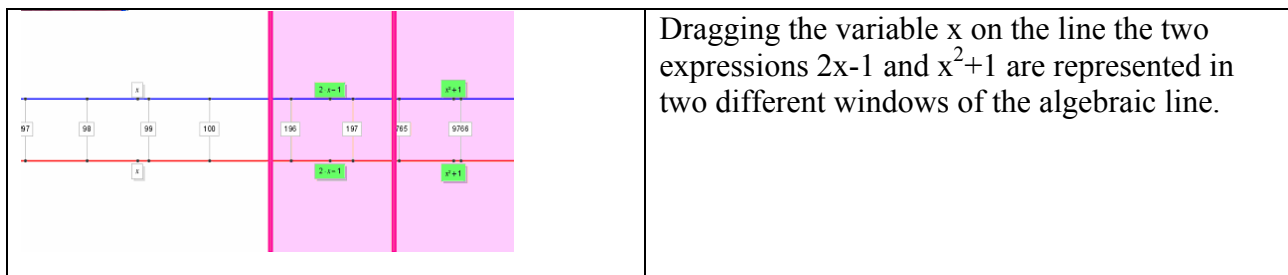


This technique, with its visual feedbacks, can be exploited to explore what an expression indicates in indeterminate way or to compare expressions among them. In the design of this component it has been decided to associate a post-it to every point represented on the line. The computer automatically manages the relation among expressions, their associated points and post-it. The post-it of a point contains all the expressions constructed by the user that denote that point. By dragging a variable on the line, dynamic representative events can occur in a post-it. These can have a great importance for the development of a discourse concerning the notions of equality and equivalence between expressions. As a matter of fact the presence of two expressions in a post-it may mean:

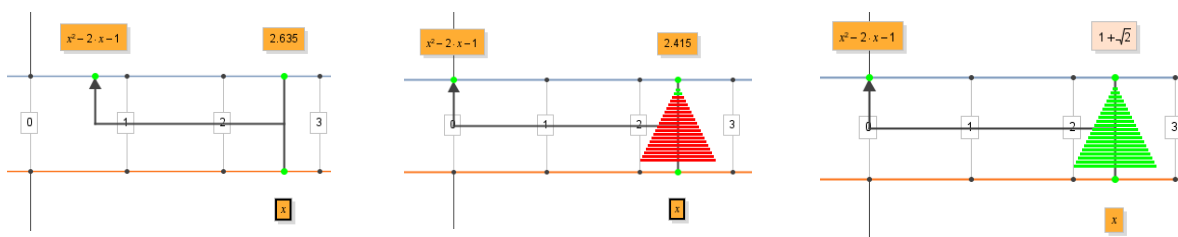
- A relationship of equality, if taking place at least for one value of the variable during its drag along the line
- A relationship of equivalence, if taking place for all the values assumed by the variable when it is dragged along the line.
- A relationship of equivalence with restrictions, if taking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



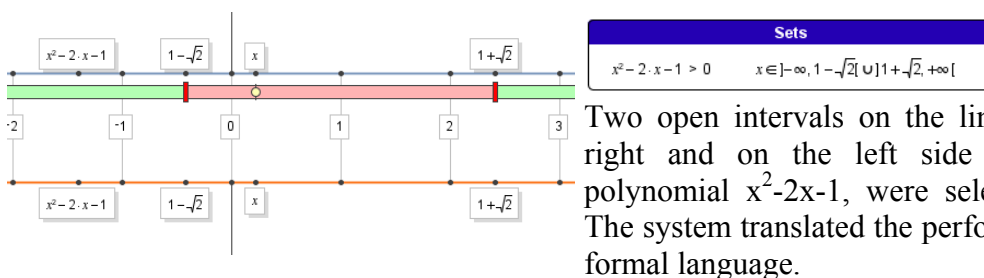
The implementation of the instrumented technique to represent expressions on the algebraic line emphasized problems of ergonomic nature that entailed new design solutions. Consider the case where during the drag of the variable, the expression is far from the variable, off the visible space of the line on the screen. In such case the user loses the control of the expression. To keep the expression under control of the user visual perception, a function of tracking of the expression was implemented which always allows to visualize the expression on the line or in its window.



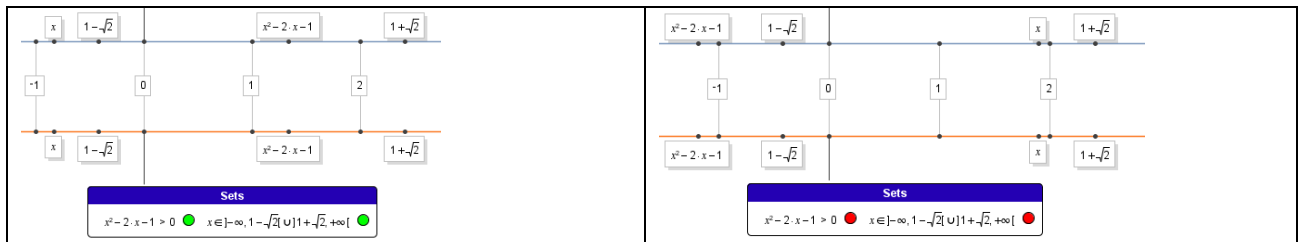
Moreover, the algebraic line component was designed to provide other two very important instrumented techniques for the algebraic activity, namely to find the roots of polynomial with integer coefficients and to identify and validate the truth set of algebraic propositions. The root of a polynomial can be found dragging the variable on the algebraic line in order to approximate the value of the polynomial to 0. When this happens, the exact root of the polynomial is determined by a specific algorithm of the program and it is represented as a point on the line. The root is always expressed in decimal numerical form and when it is possible in symbolic form.



This technique, that can be controlled by the user through his visual and spatial experience, is effective not only at a pragmatic level but also at an epistemic level, because it can concretely support the development of a discourse on the notion of root of a polynomial, as value of the variable that makes the polynomial equal to 0. The truth set of a proposition can be found via a specific graphical editor. Let us consider the inequation $x^2-2x-1 > 0$, that once edited, is visualized in a specific window of the component named “Sets”. Once the root of the polynomial associated to the inequation is represented on the line, a graphic editor can construct its truth set (see the figure).



Once the truth set of a proposition is edited, it can be validated using a specific feedback of the system. In the set window propositions and numerical sets are associated to colored (green/red) markers that are under the control of the system. The green/(red) color for the proposition means that it is true/(false) while the green/(red) color for the numerical set means that the actual variable value on the line is/(is not) an element of the set. Through the drag of the variable on the line, color accordance between proposition marker and set marker allows the user to validate the defined numerical set as truth set of the proposition (see figure below). The validation process is supported by the accordance of color between the two markers and by the quantitative feedback provided by the position of variable and of the polynomial on the algebraic line during the drag.



The feedback offered by the system during the drag of the variable is important to introduce the notions of truth value and of truth set of an algebraic proposition and to develop a discourse on their relationships. All the described instrumented techniques peculiar to the Algebraic line component make a quantitative and dynamic algebra possible.

Algebraic manipulator component

The interface of this component has been divided into two distinct spaces: a space where symbolic manipulation rules are reported and a space where symbolic transformation is realized.

User Rules Show Import... Export... Clear		
Addition $A+B \Leftrightarrow B+A$ $A+(B+C) \Leftrightarrow (A+B)+C$ $A \Leftrightarrow A+0$ $A+A \Leftrightarrow 0$ $A-B \Leftrightarrow A+(-B)$ $a_1+a_2+\dots \Rightarrow x$ $n \Rightarrow a+b$	Multiplication $A \cdot B \Leftrightarrow B \cdot A$ $A \cdot (B \cdot C) \Leftrightarrow (A \cdot B) \cdot C$ $A \Leftrightarrow A \cdot 1$ $A \cdot 0 \Leftrightarrow 0$ $-A \Leftrightarrow -1 \cdot A$ $1 \Leftrightarrow -1 \cdot -1$ $A \cdot \frac{1}{A} \Leftrightarrow 1$ $\frac{A}{B} \Leftrightarrow A \cdot \frac{1}{B}$ $\frac{1}{\frac{A}{B_1} \cdot \frac{A}{B_2} \dots} \Leftrightarrow \frac{1}{\frac{A}{B_1} \cdot \frac{A}{B_2} \dots}$ $a_1 \cdot a_2 \dots \Rightarrow x$ $n = p_1 \cdot p_2 \dots$	$a^2 - b^2$ $a \cdot a - b^2$ $a \cdot a - b \cdot b$ $a \cdot a + 0 - b \cdot b$ $a \cdot a + a \cdot b + (-a \cdot b) - b \cdot b$ $a \cdot (a+b) + (-a \cdot b) - b \cdot b$ $a \cdot (a+b) + (-1 \cdot (a \cdot b) + (-b \cdot b))$ $a \cdot (a+b) + (-1 \cdot (a \cdot b) + (-1 \cdot (b \cdot b)))$ $a \cdot (a+b) + (-1 \cdot a \cdot b + (-1 \cdot b \cdot b))$ $a \cdot (a+b) + (-1 \cdot b \cdot a + (-1 \cdot b \cdot b))$ $a \cdot (a+b) + (-b \cdot a + (-1 \cdot b \cdot b))$ $a \cdot (a+b) + (-b \cdot a + b \cdot b)$ $a \cdot (a+b) + (-b \cdot (a+b))$ $(a+b) \cdot (-a+b)$ $(a-b) \cdot (a+b)$
Powers $A^n \Leftrightarrow A \cdot A \cdot \dots$ $A^{n_1+n_2+\dots} \Leftrightarrow A^{n_1} \cdot A^{n_2} \cdot \dots$ $(A_1 \cdot A_2 \cdot \dots)^n \Leftrightarrow A_1^n \cdot A_2^n \cdot \dots$ $(A^n)^m \Leftrightarrow A^{n \cdot m}$ $A^{-n} \Leftrightarrow \frac{1}{A^n}$ $A^{\frac{1}{2}} \Leftrightarrow \sqrt{A}$	Distribute / Factor $A \cdot (B_1+B_2+\dots) \Leftrightarrow A \cdot B_1 + A \cdot B_2 + \dots$	
Computation $A \Rightarrow (A)$ Remove extra () Simplify numerical expression Expand Collect Eliminate variable	Solving $A \leq B \Leftrightarrow B \leq A$ $A \leq B \Rightarrow A-B \leq 0$ $A \leq B+T \Rightarrow A-T \leq B$ $A+T \leq B \Rightarrow A \leq B-T$ $T \cdot A \leq B \Rightarrow A \leq \frac{B}{T}$ $A^{\frac{p}{q}} \leq B \Rightarrow A^p \leq B^q$	
Logic and Set Simplify boolean expression		

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

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

The key idea characterizing the design of the Algebraic Manipulator component is the possibility to exploit pattern matching procedures of computer science to transform algebraic expressions and propositions through a structured set of basic rules that are deeply different from those of CAS. In CAS pattern matching procedures are exploited according to a pragmatic perspective oriented to produce a result of symbolic transformation that could be very complex, as in the case of commands like factor or solve. As a consequence, the techniques of transformation can be obscure for a not expert user. In the Algebraic Manipulator component of AlNuSet pattern matching procedures have been exploited according to three specific pedagogical necessities.

The first necessity is to highlight the epistemic value of algebraic transformation as formal proof of the equivalence among algebraic forms. To this aim we have designed this manipulator with a set of basic rules that correspond to the basic properties of addition, multiplication and power operations, to the equality and inequality properties between algebraic expressions, to basic logic operations among propositions and among sets. Every rule produces the simple result of transformation that is reported on the icon of its corresponding command of the interface, and this makes rule and result control easy. Moreover a fundamental function of this component allows the student to create a new transformation rule (user rule) once this rule has been proved using the rules of transformation

available in the interface. For example, once the rule of the remarkable product $a^2 - b^2 = (a+b)*(a-b)$ has been proved, it can be added as new user rule in the interface $a^2 - b^2 \leftrightarrow (a+b)*(a-b)$ and it can be used successively to transform other expressions or part of them whose form match with it (for example it can be applied on expressions such as $x^2 - 3^2$, $(4x+1)^2 - x^2$ but not on $x^2 - 9$ because it has before to be transformed into $x^2 - 3^2$). Moreover, a specific command allows to represent on the algebraic line every transformed expression automatically. Through this command it is possible to verify quantitatively the preservation of the equivalence through the transformation, observing that all transformed expressions belong to the same post-it when every variable is dragged along the line. These characteristics of the algebraic manipulator of AlNuSet may be very important at the epistemic level because they can be effectively exploited to support the comprehension of the algebraic manipulation in terms of formal proof of the equivalence between two algebraic forms.

The second necessity is to support the integration of practice of quantitative and manipulative nature. In this manipulator three rules allow the user to import the root of a polynomial, the truth set of a proposition and the value assumed by a variable on the algebraic line from the Algebraic line component to be used in the algebraic transformation. For example the rule “Factorize” uses the root of polynomial found in the Algebraic Line to factorize it. The way in which this rule works, makes the factorization technique of AlNuSet different from that of CAS. In CAS this technique is totally under the control of the system, and the result may appear rather obscure for not expert users. In AlNuSet, the factorization can be applied on the polynomial at hand only if its roots have been previously determined on the algebraic line. In AlNuSet the distribution of tasks between user and computer and the way of their interaction may contribute to understand the link between the factorization of a polynomial and its roots.

The third necessity is to offer more powerful rules of transformation when this is a necessity for the activity and specific meaning of algebraic manipulation have been already constructed. Two specific rules, also present in CAS are available in this manipulator. They determine the result of a numerical expression and the result of a computation with polynomials respectively. These rules of transformation contribute to increase the pragmatic value of the instrumented technique of algebraic transformation in AlNuSet and they can be used to introduce to the use of CAS

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

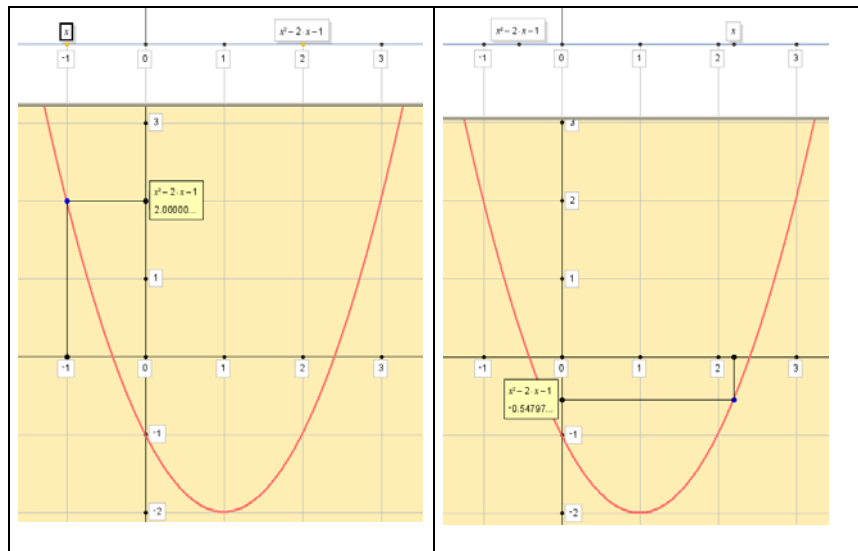
Functions component

The main idea characterizing the design of the Functions component is the possibility to connect a dynamical functional relationship between variable and expression on the algebraic line with the graphical representation of the function on the Cartesian plane. As a consequence the interface of this component has been equipped with the Algebraic line and a Cartesian plane. This idea makes this component deeply different from other environments for the representation of function on the Cartesian plane. Through a specific command and the successive selection of the independent

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

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

This instrumented technique supports the integrated development of a dynamic idea of function with a static idea of such a notion. The functional relationship between variable and expression is visualized in a dynamic way on the algebraic line through the drag of the variable point, and in a static way on the Cartesian plane through the curve. The movement of the point along the curve during the drag of the variable on the algebraic line supports the integration of these two ideas, showing that the curve reify the infinite couples of values corresponding to the variable and to the expression on the line. This instrumented technique can be very useful to orient the interpretation of the graphic on the Cartesian plane and to develop important concepts of algebraic nature.



For example, it contributes to assign an algebraic meaning to the intersection of two curves (observing that for the value of the variable that determines the intersection, the two expressions are contained in the same post-it on the algebraic line) or to the intersection of a curve with the x-axis (observing that in this case the expression is contained in the post-it of 0). Other examples are related to the construction of meaning for the sign of a function (in terms of the position of the corresponding expression on the line with respect to 0), or to relationships of order among functions (in terms of the respective position of the expressions on the algebraic line)

4) Perspectives

The ITD working group that has realized the design and the development of AlNuSet is actually involved in the realization of a Spin-off for its commercial exploitation. Currently, the demand for the spin-off and its business plan have been presented to the administration of the National Research Council. The approval of the demand is forecast for the end of December 2008.

The deployment of AlNuSet will be one of the aspect that will characterize the mission of this spin-off.

In order to favor the deployment of this system in school context specific developments are foreseen on the product, such as for examples :

1. The transformation of AlNuSet from a prototype into a commercial product with some improvements of the interface, its localization in different language, the implementation of the user profile and of an effective help-on-line.
2. The development of a web-based version of AlNuSet both in the client and the server parts.
3. The development of new applications to support on-line learning activities.

Another aspect that will characterize the mission of the spin-off are the development of services for teachers training and the deployment of didactical scenarios to support the familiarization with the system and its integration in the didactical practices related to different domains of the mathematical knowledge (arithmetic, numerical sets, algebra, function etc.).

The activities for the deployment and the development of the product AlNuSet by the spin-off company will come with research activities on the impact of its use within the school system by the ITD-CNR.

In particular, ITD-CNR is interested to continue the study of the impact and of the mediation role of AlNuSet in the scholastic context. More in particular, ITD CNR intends to study the improvements that the use of AlNuSet can provide in teaching and learning processes in mathematics, the conditions and models of use that can favour this improvement, the changes on the curricular level derived by its employ.

Hence, the missions and the interests of Spin-off company and ITD-CNR concerning AlNuSet are different and complementary at the same time and we think that their collaboration can be fruitful both for the deployment and the development of the AlNuSet product but also for the research activity in the field of the educational technology for teaching and learning mathematics.

DDA 4: Machine Lab

Authors: Psycharis Giorgos and Moustaki Foteini

Educational Technology Lab

Table of content

1	Introduction.....	2
2	Last evolutions of the DDA	2
2.1	The jMonkey Engine.....	2
2.2	The new GUI.....	3
2.3	The Property Editor.....	3
2.4	The camera viewpoints	4
3	History of the design and development of the DDA.....	5
	<i>First level: The design of MaLT as an extension based on the MachineLab platform</i>	5
	<i>Second level: The integration of programming with dynamic manipulation to the 3d space as the object of the extension</i>	6
4	Description of the final form of the DDA.....	9
4.1	The Turtle Scene	9
4.1.1	Description.....	9
4.1.2	Scene camera manipulation	10
4.2	The Logo Editor and Variation Tools.....	10
4.2.1	The Logo Editor.....	10
4.2.2	Variation Tools	10
4.3	The Property Editor.....	11
4.3.1	The Toolbar.....	11
4.3.2	The Object Palette.....	12
4.3.3	Property Editor.....	12
4.4	Using the MaLT Environment's features	13
4.4.1	Using the 1d Variation Tool	13
4.4.2	Working with the 2d Variation Tool.....	14
4.4.3	Working with the Vector Variation Tool (VVT).....	16
4.4.4	Inserting and manipulating ready-made stereometric objects.....	17
4.4.5	Manipulating the camera viewpoints	18
4.4.6	Creating logo-constructed 3d stereometric objects	19
5	Perspectives	23
5.1	In terms of developments.....	23
5.2	In terms of deployment	24

1 Introduction

MaLT is a programmable constructionist environment that allows the creation, exploration and dynamic manipulation of 3d geometrical objects. It is designed to integrate the symbolic notation –in the form of Logo programs– with the dynamic manipulation of 3d geometrical objects that are graphically represented in a 3d Turtle Scene. The geometrical objects visualised in the environment's Turtle Scene are either *constructed* by the user when running logo procedures and commands or *inserted* by the user after selecting them from a library that offers numerous ready-made stereometric objects, such as cylinders and cones. Employing multiple linked representations, MaLT allows any actions performed on one of its representations to generate immediate changes to another one or the rest of its representations, providing meaningful feedback to its users.

2 Last evolutions of the DDA

During the last months the MaLT environment has changed radically. The most important change was the substitution of the 3impact game engine with the jMonkey 3d game engine. jMonkey is java based and renders complex 3d scenes significantly faster than the 3impact, which results in boosting substantially the Turtle Scene's performance. Therefore, any object manipulations in the Turtle Scene (direct or indirect) now provoke the environment's immediate graphical response in speeds comparable to other 2d dynamic geometry tools. A second change made was the inclusion of a set of ready made stereometric objects and a special GUI to dynamically manipulate important features and measure areas, volumes, segments and locations. A third change was the development of a GUI to make the environment more useable and a final change was the inclusion of dynamic controllers for camera movement. The elements modified since the last deliverable and the elements introduced in the new MaLT are presented below in detail.

The elements modified since the last deliverable and the elements introduced in the new MaLT are presented below in detail.

2.1 The jMonkey Engine

MachineLab, the platform on which the MaLT was based, was developed in Borland Delphi, using a 3D game engine (the 3impact), a high level interface to DirectX for implementing 3D games, which ran under Microsoft Windows. However, the graphical user interface (GUI) evolved and ported from Delphi to the Java language while the underlying programming environment was re-developed to implement Java bindings for the 3impact engine.

In the new version of MaLT the 3d game engine used is no longer the 3impact. The jMonkey (<http://www.jmonkeyengine.com/>) 3d game engine constitutes a reliable high-performance Java scenegraph Application Programming Interface (API) whose main characteristic is that it is designed to allow for complex 3d scenes to be rendered faster compared to the 3impact engine. Due to its architecture, jMonkey engine boosts MaLT's Turtle Scene performance which now responds effectively to any dynamic manipulations performed using the environment's Variation Tools or manipulations performed to the objects themselves when dragging them, using their handles. The jMonkey engine enabled MaLT to speed up and work efficiently regardless of the number and the complexity of the objects present in the Turtle Scene.

2.2 The new GUI

The previous versions of MaLT consisted of multiple windows which the users occasionally had to move around and squeeze up together on their desktop so as to have in plain view all the components needed for their experimentations with the MaLT environment. The Turtle Scene was quite narrow while any changes to its dimensions caused continuous problems not only to the component's function but also the space distribution among the environment's windows.

The new MaLT environment consists solely of one window that is divided in two main areas, the left and the right one. The left area –the larger one– constitutes the environment's Turtle Scene which is now large enough and resizable. The component that appears on the right area, however, is of the user's choice. By clicking on the corresponding tab on the top of the right area, the user may select between two different components to display: the Logo Editor and the Property Editor.

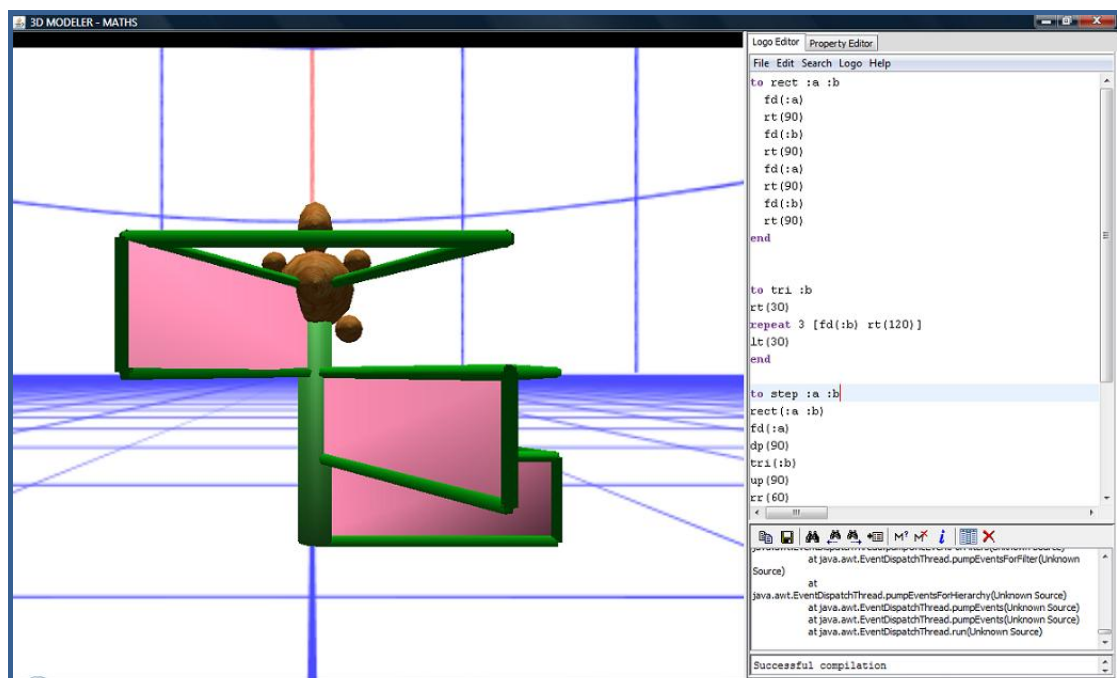


Figure 1: The new MaLT environment.

The new Logo Editor incorporates two of the previous MaLT's components: the Logo programming interface and the 1d Variation Tool. The 1dVT appears just below the Logo programming interface, only after the user defines a Logo procedure, runs it, attributing specific values to each variable, and clicks on the trace the turtle has left in the 3d Scene.

The Property Editor constitutes a new feature which will be presented next.

2.3 The Property Editor

The Property Editor is designed to provide the user a library of ready-made objects which she/he may insert in the Turtle Scene and manipulate them dynamically or by editing their properties using the Editor's corresponding value fields. It consists of the Toolbar, the Palette and the main area of the Property Editor.

By clicking on Toolbar's icons the user initialises and controls the object inserting procedure. Once the procedure begins the user may select from a variety of ready-

made objects, such as cones, spheres and cylinders, the type object she/he wishes to insert in the Turtle Scene. After selecting the object to be inserted, the main area of the Property Editor –just below the Palette– is activated and its fields become accessible to the user. The fields' default values become available for modifications.

The properties to be defined in the Property Editor are not the same for all the available types of ready-made objects. However, there is a set of standard properties that appear for all the available types. These are:

- **Name:** Each object in the 3d Scene has a unique name by which it is addressed when running logo procedures.
- **Colour:** A palette of colours from which the user may select the one for her/his object appears.
- **Transparency:** The value of the object's transparency appears. The given values should be between 0 and 255.

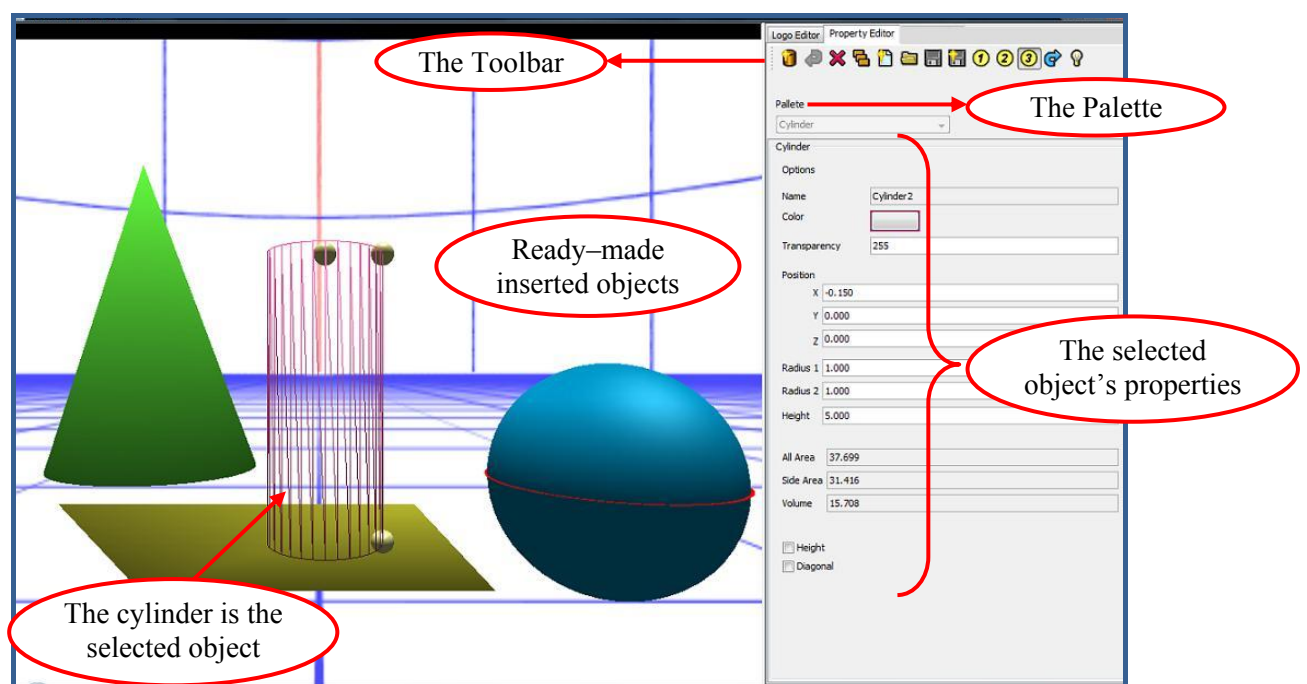


Figure 2: The new Property Editor tab.

Other kinds of properties that may appear are the: Radius, Height, Sides, Length, Width, Length of diagonal, Centre, Point and Position (X, Y, Z). Apart from the Property fields whose values are defined by the user, there is also a set of fields whose values are calculated automatically by the environment when the object is inserted in the Turtle Scene. These also vary according to the object's type. The most common of them are the: All Area, Side Area and Volume.

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

2.4 The camera viewpoints

In the new MaLT environment the user has at his disposition three cameras so as to

observe her/his constructions inside the Scene from different viewpoints. The three available cameras are:

- The first one is placed on the top of the Scene and shows the Floor View of the 3d constructions inside the Scene.
- The second one is placed on the one of the Scene's sides and shows the Side View of the 3d constructions inside the Scene.
- The third one is the main camera of the Scene and is turned towards the negative direction of the Z axis. It is the only camera whose position the user may change, using logo commands or combinations of keyboard buttons.

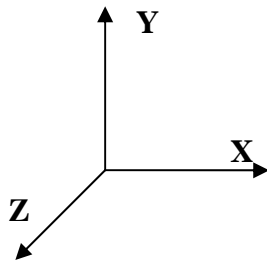


Figure 3: The Scene's XYZ coordination system.

- Shift+W: the camera zooms in,
- Shift+S: the camera zooms out,
- Shift+A: the camera moves left on a circumference of a sphere,
- Shift+D: the camera moves right on a circumference of a sphere
- Shift+Q: the camera moves up on a circumference of a sphere
- Shift+E: the camera moves down on a circumference of a sphere
- Shift+X: resets camera to its initial position

The Property Editor's Toolbar contains three icons that allow the user to shift the camera viewpoints.



The three camera viewpoints to be selected

Figure 4: The third camera is currently activated in the Property Editor's Toolbar.

3 History of the design and development of the DDA

First level: The design of MaLT as an extension based on the MachineLab platform

MaLT is a programmable constructionist environment developed entirely within the ReMath project. From a technical point of view, MaLT can be considered as a microworld built on the MachineLab platform. MachineLab was originally designed to be stand-alone application, developed in Borland Delphi and using a 3d game engine (the 3impact). However, at the beginning of the ReMath, the agenda behind the existing MachineLab platform was very board from the didactical point of view, as it was used to create programmable 3d simulations for a large variety of domains, like science (simulations of physical phenomena), chemistry and gaming. Thus, ETL needed to design a new environment that would inherit the 3d features of the MachineLab platform, but it would provide means for the creation of 3d geometrical constructions, encompassing multiple linked representations that would allow the manipulation and exploration of 3d objects. Drawing on its long experience in designing constructionist digital media and especially in designing 2d tools for studying geometrical notions, the ETL team developed the "MachineLab Turtleworlds" (MaLT) as an extension the MachineLab platform.

In absence of any previous version of MaLT, its development was not a straightforward process. The developers had to redesign and redevelop aspects of the MachineLab to a quite deep extent as the ETL team asked for functionalities that were difficult from a technical point of view and didn't exist in the previous MachineLab version (for instance selecting specific points on the 3d space). As a result, the programming interface between the game engine and the users was changed from Delphi to Java. That meant that the programming aspect of the environment had to be redeveloped and import a Java language on the 3d impact. From this point on, the Logo programming language was implemented in Java, using a custom-designed compiler that translates Logo programs into Java. Another reason for this choice was that our team wanted different windows for each representation, which was impossible in the previous platform. We wanted our students to have kinaesthetic control over 3d representations which could not happen in Delphi since it didn't support the linking between different kinds of software or windows.

Second level: The integration of programming with dynamic manipulation to the 3d space as the object of the extension

Wishing to extend the integrated “Programming–Dynamic Manipulation” idea to the 3d space, the MaLT environment was designed to embody functionalities from the two main groups of computational environments: programming tools (e.g. Logo-like microworlds) and expressive tools (e.g. dynamic geometry). Such functionalities are: a) the symbolic expression inherited from Logo as a programming language and b) the dynamic manipulation of geometrical objects with specially designed tools.

ETL has a long history in developing environments that integrate those kinds of functionalities in the 2d space. In order to define the ontological principles underlying the design of MaLT, the team had to think deeply about the turtle's move in the 3d space and its connection to the notion of vector. Viewing the vector in the MaLT as an oriented journey from one initial point to another and the Turtle Geometry as Vector Geometry, the turtle was designed to move in the 3d space in two ways: using the Cartesian and the polar geometrical system.

To combine symbolic notation through a programming language with dynamic manipulation of the geometrical objects, three Variation tools were built: the 1d Variation Tool (1dVT), the 2d Variation tool and the Vector Variation Tool (VVT). These tools were designed in chronological order on the base of the process achieved in the MaLT development during the ReMath Project. The 1dVT was offered in the first version of MaLT, the 2dVT in the second and the VVT just after the release version (Month 19).

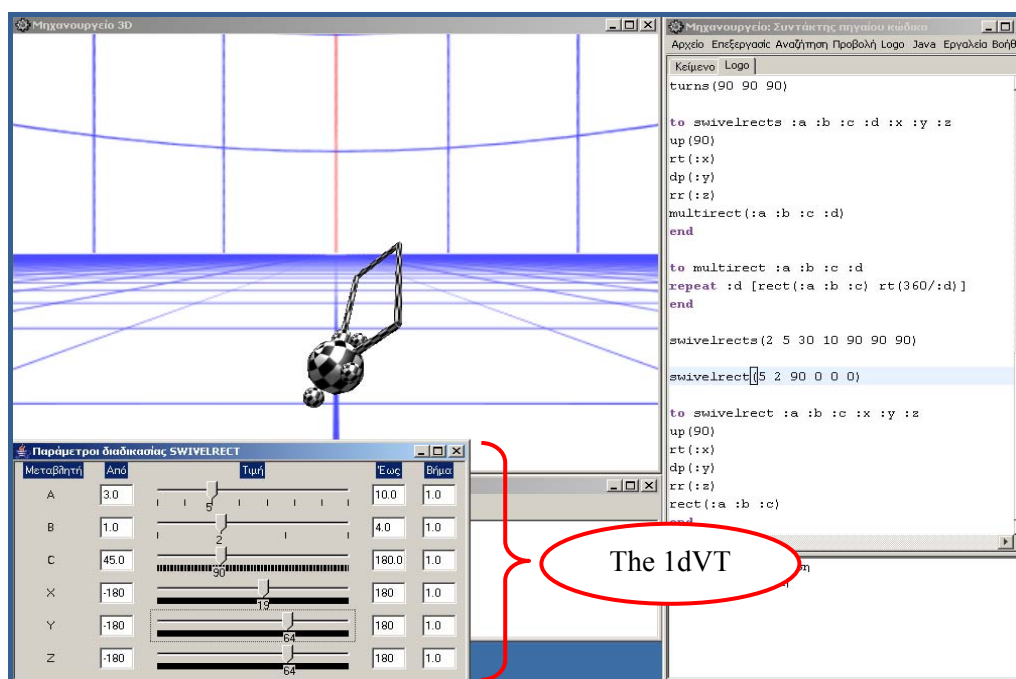


Figure 5: The first version of MaLT included the 1dVT.

The 1dVT (Figure 5) consists of sliders, each of which corresponds a variable included in the Logo procedure that creates the figure on the Scene. It is activated after executing the procedure using specific values and then clicking on the turtle's trace. Dragging one of the sliders causes the figure to change dynamically, as the values of the variable change sequentially. Apparently, what is manipulated is not the figure itself, but the value of the logo procedure's variable.

The 2dVT on the other hand allows the co-variation of two variables at the time. It is activated through the 1dVT, after selecting two variables and defining which one will be represented in which axis.

It is in the form of an orthogonal pad on which the mouse can be freely dragged, leaving behind a trace. Each position on the pad corresponds to a specific value for each of the selected variables. The changes in the mouse's position cause respective changes to the 1dVT sliders' values as well as to the figures in the Scene. The 2VT is used not just to represent a relationship but mainly to define and implement it.

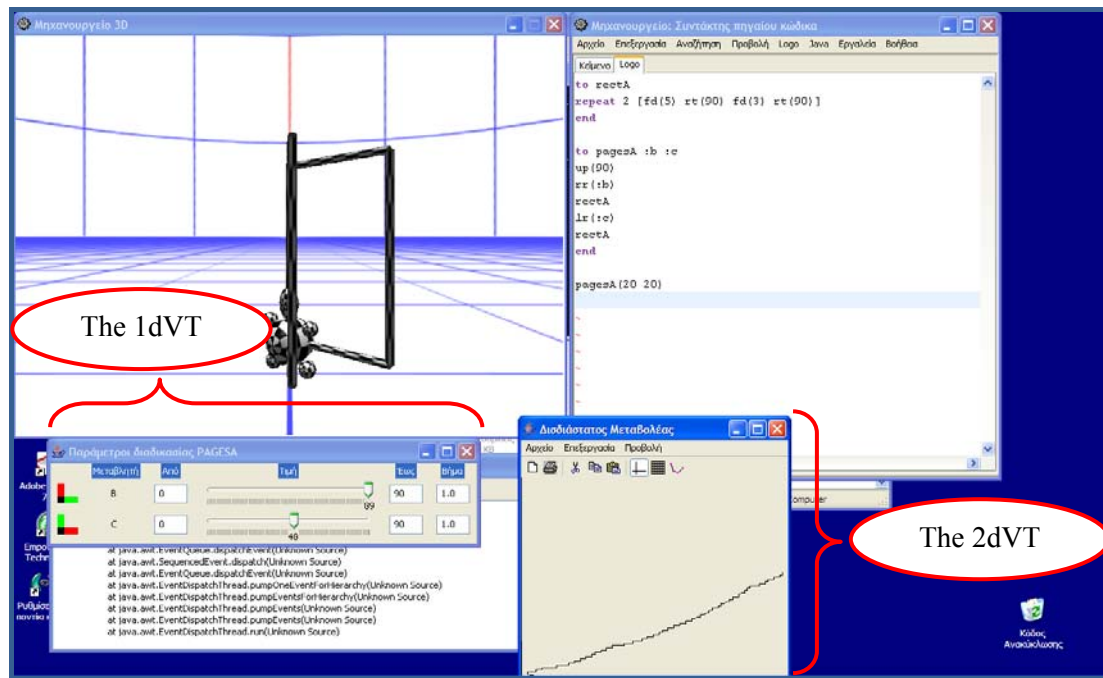


Figure 6: The second version of MaLT included the 1dVT and 2dVT.

Finally, the last developed tool is the Vector VT. The VVT requires a logo procedure with at least three variables. It is activated when clicking on the icon on the right part of a 1dVT slider. A window appears and the user may select whether the slider's variable will represent in the polar coordinate system the r (the length), the θ (the angle between the vector's projection on the xz plane and the x -axis) or the ϕ (the angle between the vector and the xz plane) coordinate.

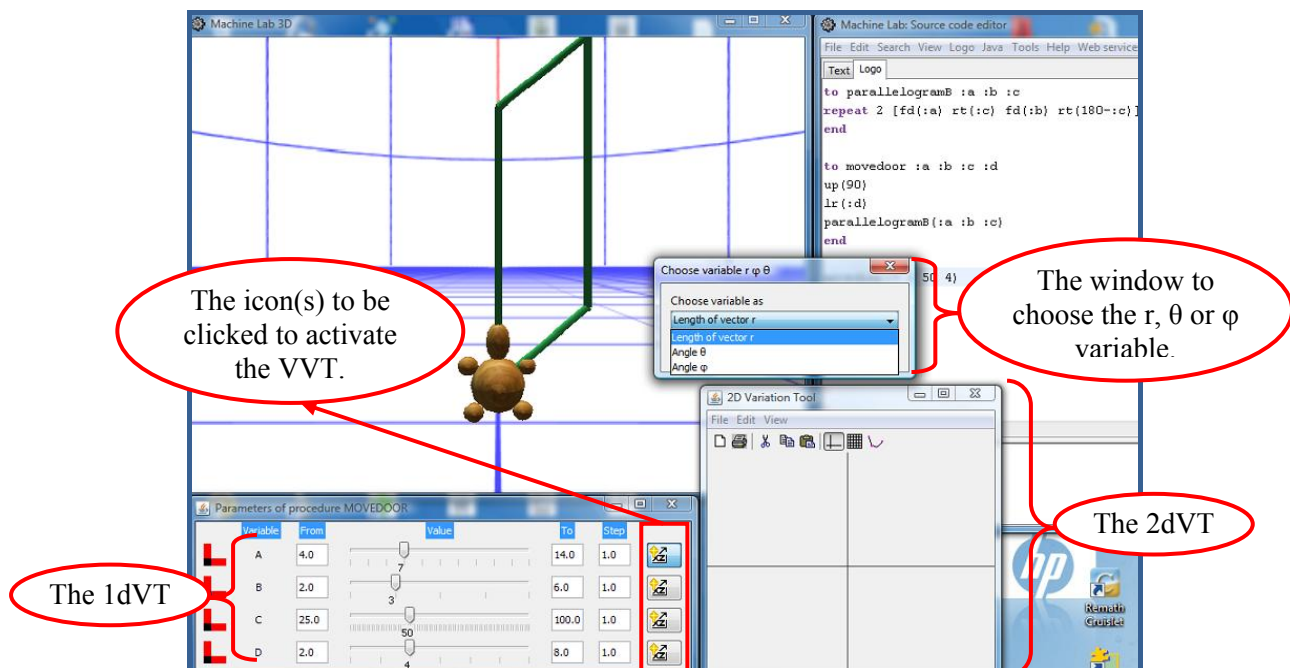


Figure 7: The release version of MaLT included the 1dVT, the 2dVT and the VVT.

It consists of two vector-like representations (the constituent projections) and a resultant vector representation (Figure 4). Using the first vector-like representation, the user can dynamically manipulate the vector's length and rotate it to control the value of angle θ , while using the second one the user can manipulate the vector's

length and rotate it to control the value of angle φ . The changes performed on the two vector-like representations are reflected on the resultant vector representation, which is not available for direct manipulations.

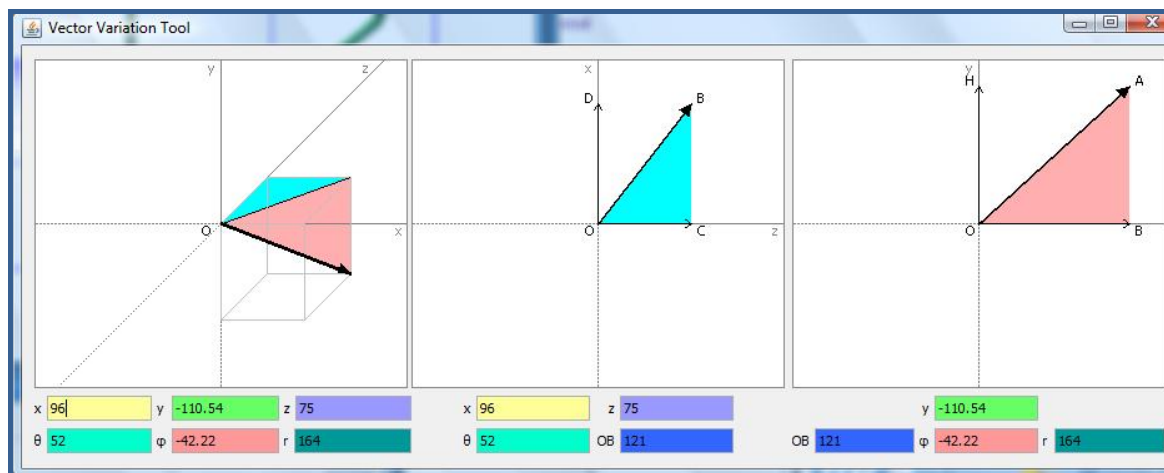


Figure 8: The Vector Variation Tool (VVT).

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

The third level of development is described in detail at the “Last Evolutions of the DDA” section.

4 Description of the final form of the DDA

The new MaLT environment consists of two main areas, the left and the right one. The left area –the larger one– constitutes the environment's “Turtle Scene”, where user-constructed or ready-made 3d figures are graphically visualized. The component that appears on the right area, however, is of the user's choice. By clicking on the corresponding tab on the top of the right area, the user may select between two different components to display: the Logo Editor (including the Variation Tool) and the Property Editor.

4.1 The Turtle Scene

4.1.1 Description

The Scene is a 3d grid-like interface where 3d geometrical objects are graphically represented. The 3d geometrical objects that appear in the Scene are either *constructed* by the user when running logo procedures or *inserted* by the user after selecting them from a library that offers numerous ready made stereometric objects, such as cylinders, spheres and cones.

The 3d turtle doesn't appear in the Scene by default. The user must type and run, at the Logo Editor, at least one command that produces a graphical result in the Scene so as to make the turtle visible. The trace the turtle leaves when moving inside the Scene –a thin 3D cylindrical line– is selectable. Clicking on this trace causes the 1dVT tool to activate and appear on the right part of the interface, right below the Logo area.

4.1.2 Scene camera manipulation

The user may look at her/his constructions inside the Scene from different viewpoints. Three cameras are available:

- The first one is placed on the top of the Scene and shows the Floor View of the 3d constructions inside the Scene.
- The second one is placed on the one of the Scene's sides and shows the Side View of the 3d constructions inside the Scene.
- The third one is the main camera of the Scene and is turned towards the negative direction of the Z axis, as shown to the XYZ representation below. It is the only camera whose position the user may change, using logo commands or combinations of keyboard buttons.

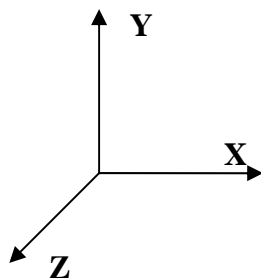


Figure 9: The semantics of the XYZ coordination system.

- Shift+W: the camera zooms in,
- Shift+S: the camera zooms out,
- Shift+A: the camera moves left on a circumference of a sphere,
- Shift+D: the camera moves right on a circumference of a sphere
- Shift+Q: the camera moves up on a circumference of a sphere
- Shift+E: the camera moves down on a circumference of a sphere
- Shift+X: resets camera to its initial position

4.2 The Logo Editor and Variation Tools

4.2.1 The Logo Editor

The Logo Editor opens up when clicking on the Logo tab situated on the right area of the MaLT environment. It consists of two interfaces: a Program Editor where the user may type commands, define new procedures and execute them and a Parser where the user may view the output of his actions on the Editor.

By running logo procedures and logo commands in the Logo Editor the user may construct 3d figures in the Turtle Scene, create and manipulate 3d stereometric objects (e.g. cylinders, spheres and cones), and control the camera so as to change the selected viewpoint. A logo command is executed when placing the cursor at the command's line and pressing the keyboard's Insert button, while when pressing the F6 button the whole procedure is executed, regardless of the line the cursor is placed.

4.2.2 Variation Tools

The MaLT Variation Tools provide users the opportunity to dynamically manipulate the values of an executed Logo procedure's variables. In the final version of MaLT it consists of the *Uni-dimensional Variation Tool (1dVT)*, the *Two-dimensional Variation Tool* and the *Vector Variation Tool*.

- *The Uni-dimensional Variation Tool (1dVT)*: The 1dVT appears just below the Logo Editor, only after the user defines a Logo procedure, runs it attributing specific values to each of its variables and clicks on the trace the turtle has left behind when constructing the graphical outcome of the procedure in the 3d

Scene. It consists of “number line”-like sliders, each of which corresponds to one of the variables used in the Logo procedure.

- *The Two-dimensional Variation Tool (2dVT):* The 2dVT allows the co-variation of two of the Logo procedure’s variables and is activated when selecting those two variables from a window that appears right next to the 1dVT. The user decides which variable will correspond to which axis (the X or the Y) on the 2dVT’s orthogonal pad and drags the mouse on the pad’s interface to co-vary of the two variables’ values.
- *The Vector Variation Tool (VVT):* The VVT allows the co-variation of three variables by using two 2d representations of a vector defined by these variables according to an (r, ϕ, θ) polar semantic in the 3d space. The VVT requires a Logo procedure of at least three variables and is activated when clicking on the icon next to the 1dVT. A pop-up menu appears so as for the user to select which variable will to correlate to r , ϕ and θ (r stands for length, θ for the angle between the vector’s projection on the xz plane and the z -axis and ϕ for the angle between the vector and the xz plane). The three variables’ values can be manipulated by dragging and rotating vectors in the VVT window that appears.

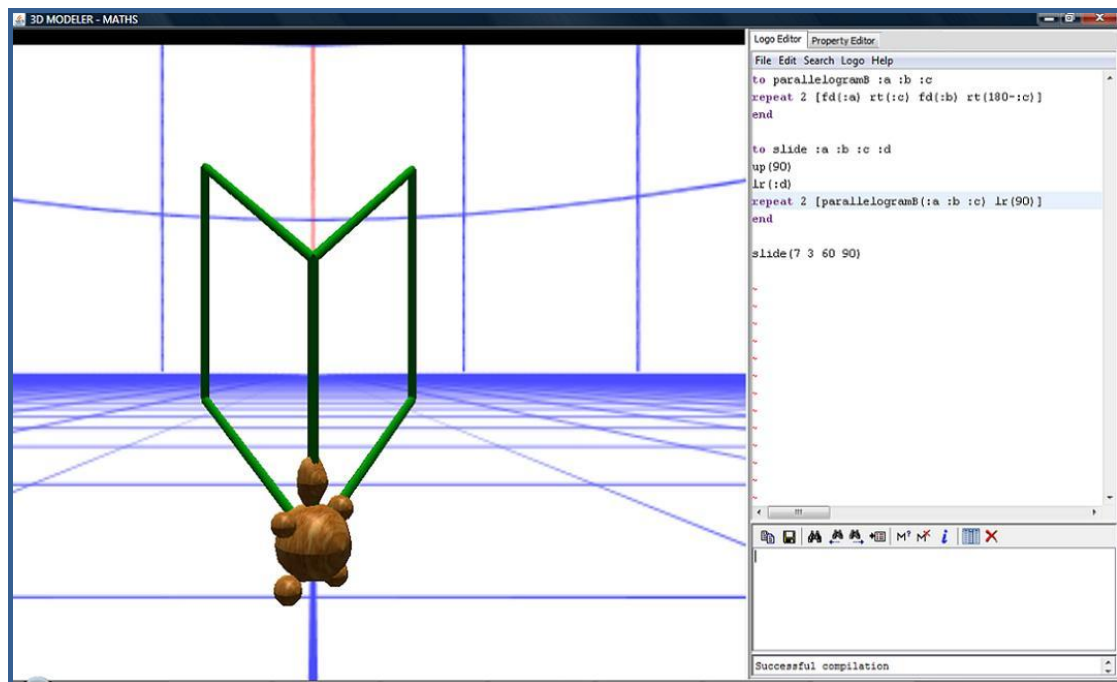


Figure 10: The Turtle Scene and Logo Editor in the final version of MaLT.

4.3 The Property Editor

Just like the Logo tab, the Property Editor tab is situated on the right part of the new MaLT interface. It is designed to provide the user a library of ready made objects which she/he may insert in the Scene and manipulate them by editing its properties. It consists of the Toolbar, the Palette and the main area of the Property Editor.

4.3.1 The Toolbar

In order to insert and manipulate ready-made objects, the user has at his disposal several features in the form of icons at the Editor’s Toolbar. These are:

- **New Object:** Clicking on the New Object icon, the Object Palette opens up and

the object inserting procedure begins. Deselecting the icon the New Object procedure is aborted.

- **Insert Object:** It consummates the object insertion procedure. After the object is inserted the insert Object procedure deactivates.
- **Delete Object:** It removes the selected object off the Scene. A pop up window asks the user whether the wants to proceed or not.
- **Unfold Object:** It causes the generation of a 2d object that has the same area as the selected object. The button is active only if there is a selected object and if the selected object can be unfolded.
- **New Activity:** It deletes any ready-made object in the Scene and begins a new activity for the user.
- **Save:** It saves the activity in an already existing file or a new file if the activity is not saved before.
- **Save as:** It saves the activity in a new file.
- **Camera button 1 (Floor view):** The floor view camera is activated or deactivated.
- **Camera button 2 (Side view):** The side view camera is activated or deactivated.
- **Camera button 3 (Main view):** The main view camera is activated or deactivated.
- **Reset main view:** Resets camera 3.
- **Light:** By clicking on this button a manipulable light source appears in the Scene. Deselecting it the light source is removed.

4.3.2 *The Object Palette*

The Object Palette contains ready-made objects which the user may insert in the Scene using the *Insert Object* button. The kinds of objects the user may insert are:

- Spheres
- Cylinders
- Cones
- Pyramids
- Orthogons
- Canonical Primitives
- Canonical Polygons
- Lines
- Planes
- Line Segments
- Circles

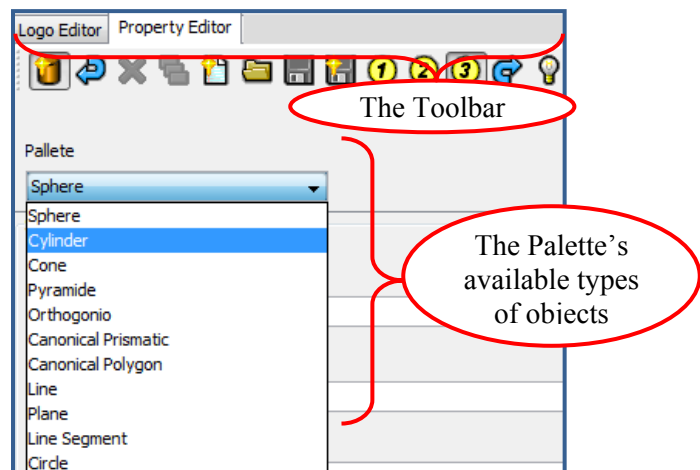


Figure 11: The Toolbar and the Object Palette.

4.3.3 *Property Editor*

The Property Editor is activated each time the user selects a ready-made object from the 3d Scene. The type of the selected object appears in the Palette while the Property Editor's fields –just below the Object Palette– become available to the user so as to modify the attributed values. Changing a value in one of the Property Editor's fields produces an immediate visual result to the object inserted in the Scene.

The Property Editor is also activated each time the user begins an object inserting procedure by clicking on the New Object button. Since the properties to be defined in the Property Editor are not the same for all the available types of ready-made objects, the user has to choose first from the Palette the kind of object she/he wishes to insert. The default values for the selected type appear on the Editor. The user may modify these values so as to define the kind of properties her/his object wishes to hold. Although for each type different properties appear on the Property Editor, yet, there is a set of standard properties that appear for all the types of objects. These are:

- **Name:** Each object in the 3d Scene has a unique name by which it is addressed when running logo procedures.
- **Colour:** A palette of colours from which the user may select the one for her/his object appears.
- **Transparency:** The value of the object's transparency appears. The given values should be between 0 and 255.

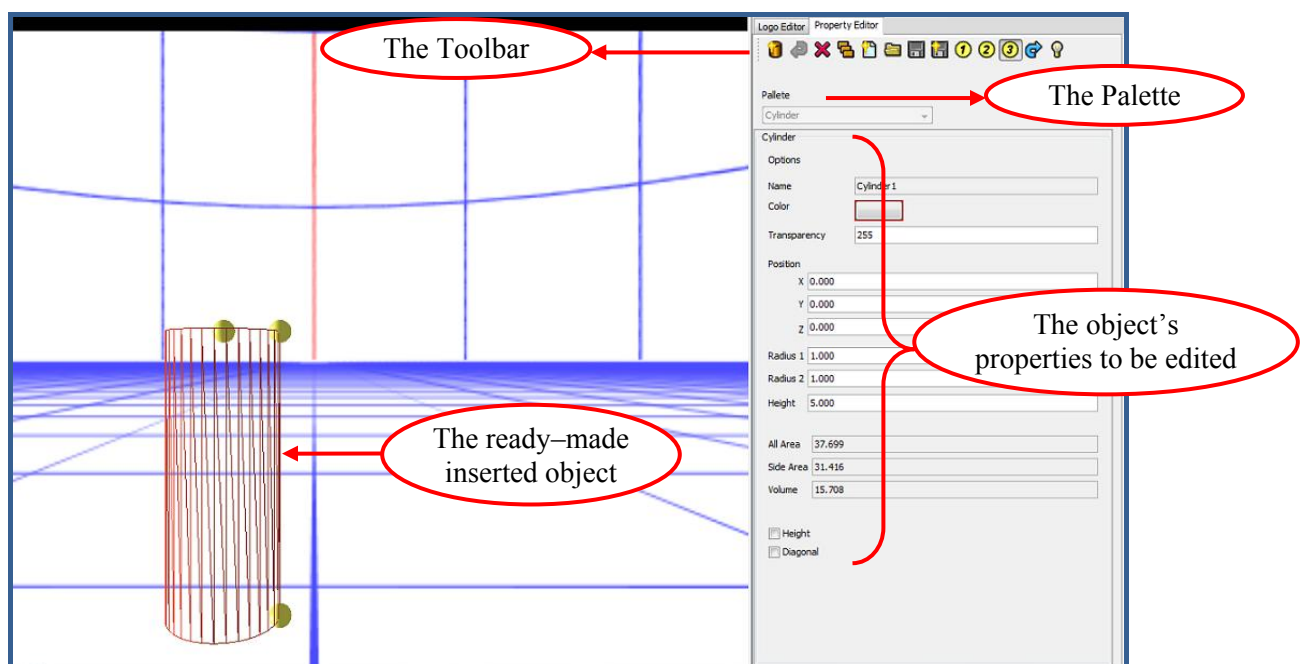


Figure 12: The Turtle Scene and the Property Editor tab in the new MaLT.

Other kinds of properties that may appear according to the type of the object inserted in the Turtle Scene are the: Radius, Height, Sides, Length, Width, Length of diagonal, Centre, Point and Position (X, Y, Z).

Apart from the Property fields whose values are defined by the user, there is also a set of fields whose values are calculated automatically by the environment when the object is inserted in the Turtle Scene. These also vary according to the object's type. The most common of them are the: All Area, Side Area and Volume.

4.4 Using the MaLT Environment's features

4.4.1 Using the 1d Variation Tool

The 1dVT doesn't appear in the MaLT's interface by default. In order to make it visible and work with it, follow the steps mentioned below:

Step 1: Type at the Logo Editor a procedure of at least one variable.

to rect :a :b :c


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

Step 2: Run the procedure using an arithmetical value for each variable.

Type the procedure's name and inside the parentheses type the arithmetic values for each variable.

```
rect(20 30 40)
```

Press F6 to run the command. A rectangle corresponding to these specific values appears in the Turtle Scene.

Step 3: Click on the turtle's trace.

When clicking on the turtle's trace the 1dVT appears on the Logo Tab just below the Logo Editor.

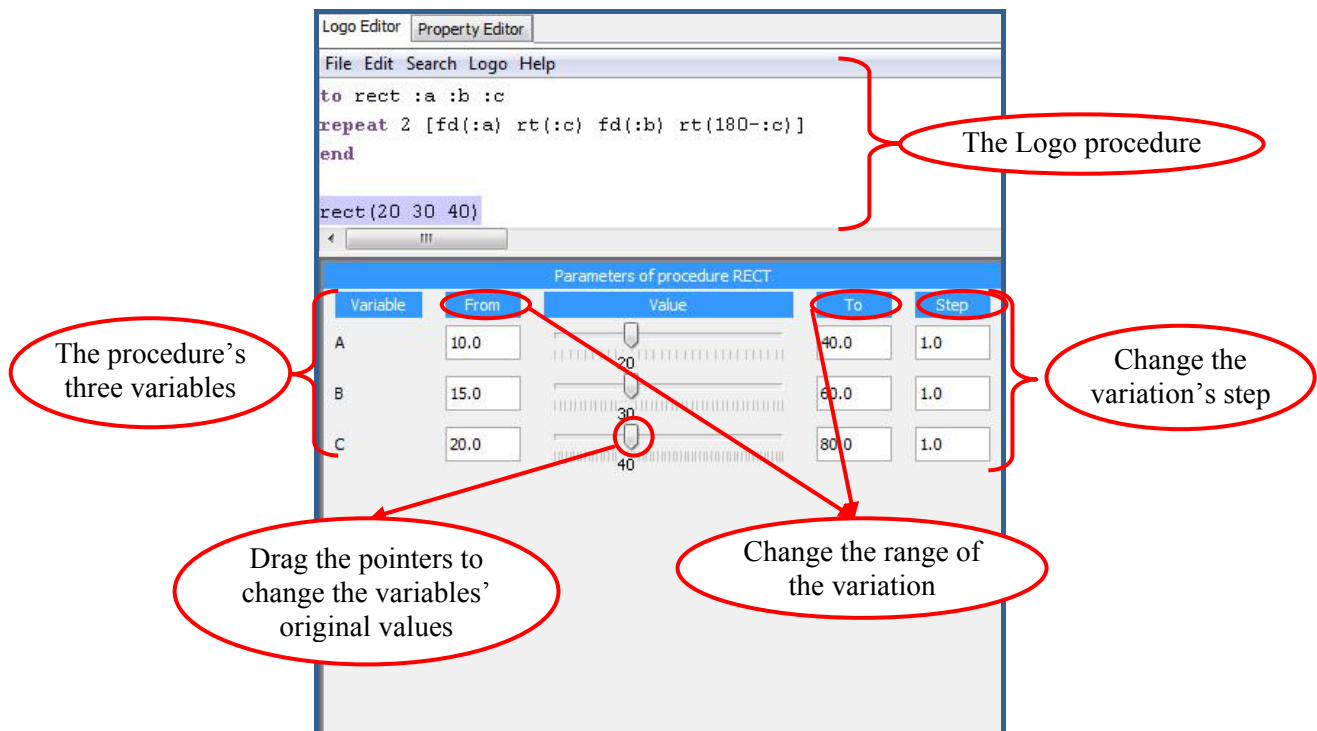


Figure 13: The 1d Variation Tool (1dVT).

The 1dVT consists of three sliders, each of which corresponds to one of the “rect” procedure's variables. Dragging one of the sliders causes the figure in the Scene to change, as the values of the variable change consequently and the logo procedure is executed again and again using each time a new arithmetic value for the variable.

To clear the Turtle Scene press F5.

4.4.2 Working with the 2d Variation Tool

Just like the 1dVT, the 2dVT doesn't appear in the MaLT's interface by default. In order to make it visible and work with it, follow the steps mentioned below:

Step 1: Type at the Logo Editor a procedure of at least two variables.

```
to rect :a :b :c
up(90)
repeat 2 [fd(:a) rt(:c) fd(:b) rt(180-:c)]
end
```

Step 2: Run the procedure using an arithmetical value for each variable.

```
rect(7 3 55)
```

Press F6 to run the command. A rectangle corresponding to these specific values appears in the Turtle Scene.

Step 3: Click on the turtle's trace.

When clicking on the turtle's trace the 1dVT appears on the Logo Tab, just below the Logo Editor, while the 2dVT appears in a separate window. However, the 2dVT is not yet activated.

Step 4: Select which variable will be represented in which axis.

The 2dVT is activated through the 1dVT. Right beside each variable's name in the 1dVT there is an icon displaying a red orthogonal bi-axial system. Choose the vertical or the horizontal axis on the icon so as to define in which of the 2dVT's axis the selected variable will be represented. When clicking on the axis you wish the variable to be represented, the colour of the selected axis turns to green.

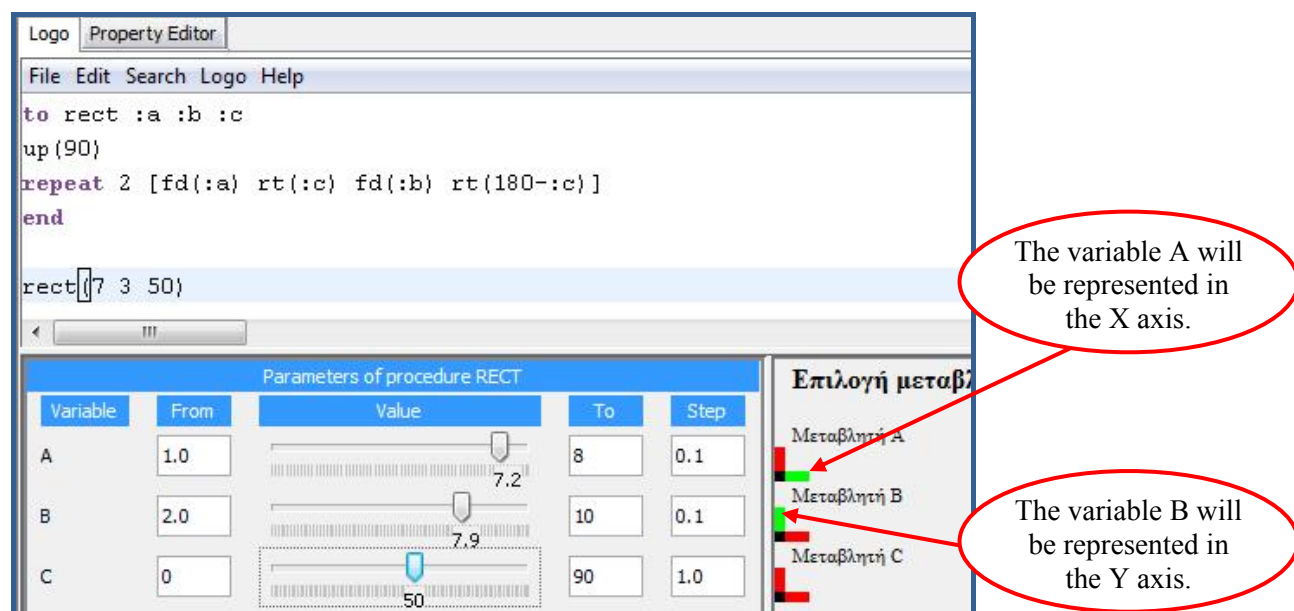


Figure 14: Defining which variable will be represented in which 2dVT axis.

Step 5: Drag the mouse on the pad and define a relationship between the variables

The mouse can be freely dragged on the 2dVT's pad, leaving a trace behind. Each position on the pad corresponds to a specific value for each of the selected variables. By dragging the mouse on the 2dVT a functional relationship between the two variables is defined. The changes in the mouse's position cause changes to the 1dVT sliders' values as well as to the figure in the Scene according to this functional relationship. The resulting line on the 2dVT's pad corresponds to the graph of this functional relationship.

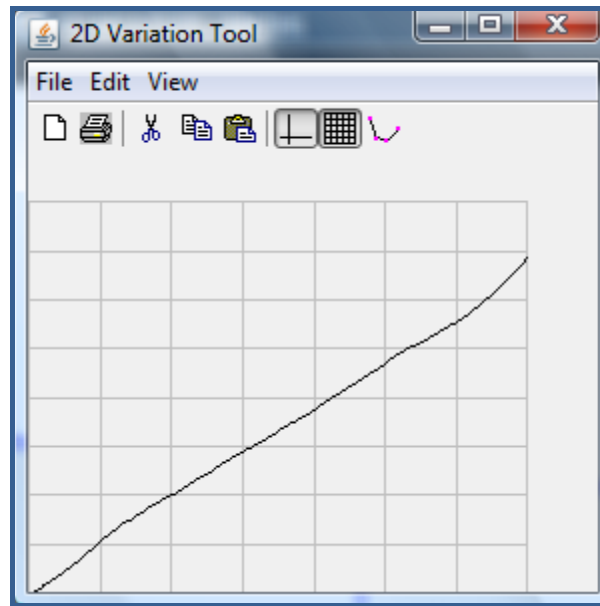


Figure 15: Creating the $y = x$ functional relationship between the two variables.

4.4.3 *Working with the Vector Variation Tool (VVT)*

Just like the 1dVT and the 2dVT, the VVT doesn't appear in the MaLT's interface by default. In order to make it visible and work with it, follow the steps mentioned below:

Step 1: Type at the Logo Editor a procedure of at least three variables and run the procedure. Click on the turtle's trace in the 3d Scene.

```
to rect :a :b :c
  up(90)
  repeat 2 [fd(:a) rt(:c) fd(:b) rt(180-:c)]
end
rect(7 3 55)
```

Step 2: Select which variable will represent r , θ and ϕ .

The VVT is activated when clicking on one of the icons displaying a polar coordinate system on the 1dVT slider. The pop-up menu that appears will help you define whether the selected variable will represent in the polar coordinate system the r (the length), the θ (the angle between the vector's projection on the xy plane and the x -axis) or the ϕ (the angle between the vector and the xy plane) coordinate.

Step 3: Manipulating the r , θ and ϕ values.

The VVT consists of two vector-like representations (the constituent projections) and a resultant vector representation. Using the first vector-like representation, you may dynamically manipulate the vector's length and rotate it to control the value of angle θ , while using the second one you may manipulate the vector's length and rotate it to control the value of angle ϕ . The changes performed on the two vector-like representations are reflected on the resultant vector representation, which is not available for direct manipulations. The polar and the Cartesian values are represented graphically by the vectors themselves and their projections and numerically in the text boxes just below the vectors. Any manipulations performed to the vector presentations are visualised in the Turtle Scene as the figure constructed changes dynamically.

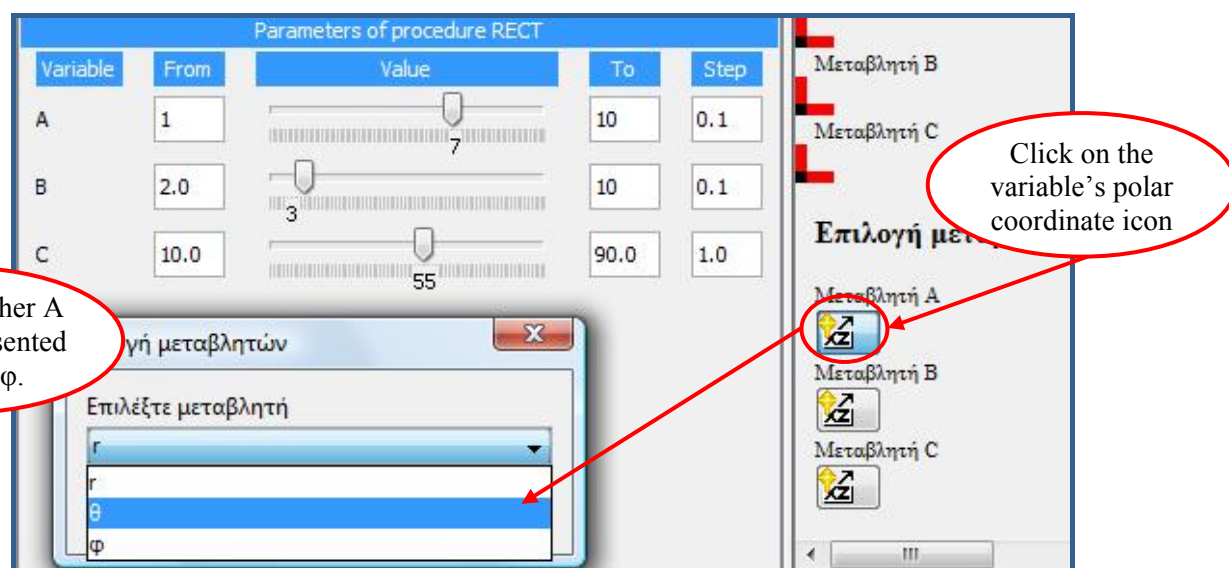


Figure 16: Opening up the Vector Variation Tool.

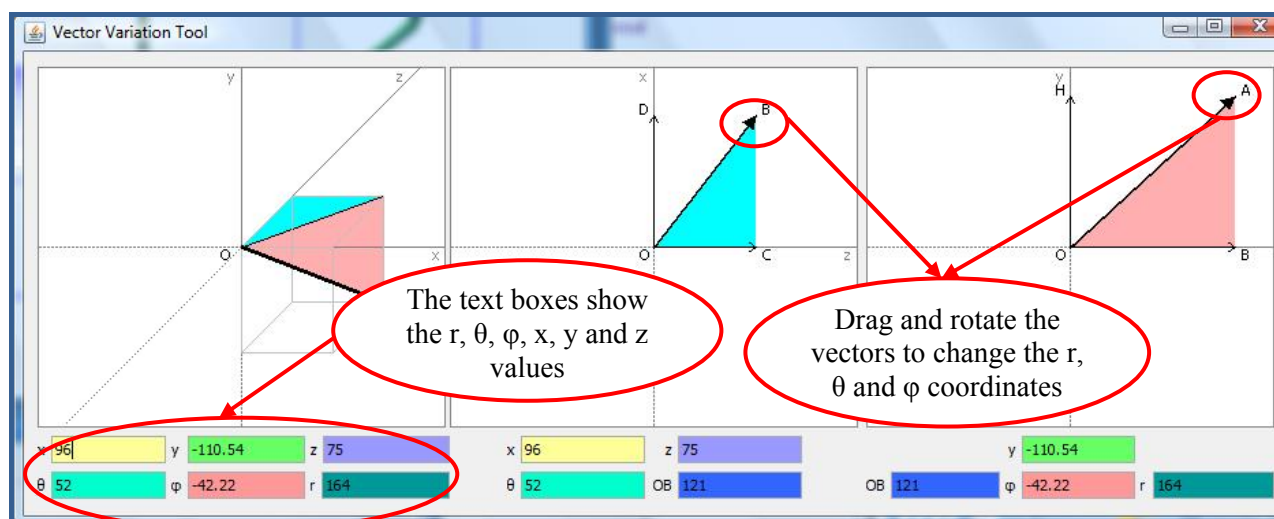


Figure 17: The Vector Variation Tool (VVT).

4.4.4 Inserting and manipulating ready-made stereometric objects

To insert ready made objects in the Turtle Scene, click on the Property Editor tab and follow the steps mentioned below.

Step 1: *Activate the Editor.*

Click on the New Object icon on the Editor's Toolbar to initialise the inserting procedure.

Step 2: *Select the type of object to insert.*

Choose from the Palette's pop-up menu the type of object you wish to insert in the Turtle Scene. Notice that the properties on the area below the Palette change according to the type of object you choose. Choose for example a cone

Step 3: *Define your object's properties.*

Define your object's Name, Colour, Transparency, Position (X , Y , Z), Radius and Height.

Step 4: *Insert your object in the Turtle Scene.*

Press the Insert Object icon on the Toolbar so as to make the cone appear in the Turtle Scene. After inserting the object, the MaLT environment calculates the object's All Area, Side Area and Volume and provides their values on the Editor's corresponding fields.

Step 5: *Manipulate your object.*

The little white spheres that appear on your object are its handles. By clicking and dragging a handle you may modify your object's dimensions. These changes become visible not only graphically in the Scene but also on the Editor's fields as the Radius and Height values also change.

To move your object in the XY plane click at any point on its surface and drag it towards the direction you wish to move it. By using both the left and the right mouse button you may also drag your object along the Z axis.

When clicking at any point of the Scene the object is deselected, its handles disappear and its surface appearance turns from grid-like to solid-like. To select it again, just click on it.

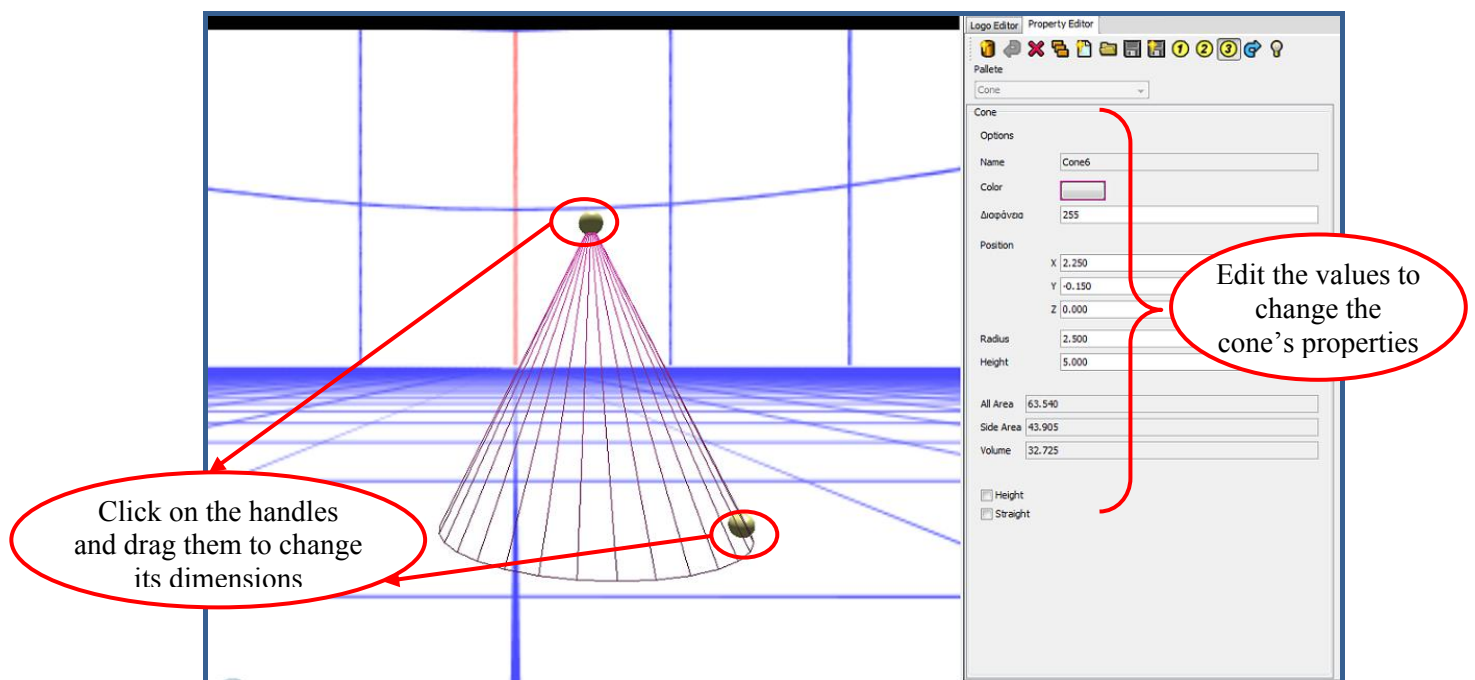


Figure 18: A cone inserted in the Turtle Scene.

The Property Editor's objects can also be inserted through the Logo Editor using logo commands, specialised for each type of object.

4.4.5 Manipulating the camera viewpoints

In the new MaLT environment the user has at his disposal three cameras. Each one offers a different viewpoint of the ready-made and logo-constructed geometrical objects in the environment's Turtle Scene. The selection of the viewpoint from which you may observe your constructions is performed through the Property's Editor Toolbar.

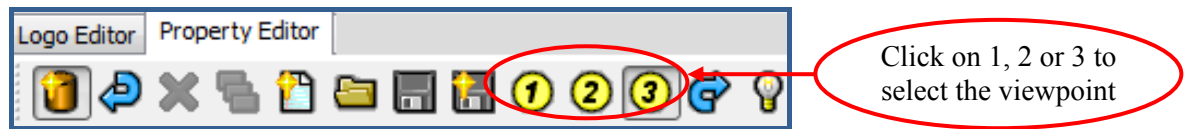


Figure 19: The third camera is currently activated in the Property Editor's Toolbar.

Camera 1 is placed on the top of the Scene and shows the Floor View of the constructions inside the Scene. Camera 2 is placed on the one of the Scene's sides and shows the Side View of the constructions inside the Scene. Camera 3 is the main camera of the Scene and is turned towards the negative direction of the Z axis. It is the only camera whose position the user may change, using logo commands or combinations of keyboard buttons.

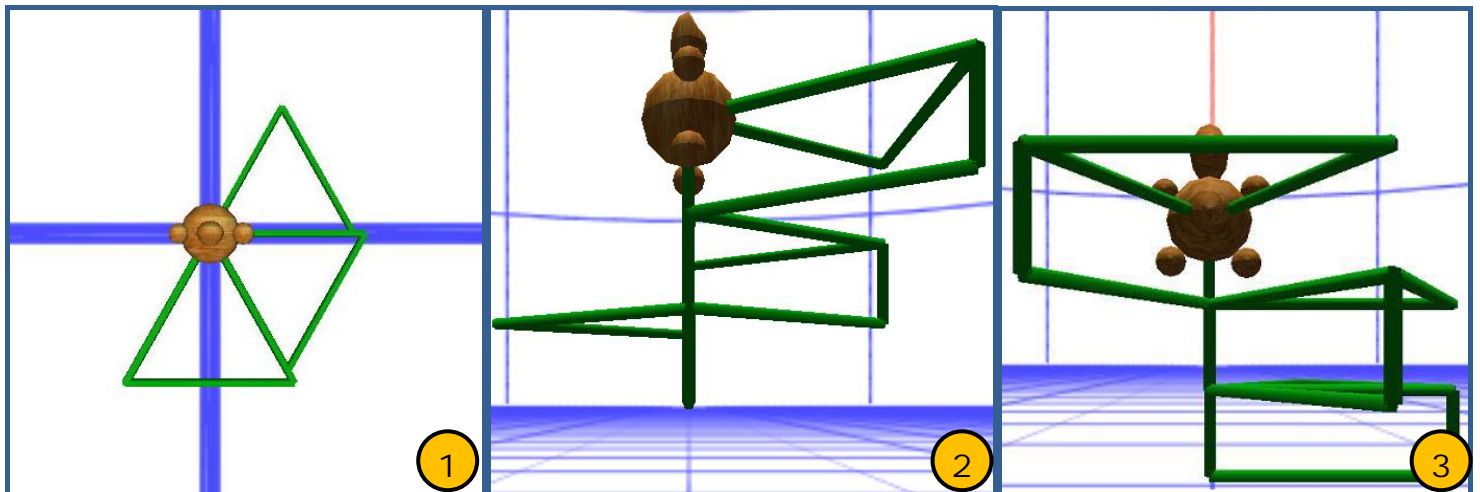


Figure 20: The Camera 1, Camera 2 and Camera 3 viewpoint.

4.4.6 *Creating logo-constructed 3d stereometric objects*

In the MaLT environment the user may also construct complex stereometric objects using simple logo commands and procedures. Follow the steps mentioned below to construct a dodecahedron.

Step 1: Define the "Pentagon" procedure to create a pentagon

The dodecahedron will consist of 12 facets each of which will have the shape of a pentagon. The following procedure describes the turtle moves to be performed so as to create a pentagon:

```
(line 1)  to pentagone :a
(line 2)  repeat 5 [
(line 3)  fd(:a) rt(72)]
(line 4)  end

(line 5)  pentagone(3)
```

The pentagon will be constructed as the turtle will repeat 5 times (line 2) the procedure: fd(:a) rt(72). The turtle will move forward maintaining its direction by a number of pixels that will be defined by the (:a) variable. As the pen is down, the turtle will draw a line whose length will be (:a) pixels. After moving for (:a) pixels, the turtle will turn clockwise by 72 degrees. As the fd(:a) rt(72) repeats 5 times a pentagon whose sides is (:a) pixels long will appear.

Try to run the procedure attributing an arithmetical value to the (:a) variable (e.g. 3, line 5) and observe the graphical result in the Turtle Scene.

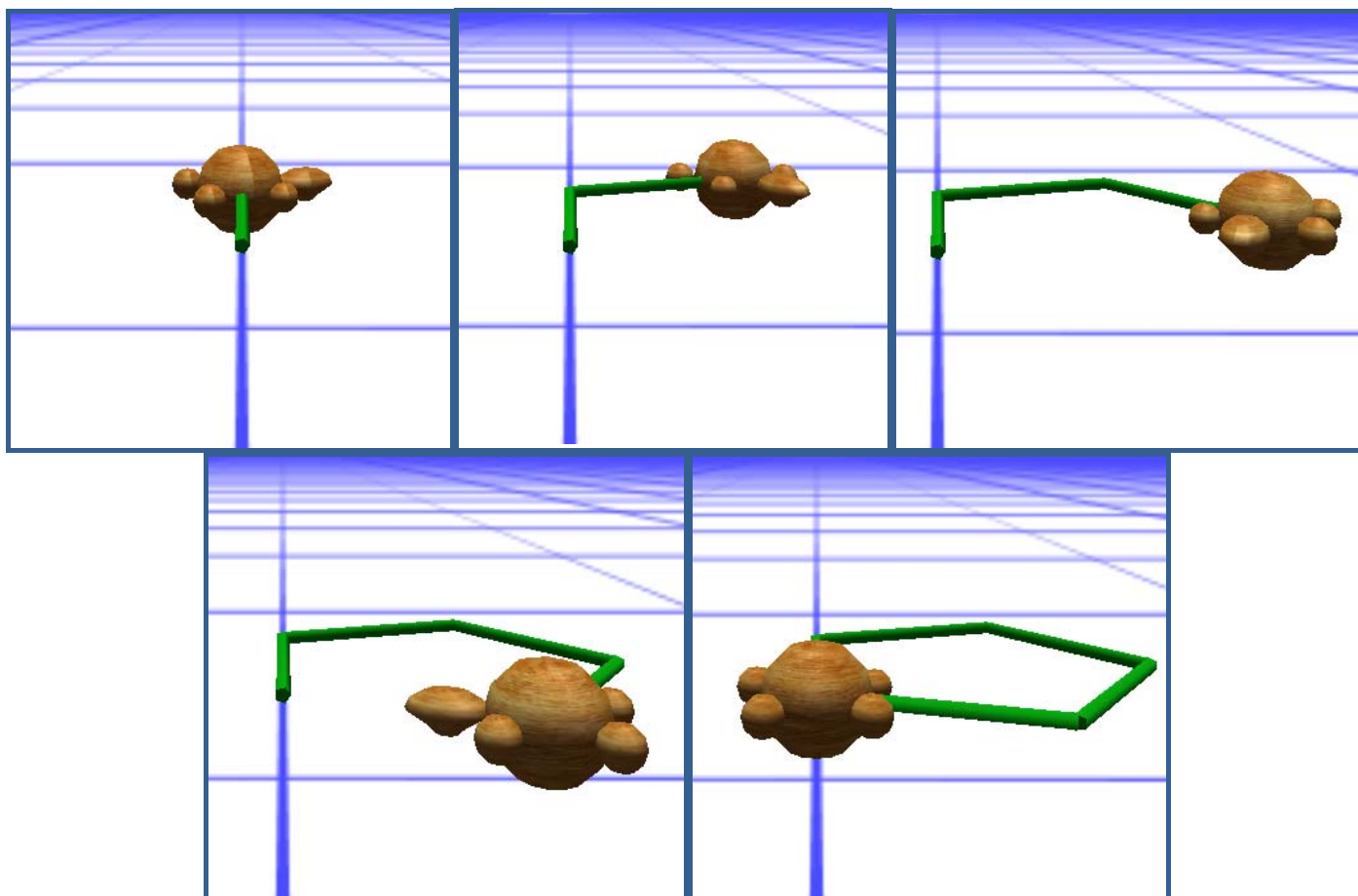


Figure 21: Running 5 times the fd(:a) rt(72) procedure (a = 3).

Step 2: Define the “cup” procedure to create six adjacent pentagons

For the construction of the bottom half of the dodecahedron six pentagons will be needed. The next procedure will help you create 6 adjacent pentagons forming a kind of “cup”. The dihedral angle between two adjacent pentagons for the formation of a dodecahedron equals to $2 \cdot \arccos(0.5/\sin(72))$.

```
(line 6)  to cup :a
(line 7)  repeat 5 [
(line 8)  rr( $2 \cdot \arccos(0.5/\sin(72))$ )
(line 9)  pentagone(:a)
(line 10) lr( $2 \cdot \arccos(0.5/\sin(72))$ )
(line 11) fd(:a) rt(72)]
(line 12) end
```

```
(line 13) cup(3)
```

The bottom part of the dodecahedron will be constructed as the turtle will repeat 5 times (line 7) the procedure:

```
(line 8)  rr( $2 \cdot \arccos(0.5/\sin(72))$ )
(line 9)  pentagone(:a)
(line 10) lr( $2 \cdot \arccos(0.5/\sin(72))$ )
```

```
(line 11)  fd(:a) rt(72)
```

Initially, the turtle will rotate to the right by $2 \cdot \arccos(0.5/\sin(72))$ degrees (i.e the degrees of the dodecahedron's dihedral angle) (line 8). The pentagon procedure will be executed and the first pentagon will appear. Its side length will be defined by the (:a) variable (line 9). The turtle will then return to the position it held before the execution of the pentagon procedure.

The turtle will rotate to the left by $2 \cdot \arccos(0.5/\sin(72))$ degrees (i.e the degrees of the dodecahedron's dihedral angle) (line 10). It will then move forward by a number of pixels that will be defined by the (:a) variable and turn clockwise by 72 degrees.

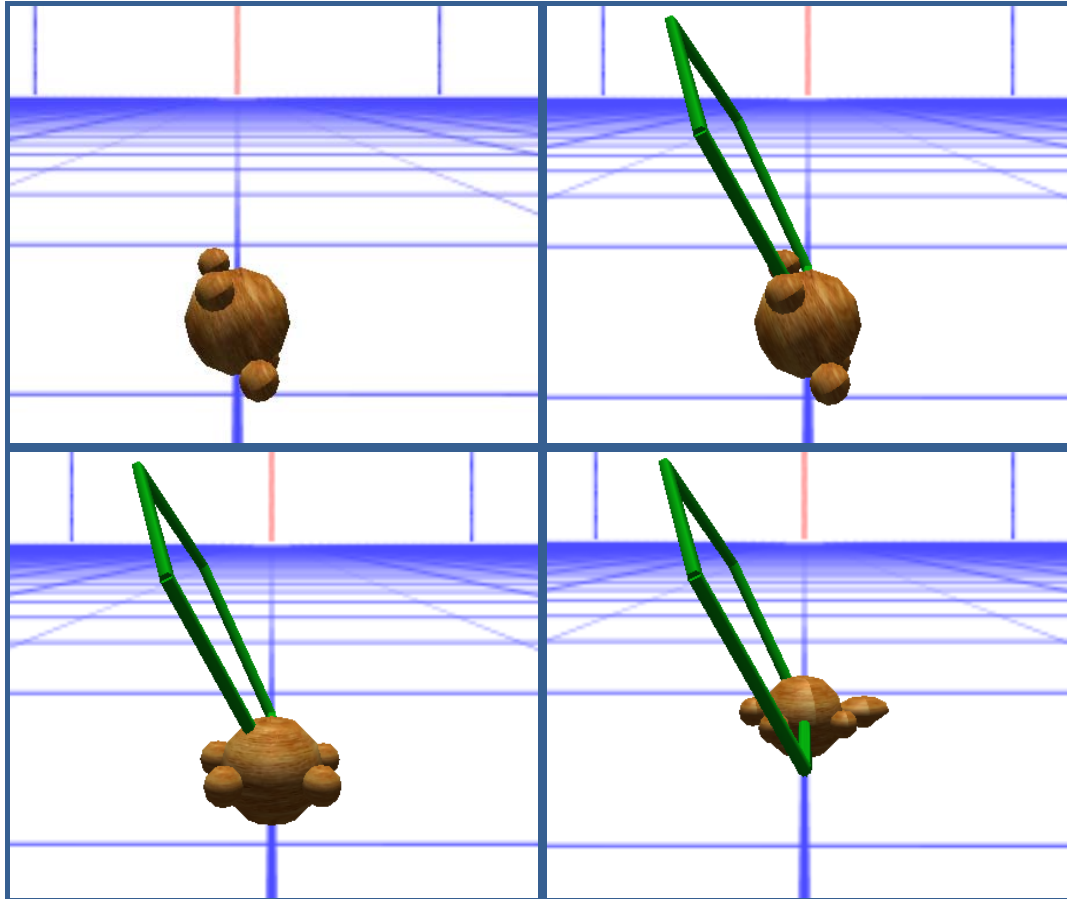


Figure 22: Running lines 8 – 11.

Try to run the procedure attributing an arithmetical value to the (:a) variable (e.g. 3, line 8) and observe the graphical result in the Turtle Scene. Notice that the sixth pentagon (the one on the base of the figure) is not logo-constructed but shaped by the other five ones.

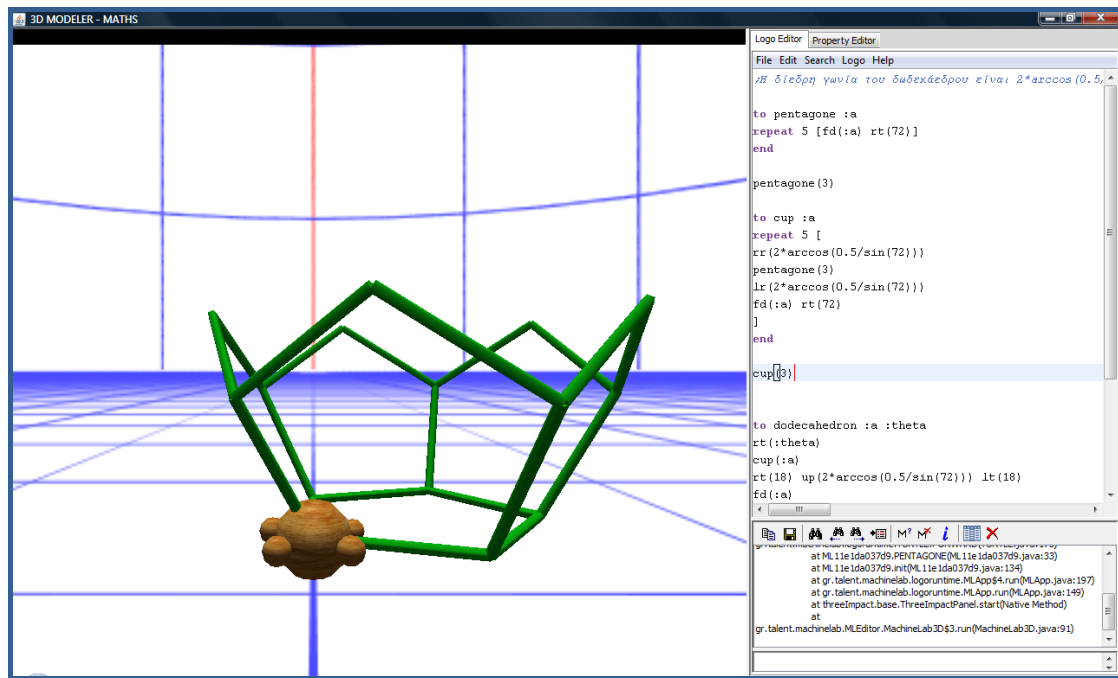


Figure 23: Running the “cup” procedure ($a = 3$).

Step 3: Define the “dodecahedron” procedure to create the figure

For the construction of the dodecahedron you will need two “cup” procedures. The first one will create bottom half of the dodecahedron and the second one the upper half. The upper half will be a reversed “cup”.

```
(line 14)  to dodecahedron :a :theta
(line 15)  rt(:theta)
(line 16)  cup(:a)
(line 17)  rt(18) up(2*arccos(0.5/sin(72))) lt(18)
(line 18)  fd(:a)
(line 19)  lt(18) dp(180-2*arccos(0.5/sin(72))) lt(18)
(line 20)  rt(108) fd(:a) lt(72) fd(:a)
(line 21)  rt(18) dp(180-2*arccos(0.5/sin(72))) rt(18) rr(180)
(line 22)  cup(:a)
(line 23)  end
(line 24)
(line 25)  dodecahedron(3 0)
```

The point at which the construction of the dodecahedron will begin is defined in line 10. The turtle will turn right clockwise by a number of degrees that will be defined by the (:theta) variable. At that point the construction of a “cup” will begin. However, regardless the value of the (:theta) variable, the turtle will finish the construction of the cup at the bottom of the figure. To move upwards and find the right node so as to start the construction of the upper cup, the turtle will need to move according to lines 17–21. To avoid creating unnecessary lines, it will have to move on the bottom cup’s existing lines.

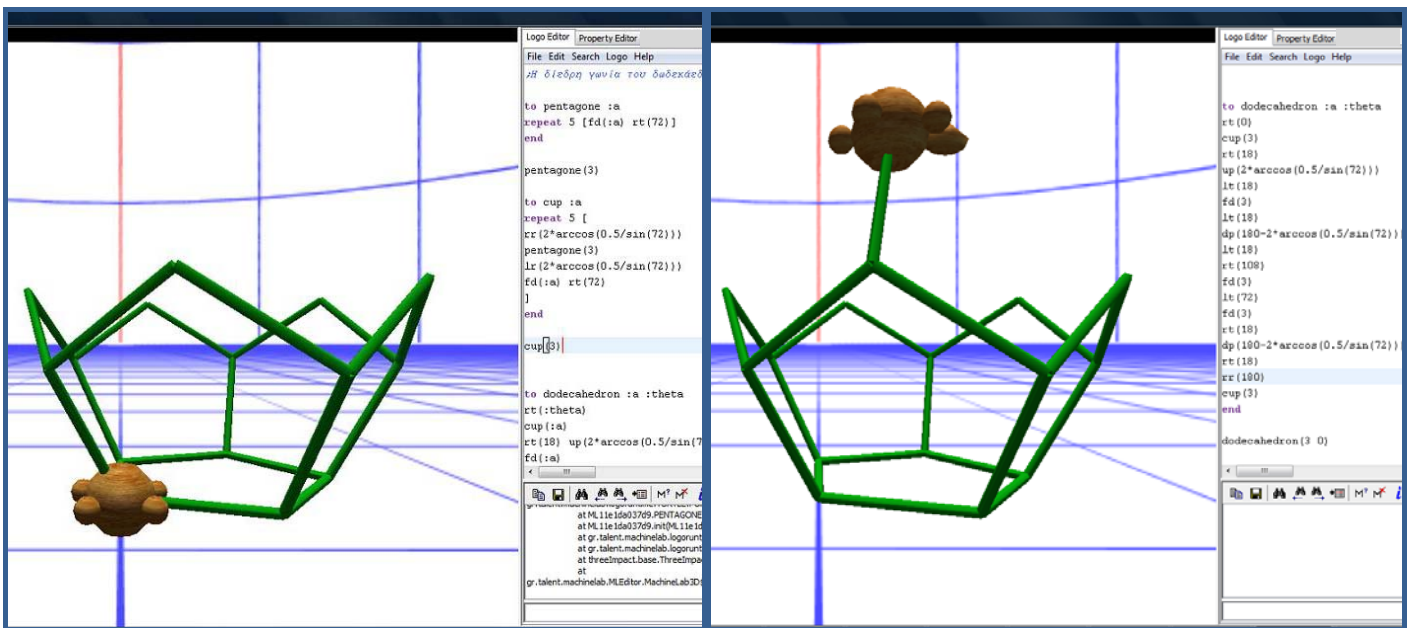


Figure 24: Running lines 17 – 21 ($\theta = 0$, $a = 3$).

Try to run the procedure attributing an arithmetical value to the (:a) and (:theta) variable and observe the graphical result in the Turtle Scene. Notice that the sixth pentagon on the upper base of the figure) is not logo-constructed but shaped by sides of the other ones.

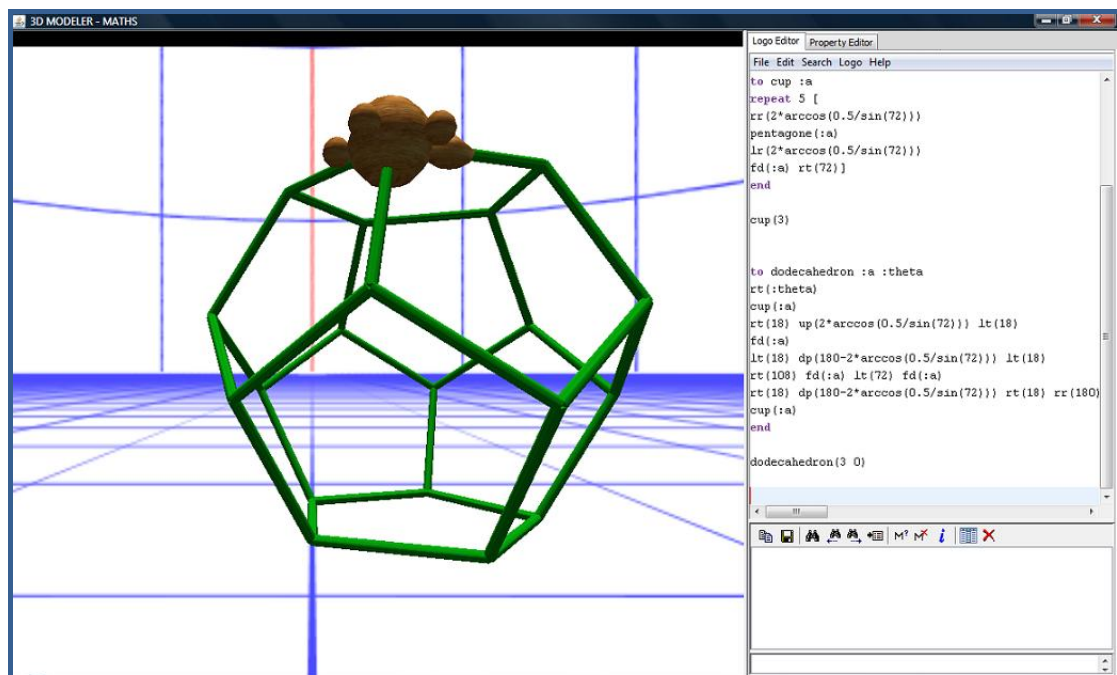


Figure 25: Running the “dodecahedron” procedure ($\theta = 0$, $a = 3$).

5 Perspectives

5.1 In terms of developments

The idea was to employ new technologies emerging for 3d animations to develop a mathematical microworlds environment integrating the two most prominent

technologies for engagement with geometry, turtle geometry based programming and dynamic manipulation. Both of these perspectives are part of the theoretical perspective developed at ETL to engage in design research involving interventions in classrooms aiming to immerse students in constructionist mathematics including the use of formalism to generate mathematical meanings. The main problems met in the process of developing the software were due to the need to connect 3d engines to java based programming and develop a logo language with java to construct and manipulate models. The best available 3d engine originally was “3impact” which we had identified before the beginning of the project and ported a Delphi programming language to build custom 3d models. In ReMath we thus needed to change the language to java so as to have multiple windows and dynamic manipulation. The manipulation however was very slow due to fragmentation in the design of the different compartments of the software. We thus identified a recently available new 3d engine and ported the java–logo components on to that with highly improved results. We also created a better GUI to make the software more stable. The prospect is to use this in the future as a base for a larger variety of programmable mathematical and scientific models and expand to include games, hopefully playable by many on the web.

5.2 In terms of deployment

In the time of the project we used MaLT as one of the main pieces of software taught in two regular master’s courses (titled Methodology and Didactics of Mathematics and Didactics of Special Subjects with new Technologies respectively). We also used MaLT in a large–scale initiative of the Ministry of Education targeting 4000 mathematics teachers to receive a 96 hour course in using digital media in the teaching of mathematics. We were part of a project within this initiative to train 100 mathematics teacher educators and carried out a 350 hour course to 20 of them and took part in the teaching of all the others. MaLT featured as the 3d version of Turtleworlds an ETL designed widely available software. We made MaLT available on the ETL website and advertised it as a resource in these courses.

DDA 5: Cruislet

Authors: Tryfona, Tsironis, Markopoulos

Talent SA, Educational Technology Lab

Table of content

1)Introduction	2
2)Last evolutions of the Cruislet DDA	3
3)History of the design and development of the DDA	7
4)Description of the final form of the DDA	8
1. <i>Introduction</i>	8
2. Terrain scene.....	8
3. Avatar tab.....	9
3.1 Avatar.....	9
3.1.1 Creation.....	9
3.1.2 Deletion.....	9
3.1.3 Avatar properties.....	9
3.1.4 Vectors	9
3.2 Navigation in space.....	10
3.2.1 Systems of reference	10
3.2.2 Select a destination	11
3.2.3 Transfer commands in Logo Tab	11
3.3 Camera	12
4. Contents tab	12
5. Logo tab	12
5.1 Logo commands.....	12
6. Menu	13
5)Perspectives	14
o In terms of developments.....	14
o In terms of deployment	14

Introduction

The 'Cruislet' environment is a state-of-the-art dynamic digital artefact that has been designed and developed within the ReMath project. It is actually 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. In particular, the experimentation with the Cruislet environment focuses on the study of the development of student's conceptions concerning the mathematically driven navigations in virtual 3-d geographical spaces.

The 'Cruislet' environment is a digital medium based on GIS (Geographic Information Systems) technology that incorporates a Logo programming language. It is designed for mathematically driven navigations in virtual 3d geographical spaces and is comprised of two interdependent representational systems for defining a displacement in 3d space, a spherical coordinate and a geographical coordinate system. We consider that the new representations enabled by digital media such as Cruislet can place mathematical concepts in a central role for both controlling and measuring the behaviours of objects and entities in virtual 3d environments. The objects (avatars) are airplanes and their displacement is represented by a vector. The user is able to navigate airplanes either by using the Avatar tab or through the use of a Logo editor Tab (see Figure 1).

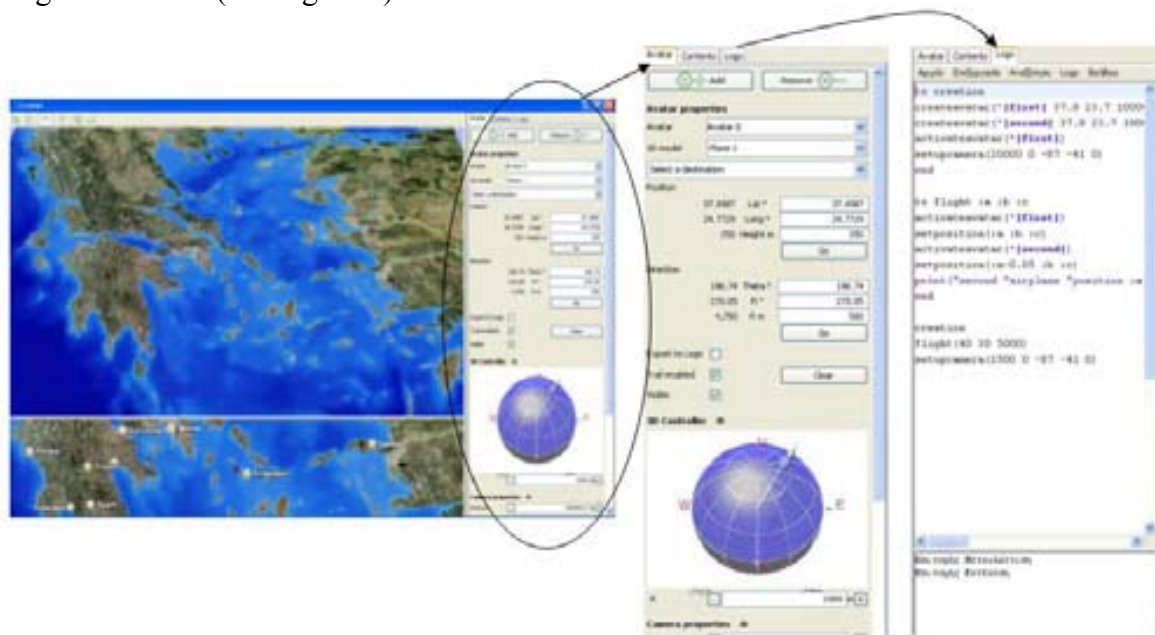


Figure 1. Cruislet environment – Avatar Tab – Logo Tab

In the Avatar Tab, the user can create airplanes and define their position in either of the following ways:

- a) by determining the latitude, the longitude and the height in which the airplane will be placed or
- b) by determining the vector of displacement, i.e. the angles θ and φ and the length of the vector of displacement.

In the Logo Tab the user can actually edit and run Logo programs and thus create multiple or relative displacement rules in 3d space. The Logo programmability is considered necessary as it provides users with the option to actually anticipate the

result of their action and engage in expression of mathematical ideas through meaningful formalism by means of programming. In this sense, Cruislet can be conceived as a constructionist medium as the user can construct flights and build dependency between flights.

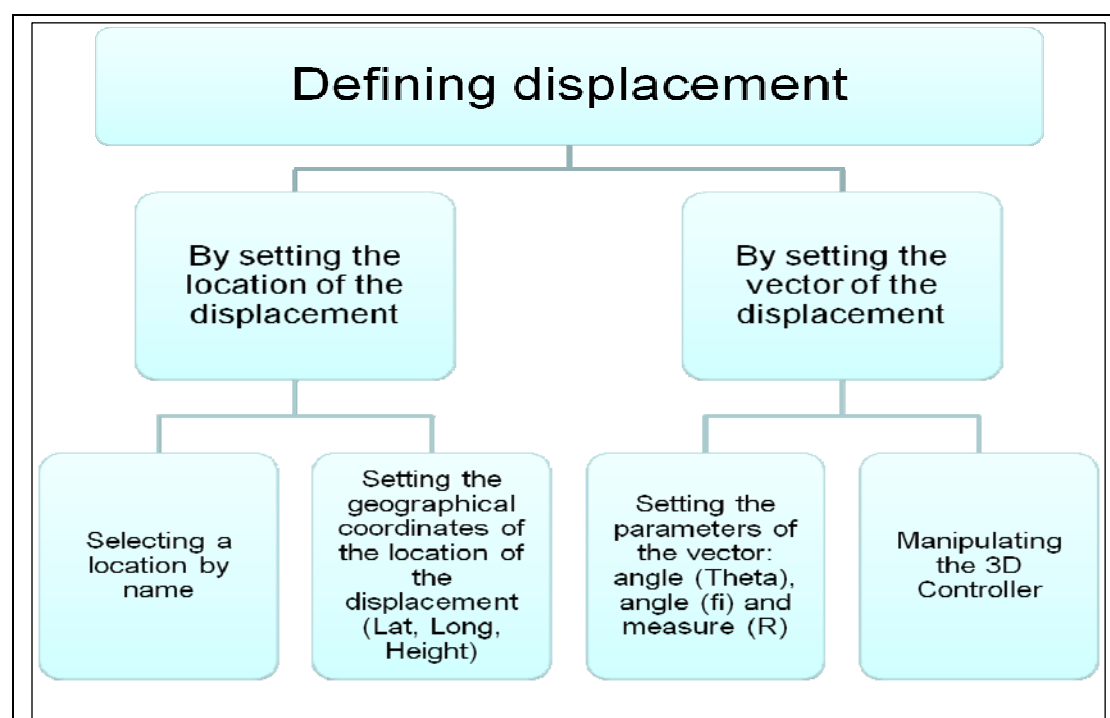
Our initial aim when designing Cruislet was to create a realistic context such as a 3d geographical map where geo-coded data will be integrated with mathematic representations, in which students would be able to explore, experiment and construct virtual reality microworlds. Students' constructions in these microworlds are airplanes' trips as well as rules of airplanes' displacements that are defined in Logo procedures. In that sense, navigations in virtual 3d geographical spaces within Cruislet could be conceived as game play simulations, as users manipulate objects in a 3d game-like environment according to rules built on this which are defined by the designer of each microworld.

1) Last evolutions of the Cruislet DDA

The last evolutions of Cruislet provide a dynamic visual medium that support immediate visualization of *multiple linked representations*. The evolutions of the DDA involve the development of a number of functionalities in the Avatar tab as well as the development of new and clearer Logo programming primitives.

In particular, a displacement of an avatar is defined by employing either a geographical (lat-long-height) coordinate system by setting the displacement location or a spherical (θ, ϕ, r) coordinate system by setting the direction and the length of the vector of displacement. So, Cruislet microworld has been re-designed so that its interface for avatar navigation comprises of two options: geographical coordinates and spherical coordinates. Also a 3D controller representation of spherical coordinates was added as an extra option.

The diagram below shows the provided options for the definition of the displacement of the avatar.



Moreover, emphasis has been given to the interdependent characteristics of these options. The members of ETL team were influenced by the constructionist perspective and focused on the on the provided representations. On the other hand, developers considered these representations as different, independent options for the user.

The interaction between the developers and the ETL team resulted in the development of the possible links between the different representations systems and between the different functionalities. In particular, the different ways of defining the displacement are interrelated. The variation of the end-position of the vector resulted the automatic variation of the parameters of the vector (ϕ , θ , R) and vice versa. Moreover, the dynamic manipulation of the 3D controller causes the direct variation of the end-position as well as the parameters of the vector. In addition, the direct displacement in avatar Tab is linked with the Logo programming language as the possibility to export the direct manipulation to the Logo editor has been developed. Finally, we should mention the linking between the provided geo-coded information with the functionality of the displacement of the avatar in the direct manipulation view as well as in the Logo editor view.

Consequently, the last version of the DDA involves the following evolutions:

I. Spherical coordinates

The displacement of the avatar according to its direction is defined by a vector in terms of its length (R), an angle between the vectors projection onto the xy plane and the y -axis (Theta) (figure 2, schema i) and the angle between the vector and the x -axis (Fi) (figure 2, schema ii).

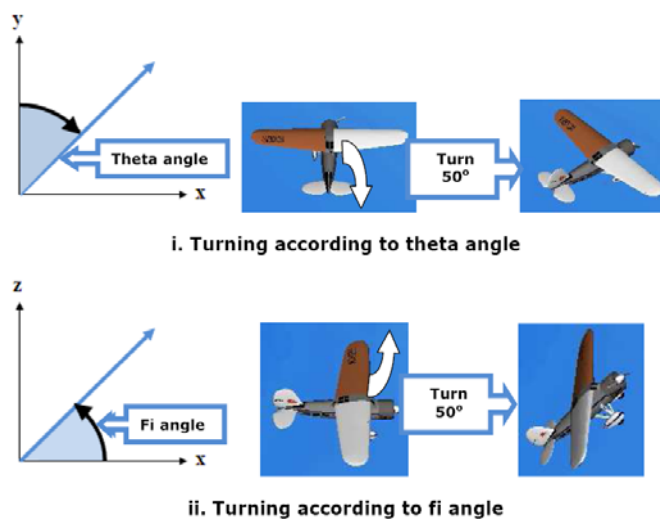


Figure 2: Spherical coordinates

II. Linkage between spherical and geographical coordinates' values so there is an interdependent relationship between them

While changing avatar's relative position by using one frame of reference, the other one changes its values correspondingly to avatar's displacement. If for example the user chooses spherical coordinates to displace the avatar, geographical coordinates will change according to the position that the avatar will be placed.

III. Select a destination

The user can move the avatar to an absolute position by selecting the destination of the avatar. This can be done by using the selector. The user can choose one of the predefined cities that appear. By clicking on a destination (city), the corresponding coordinate values appear on 'Position' and 'Direction' (figure 3). On 'Position' the user can see the geographical coordinates of the selected city, while on 'Direction' the user can see the angles the avatar must turn and the distance the avatar must cover in order to get to the destination.

Figure 3: Linkage between the selection of a destination and spherical - geographical coordinates' values

IV. 3D controller representation of spherical coordinates

The direction of the avatar can also be controlled by a build-in 3D controller. The controller is a semi transparent sphere (figure 4). Using the controller the user actually defines the direction of the displacement of the agent. The centre of the sphere is the starting point of the vector and the user can actually drag the arrow of the vector and by moving around the mouse can vary each of the 2 variables (theta, fi) that define the vector.

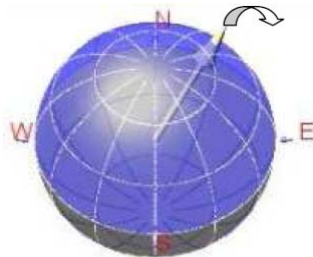


Figure 4: 3D controller

The length of the vector, can be defined by the user either by editing it or changing its value using the buttons with the '+' and '-' symbols. Pushing these buttons causes the increment or decrement of the vector's length by 1000 meters (figure 5).

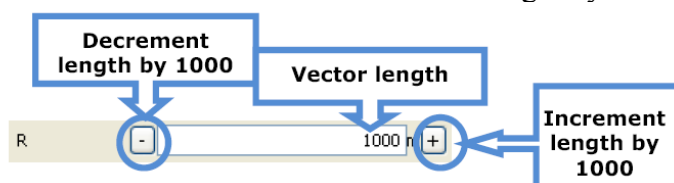


Figure 5: 3D controller's vector length

V. Linkage between spherical - geographical coordinates' values and 3D controller representation

While the user uses the 3D controller the values of geographical and spherical coordinates change correspondingly

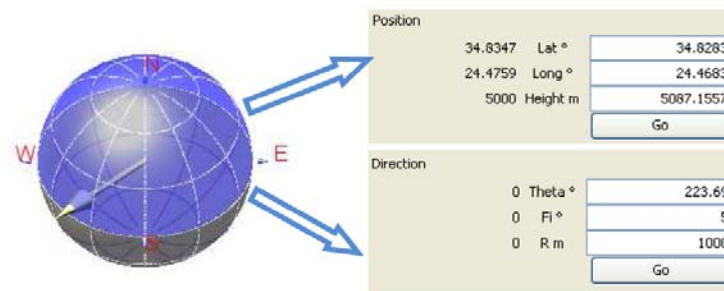


Figure 6: Link between 3d controller and and spherical - geographical coordinates' values

VI. Addition of the 'Export to Logo' button

The user can set the position or the direction of the avatar and transfer the corresponding commands in Logo Tab. (figure 7). The user is able to run the commands by using the Logo Tab and move the avatar in terrain scene.

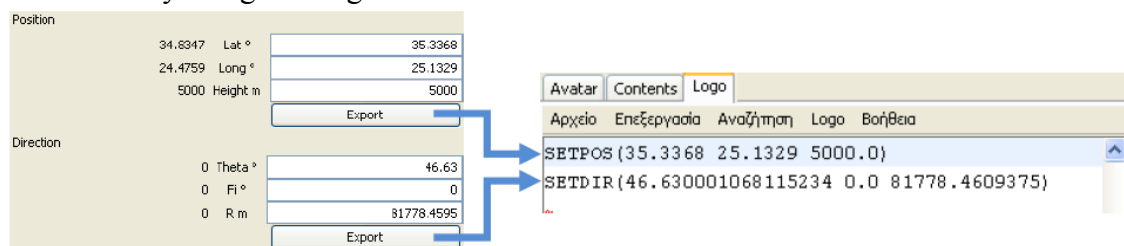


Figure 7: Transfer Commands into Logo tab

VII. Camera properties

- The user is able to define if camera properties will be visible or not
- Change camera distance by pressing the corresponding buttons ('+' and '-')

VIII. Multiplication of vectors

The multiplication of decimal numbers with the selected vector have been developed.

IX. Font size can be changed

X. Logo commands

New Logo programming primitives including the navigation of the avatar by employing the spherical coordinate (θ, ϕ, r) system have been developed. In particular, the following primitives could be used:

- SETDIRECTION(θ ϕ r) - shortcut {SETDIR}
- OUTPUTPOSITION - shortcut {OPPOS}
- OUTPUTDIRECTION - shortcut {OPDIR}
- 'include' command

Moreover the execution of all Logo commands has been debugged.

XI. MathDills

The ability to exchange files within MathDills environment has been developed. In particular, the user can upload and download files using MathDiLS digital library.

2) History of the design and development of the DDA

The main phases of the design and development of the Cruislet DDA were:

- Collaboration between ETL researchers and Talent developers in order to determine the software functional requirements posed by the theoretical framework of WP1.
- The design of the specifications of the domain and the main part of Cruislet.
- The 1st version of Cruislet
 - Avatar's relative position is defined by its geographical coordinates and its height. Lat and Long inputs are actually the conversion of the geographical coordinates to its decimal form. The Height of the avatar is measured in meters.
- The beta version of Cruislet
 - vectors as displacements in geographical and spherical coordinate systems
 - navigating objects in 3d maps with geo-coded information
- The last version of Cruislet
 - Emphasis has been given to the development of the possible links between the different representations systems and between the different functionalities.

3) Description of the final form of the DDA

1. Introduction

Cruislet environment is a digital medium based on GIS (Geographic Information Systems) technology that incorporates a Logo programming language. It is designed for mathematically driven navigations in virtual 3d geographical spaces and is comprised of two interdependent representational systems for defining a displacement in 3d space, a polar coordinate and a geographical coordinate system.

The user is able to explore spatial visualization, geographical and mathematical concepts by controlling and measuring the behaviours of objects. The objects are airplanes and their displacement is represented by a vector. The user is able to navigate the airplanes either by using the predefined functionalities of the environment or through the use of Logo programming language.

The environment provides dynamic visual means that support immediate visualization of multiple linked representations, meaning that any action carried on a specific representation provides immediate change and feedback in all representations. In this way, mathematics are integrated with geo-spatial representations and information, providing opportunities for processes of mathematisation of geographical space.

2. Terrain scene

The terrain scene is splitted into two different windows. These windows are the two map viewers. The above window contains the 3d representation mode of the geographic map of Greece while on the below window the 2d representation mode of the map is displayed respectively (figure 1).

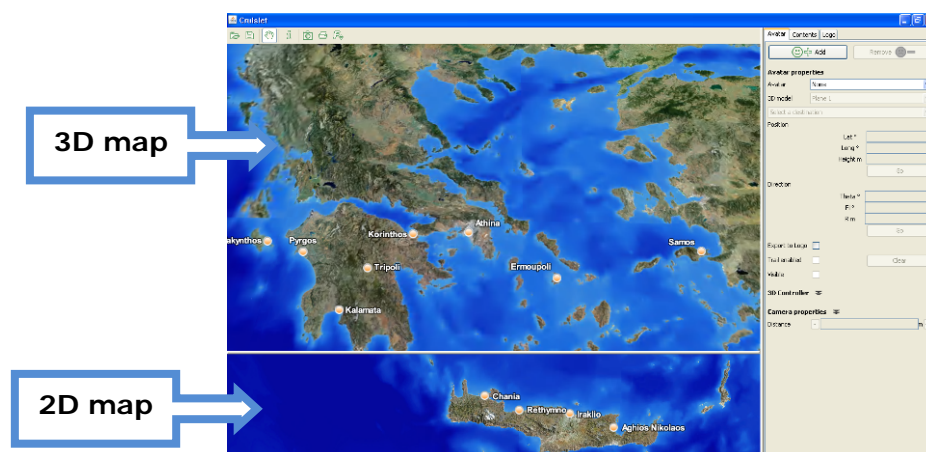


Figure 1. Cruislet map viewers

The two map viewers take up almost the 2/3 of the screen. The remaining 1/3 of the screen area is called the Cruislet area and contains the tabs selection. There are three tabs:

1. Avatar tab

2. Contents Tab

3. Logo Tab

3. Avatar tab

Avatar's tab functionalities can be divided in three categories according to the object manipulated or the way it is manipulated by the user:

1. Avatar properties
2. Navigation: moving the avatar by using either geographical or spherical coordinates
3. Camera properties

3.1 Avatar

Avatar properties and manipulation consists of:

3.1.1 Creation

By clicking on the 'Add' button a new avatar is created on the terrain area. The avatar is placed by default at the point the user last clicked with the mouse and in 5000 height.

3.1.2 Deletion

By clicking on the 'Remove' button the avatar is deleted. If more than one avatar is placed on the map, then the active avatar is deleted. If the user wants to delete a specific avatar, he must choose among the avatars and then press the 'Remove' button.

3.1.3 Avatar properties

The user, through the use of the administrator tab can vary the properties of the active avatar, such as its name, model, etc.

3.1.4 Vectors

The Avatar leaves a trace as it moves. The trace is a single, selectable, three-dimensional object (a thin cylinder ending on an arrow). It is actually a 3d vector representation.

By clicking with the right mouse button on a selected vector, a menu pops up.

⇒ Multiply

By selecting 'Multiply' a window pops up where 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 either from the beginning of the selected vector or from the point of the map where the right-click was occurred. The new vector has the same direction with the selected one, but its measure is equal to the length of the selected vector multiplied by the number defined by the user.

⇒ Divide

By selecting 'Divide' a window pops up where the user can define a number with which the selected vector will be divided. The result is a new vector which is displayed on the map. The new vector begins either from the beginning of the selected vector or from the point of the map where the right-click was occurred, as in the case of multiplication.

The new vector has the same direction with the selected one, but its measure is equal to the length of the selected vector divided by the number defined by the user.

⇒ Delete all selected

If the user wants to delete more than one vector, he is able to select them by pressing the Ctrl key.

⇒ Add all selected

This option appears when the user has selected more than one vector. The addition is applied in all of the selected vectors. The addition produces a new vector which begins either from the beginning of the first selected vector or from the point where the user made the right-click.

3.2 Navigation in space

Navigation in 3D geographical space can be done by using one of the following choices:

3.2.1 Systems of reference

There are two different systems of reference that define avatar's displacement: geographical coordinates and spherical coordinates.

Geographical Coordinates: Every spot of the 3D map corresponds to specific geographical coordinates (latitude and longitude) and height. Therefore, the position of each avatar is specified by three coordinates (lat, long, height). Lat and Long inputs are actually the conversion of the geographical coordinates to its decimal form and the Height of the avatar is measured in meters.

Avatar's displacement in 3D space can be done by editing the values of the three coordinates and pressing the 'Go' button. After pressing the button, avatar's displacement appears on the terrain scene and on the left column of 'Position' the current position of avatar appears. Every time the user changes avatar's position in the right column of 'Position', on the left column there are available the previous avatar's position. If the user wants to keep the same values in coordinates, he is able to transfer one or more of them by clicking on the middle column.

Spherical coordinates: The displacement of the avatar according to its direction is defined by a vector in terms of its length (R), an angle between the vectors projection onto the xy plane and the y-axis (Theta) and the angle between the vector and the x-axis (Fi).

The user is able to displace the avatar by pressing the 'Go' button. After pressing the button, avatar's displacement appears on the terrain scene and on the left column of 'Direction' the current direction of avatar appears. Every time the user changes avatar's relative position in the right column of 'Direction', on the left column there are available the previous avatar's relative position. If the user wants to keep the same

values in coordinates, he is able to transfer one or more of them by clicking on the middle column.

While changing avatar's relative position by using one frame of reference, the other one changes its values correspondingly to avatar's displacement. If for example the user chooses spherical coordinates to displace the avatar, geographical coordinates will change according to the position that the avatar will be placed.

The direction of the avatar can also be controlled by a build-in 3D controller. The controller is a semi transparent sphere (figure 2). Using the controller the user actually defines the direction of the displacement of the agent. The centre of the sphere is the starting point of the vector and the user can actually drag the arrow of the vector and by moving around the mouse can vary each of the 2 variables (theta, fi) that define the vector.

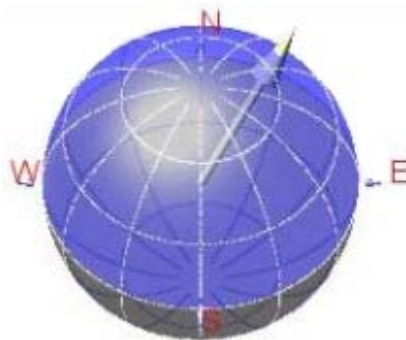



Figure 2. 3D controller

The length of the vector, can be defined by the user either by editing it or changing its value using the buttons with the '+' and '-' symbols. Pushing these buttons causes the increment or decrement of the vector's length by 1000 meters.

While the user uses the 3D controller the values of geographical and spherical coordinates change correspondingly.

The user watches at the controller at the same view angle that watches the active agent. Consequently, by changing the position of the camera, the view of the 3D controller changes correspondingly.

3.2.2 Select a destination

The user can move the avatar to an absolute position by selecting the destination of the avatar. This can be done by using  selector.

By clicking on a destination (city), the corresponding coordinate values appear on 'Position' and 'Direction'. On 'Position' the user can see the geographical coordinates of the selected city, while on 'Direction' the user can see the angles the avatar must turn and the distance the avatar must cover to get to the destination.

3.2.3 Transfer commands in Logo Tab

The user can set the position or the direction of the avatar and transfer the corresponding commands in Logo Tab. This can be done by checking the 'Export to Logo' box and pressing the 'Export' button that has replaced the 'Go' button. The

user is able to run the commands by using the Logo Tab and move the avatar in terrain scene.

3.3 Camera

Each avatar has a camera adjusted at initial distance 1000 meters off it. The camera can be moved around the avatar forming an imaginary sphere. The avatar is actually on the centre of the sphere and its camera can be moved across the circumference of the sphere. The distance between the camera and the avatar is the radius of this sphere. The user is able to change the distance either by editing it or changing its value using the buttons with the '+' and '-' symbols. Pushing these buttons causes the increment or decrement of the distance by 1000 meters.

The relative position of the camera is defined through the use of the two position controllers: azimuth and pitch. The variation of the azimuth controller is actually shows the right or left motion of the camera on the circumference of the sphere while the variation of the pitch controller shows the motion to the forward or backward respectively. The orientation of the camera view is defined using two controllers: pitch and yaw. Pitch is varying the up – down camera's orientation, and yaw is varying the right – left camera's orientation.

4. Contents tab

The Contents tab contain a thematic browser with the configure tree of the layers' visibility of the geo – coded information. The real geo-coded information contained in the terrain scene is descriptive information of the various land features (e.g. mountains, rivers, etc.) and geographic entities (e.g. cities).

5. Logo tab

Logo tab is a Logo – like programming environment where the user is able to write Logo commands (or procedures) and to run them by pressing the 'Insert' key. If the user wants to run all the commands edited on the Logo Editor, he must press 'F6' key. 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 commands or procedures the Logo Response component automatically notifies the problem providing comments.

5.1 Logo commands

Cruislet Logo is based upon Turtletracks language created by Daniel Azuma¹. Below, you may find the commands that were added to the Logo mentioned.

- *createavatar(lat long height model_name)*: Creates an avatar.
- *createavatar (name lat long height model_name)*
- *removeavatar()*: Removes the active avatar.

¹ You may find a detailed list of Logo commands in <http://turtletracks.sourceforge.net/>.

- *activateavatar(name)*
- *setposition(lat long height)*: Sets the avatar in the position specified by the input, that is the latitude, the longitude and the height correspondingly.
- *outputposition()*: Returns the current position of the avatar in the form of a three-element list, where the first element is the latitude, the second element is the longitude and the third element is the height.
- *setdirection(theta fi r)*: Turns the avatar according to theta and fi angles and moves r meters.
- *outputdirection()*: Returns the current direction of the avatar in the form of a three-element list, where the first element is the theta angle, the second element is the fi angle and the third element is the r distance (in meters).
- *stopavatar()*: Stops the active avatar.
- *cleartrail*: Deletes the trail of the active avatar.
- *setcamera(distance azimuth pospitch orientpitch orientyaw)*: Sets camera's properties.
- *camdist setcameradistance(distance)*
- *setcameraazimuth(azim)*
- *setcamerapospitch(pospitch)*
- *setcameraorientationpitch(orientpitch)*
- *setcameraorientationyaw(orientyaw)*

6. Menu

The menu of Cruislet consists of:

Open file: The user is able to open files that have been created by Cruislet environment. When the file is opened all the avatars included in the file are displayed on the terrain scene. In order to see the avatars, the user must change map's viewpoint, as it doesn't change when opening a file. Additionally the user is able to see the Logo code saved in the file.

Save file: The user is able to save the avatars or / and the Logo code he has created in Cruislet environment.

Moving map: The user is able to move the 3D map by using the mouse.

Information: By pressing this button a box is displayed on the bottom left corner of the 3D map. This box displays the description of the location, that is the geographical

coordinates (Latitude and Longitude) and the height of the point of the 3d map that user points using the mouse. By pressing once again the 'Information' button, the box is disabled and disappears from the map.

Print: By pressing the button a window pops up with several print options. The user is able to print the 3D map with additional information about the location description the user is pointing, either in portrait or landscape printing format.

Font size: By pressing the button the user is able to change the font size at the tabs. The default font size is 11.

4) Perspectives

○ In terms of developments

Cruislet environment is a microworld designed to provide students with the ability to be involved in exploratory activities focusing on the use of vectors navigating in 3d large scale spaces. Cruislet can be considered as a microworld developed within the ReMath project as an extension of Cruiser platform (www.cruiser.gr). This extension was based on the notion of navigational mathematics, the mathematical concepts and the mathematical abilities, the development of which is supported navigating in 3d large scale spaces.

From the beginning, Cruiser platform has been developed by [Talent Information Systems S.A.](#) as a new medium for publishing, advertizing, searching, expressing and communicating on the web, based on maps and geographic space. In particular, users with Cruiser can navigate in 3D spaces, browse maps and inspect objects, as well as they can add their own content, share it with friends and import from / export to GPS their own creations.

The development of Cruislet as an extension of Cruiser platform was based on a component-oriented architecture that can be used to provide synergy between end-user programming and efficient behavior of components. Such kind of software environment for exploratory mathematics provides formal ways of expression while maintaining a high level functional efficiency and exploits state-of-the-arts software technologies. Consequently, Cruislet has been designed to provide students with the ability to be involved in exploratory activities to formalize navigational mathematics.

One of the main future developments of Cruislet is to provide alternatives concerning the geographical map that is included in the terrain scene. A library of available maps could be developed offering the possibility to choose the appropriate map for the terrain scene. The provided maps should contain the appropriate geo-coded information in order to take into advance all of the DDA's functionalities.

Another perspective development could involve the construction of a game play joystick that could be used in order to freely navigate the avatars without typing the geographical or the spherical coordinates. This functionality should be useful for younger (primary school) students. This functionality should not replace the use of the current navigating functionalities but rather support them in a preliminary phase of getting accustomed with the included mathematical notation.

○ In terms of deployment

In the time of the project we used Cruislet in a large-scale initiative of the Ministry of Education targeting 4000 mathematics teachers to receive a 96 hour course in using digital media in the teaching of mathematics. We were part of a project within this initiative to train 100 mathematics teacher educators and carried out a 350 hour course to 20 of them and took part in the teaching of all the others. We made Cruislet

available on the ETL website and advertised it as a resource in these courses. The spring semester of the academic year 2008-2009, the use of Cruislet DDA will be part of the “Mathematics Education” course for the undergraduate students in the Primary Education Department of the University of Patras.

DDA 6: Mopix

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

London Knowledge Lab and Institute of Education

Table of content

1) Introduction..... 2

2) Development History 2

3) Perspective/Rationale 4

4) Putting the principles into action: description of the final form of MoPiX 2.0. 10

5) Future Work..... 21

6) References..... 22

1) Introduction

MoPiX (2.0), a software construction kit for making graphics, animations, simulations, and games using equations. Learners using MoPiX relate to equations as a means of expressing themselves creatively. Equations empower MoPiX users to make interacting objects move, spin, and change size, colour and shape. Objects can leave trails as they move. Interactive applications and games can be created containing objects whose behaviour is a function of the state of the mouse or keyboard. MoPiX 2.0 can be used for “serious” purposes such as implementing (and learning about) Newtonian mechanics or for playful creations of colourful animated works of art. Collaborative creations are well supported due to the extreme modularity of applications built upon algebraic equations.

DDA development in Year 3 has concentrated on moving from a Flash-based implementation of MoPiX. MoPiX 2.0 is a rich internet application (a web application with features comparable to desktop applications) using AJAX. The Google Web Toolkit (GWT) supports the interface elements such as tabs, panels, buttons, and editors as well as facilitating communication with server.

2) Development History

Development of MoPiX up to Year 3 has been detailed in previous deliverables and is summarized here:

- Major extension to the equation engine
- Increased equation editor functionality
- Added ability to save and share equations and objects between users
- Simplified user account creation
- Ability to view, copy, and paste underlying MathML
- MathDils integration (load and save)
- Equations categorised
- Description tags for equations

Year 3 developments (in detail) are as follows:

Choosing to re-implement MoPiX using the Google Web Toolkit (GWT) for several reasons, the main ones being:

- Cross-browser support (GWT translates standard Java code into JavaScript that then runs on most browsers)
- Relatively easy UI design with pre-existing elements (widgets) and styling with CSS
- Replacing a long list of scrollable pre-defined equations with a browsable set of web pages containing equations and supporting material and text
- code-test-debug cycle made easier with the programming in hosted mode
- Project structure: standard GWT layout with different packages makes it easy to separate client and server code

Other key developments:

- Design issues:
 - New version based on tabs, separation of the main parts: construction area, Editor, library (construction and running area originally separated too, to become one Stage area later on);
 - Updated Equation code
- Porting of the Editor:
 - Changes in styling (now using GWT widgets with CSS)
 - “Expanding” rather than squeezing text fields (and then zooming) since the browser windows support scrolling and that brings clarity to long equations
 - Choosing and discussing (trying) different panels widgets and other components
 - Creating new equations via recursive call of panel formations
 - Adding MathML generation (including “custom” built functions)
- Interface and functionality
 - Abandoning ‘drag & drop’ principle, one of the features from the MoPiX1 and going for the menu driven actions (one or a double click on the mouse button opens menus with the options)
 - Working on the cascading menus within the Editor (offer of all the operations); starting opening of existing equations; deletion and partial deletion finished; changed/improved the UI with additional buttons, comments etc.
 - Added ability to add equations to objects; run models, and inspect objects
 - Temporal navigation so you can run or step forward or backward
 - Added ability to directly change the size, colour or orientation of objects
 - Added ability to search for nearest time where an object matches its current manipulated state or to set the time explicitly
 - Improved the HTML rendering of equations to use subscripts and subscripts
 - Unary functions programmed into the Editor; work on the opening of the existing equations; testing of the new version
 - Load/Save works using the ReMath MathDiLS repository
 - Publishing of MoPiX2.0 for testing
 - Added ability to give appearances to equations
 - Added ability to copy objects and groups of equations
 - Added ability to split the interface into stage and browser panels
 - Added ability to adjust to the size of a browser (including small ones on mobile devices)
- Greek version – deployment work on v 2.0. In collaboration with the Greek ReMath partners the entire MoPiX system (all interface elements, all documentation, and equation libraries) is available in Greek (by simply adding &locale=el to the URL).

- While exploring potential collaborations with MiGen Project (<http://www.migen.org>), the ability to provide an HTML veneer (icons or rich text) to equations was added. The ability to view equations as sentences in English or Greek was also added. The user can choose how equations are viewed (algebra, algebra with icons, natural language, or rich text/icons). Algebraic expressions can be retrieved for properties of a shape and larger expressions and equations can be constructed by assembling these expressions. The expressions can have iconic or natural language veneers.

3) Perspective/Rationale

Designing MoPiX 2.0

In designing MoPiX 2.0 we took the novel approach of *combining Web 2.0 principles with socio-constructivist approaches to learning*. Web 2.0 principles (O'Reilly, 2005) take the view that learners are producers rather consumers of information. The principles originated in the technology community where it is common for successful entrepreneurs to be self-taught, outside of the formalities of the school curriculum. The overarching rationale for Web 2.0 is that technology should enable community engagement through a participatory culture.

Web 2.0 principles support the interdisciplinary design of learning experiences by focusing software developers on services that support sharing, communication and collaboration between their users and enabling educational practitioners to engage with the underpinnings of many successful knowledge sharing web applications (for example, Jaiku, YouTube, flickr, del.icio.us). This acts as a basis for practitioners and developers to communicate to each other their design knowledge (Mor and Winters, 2007). In this way, learning with technology becomes less about studying the use of the technology and much more about embedding the principles in the design of great learning technologies. These principles are as follows (O'Reilly, 2005):

- *The web as platform*: the web is a platform for participation, hosting (database-backed) applications where many small players (the 'long tail') produce the majority of the content.
- *Harnessing Collective Intelligence*: this is the process by which network effects occur organically as a result of user activity and contributions. In essence, the audience decides what is important. Examples include the blogosphere, del.icio.us and flickr.
- *Data is the next Intel Inside*: Web 2.0 services required data, in particular data contributed by users.
- *End of the software release cycle*: software is delivered as a service, not as a product. Users as viewed as co-developers, resulting in the "perpetual beta". Gaining an understanding of real time user behaviours becomes important.
- *Lightweight programming models*: unlike traditional programming, Web 2.0 applications leverage scripting languages, where the barriers to re-use of code are low. This leaves open opportunities for integrating services provided by others.
- *Software above the level of a single device*: software is no longer bound to only computers. For example, iTunes can link with an iPod and is successfully partly because it runs across multiple devices.

- *Rich user experience*: Web 2.0 applications offer local PC-quality user interfaces and extend these with new innovations, for example collaborative editing of documents.

Putting the principles into practice in mathematics education

In this Section, we outline how MoPiX 2.0 instantiates Web 2.0 design principles. The principles are adapted to the context of mathematics learning based on socio-constructivist principles.

Focus on all learners' uses (The Long Tail)

As MoPiX 2.0 is an 'engine' based upon algebraic equations, learners can engage with different areas of mathematics and physics when using it. One implication of this is that different domains (e.g. Newtonian Mechanics) will have different numbers of learner-contributed models and activities associated with them, i.e. each 'use' has a long tail. With MoPiX 2.0, each use has a place where learners can share and collaborate. Thus, all learners are catered for: long-tail use is designed for in the same manner as more popular activities.

Design decisions that support this principle

- *Ease of saving and access to all resources*: All models are saved on the server using *tags*. An important consideration here is the way in which tagging is supported by the interface, and more generally by the underlying MoPiX architecture. The idea of harnessing collective intelligence can act as a guide. The aim is to support *increased* participation, communication and collaboration by designing interactions where valuable gains are made from participating instead of passively observing. A good example is del.icio.us; if a user saves a link to a webpage, the links of other users who also saved the same link are viewable. Thus, there is a clear advantage to participating rather than using traditional bookmarks.

Focus on resources (Data is the Next Intel Inside)

Resources in MoPiX are primarily student *models* built using *equations* and their associated *meta-data*, such as tags. MoPiX is designed for students to directly interact at the *level of equations* to get things done.

An important point about Web 2.0 resources is that they are useable by other services. This means that the resources produced by MoPiX can be directly fed into other applications. In the context of mathematics education, this means that a teacher might want to have a visualisation of the student activities. By using a web-based tool (e.g., <http://services.alphaworks.ibm.com/manyeyes/app>), the visualisation can use the same data in a 'mash-up'.

Since MoPiX 2.0 is web-based, this allows us to mine data from the server. The primary rationale for doing so is for the teacher to get a detailed understanding of how students are using MoPiX. Example include:

- Interactions
- Equations used
- What students got into trouble?
- Who students models are shared with?

- Which models are the most popular? (Think of Amazon.com's recommendations)
- Each of these can be determined by a simple query of the database. For example, an RSS feed could provide a feed of what are the daily/weekly most popular activities.

An important aspect of being able to mine the data regarding student interaction is that this can be used as the basis for improving the user interface and interactions. Every aspect of screen-based student interaction is logged. Using appropriate mining algorithms, particular events can be identified, for example users dragging to an incorrect location.

The current implementation of MoPiX 2.0 stores every saved model along with user-provided tags. The model is saved as a monolithic MathML expression. We have designed a database schema that would enable a very rich set of queries. In addition to storing models in small, hierarchically linked, database records, we have begun experimenting with maintaining database records of model construction actions (e.g. adding or removing an equation, creating a new object). When these enhancements become operational, database queries can probe for related models with different authors, common problems (e.g. the frequent addition of a group of equations followed by model executions followed by removing those equations), the variety of models that make use a particular equation, which models were built by enhancing the models of others, any students who are frequently changing their model but not making progress and much more). A user-friendly web interface to these queries can be added. It is possible to provide syndicated feeds (RSS) so that contributors could be notified if anyone saves a model using one of their equations or that extends one of their models.

Design decisions that support this principle

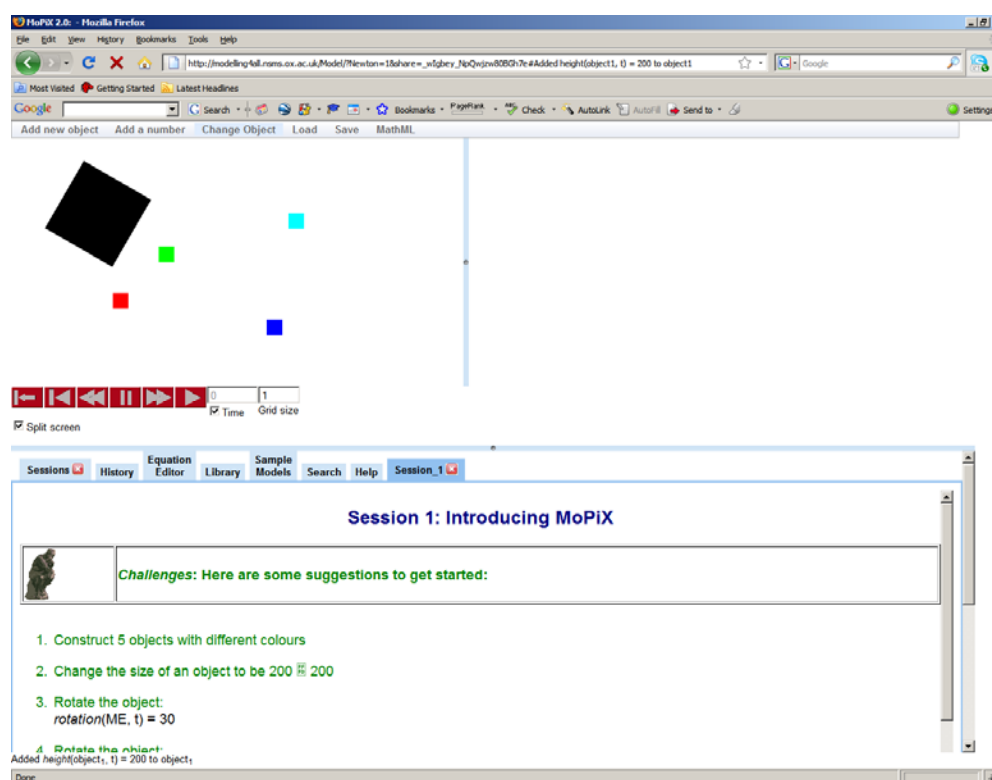
- *Equations as building blocks*: For many students Algebra is experienced as boring and confusing manipulations of symbols that have no personal connection with their lives and aspirations. By using MoPiX, equations become empowering. Students come to see equations as a way to express animations and games. Furthermore, students can *use* equations before acquiring an understanding of how to create or manipulate them. If working with very young students, their first experiences with equations may be as “magical incantations” that cause things to move, change their appearance, or draw a trail. Given a rich library of pre-defined equations, they can use MoPiX to create a wide variety of animations or simulations without first mastering algebra. A student who delves deeper into the *meaning* of equations will acquire greater expressive power by customizing equations and creating new ones.
- *Use of MathML*: MathML is the Mathematical Markup Language (<http://www.w3.org/Math/whatIsMathML.html>) which “facilitate[s] the use and re-use of mathematical and scientific content on the Web”. In our case, it allows for the sharing of resources (including models) between different systems. Thus, anything produced in MoPiX 2.0 can be read by any MathML-compatible system.

Support learners and teachers in adding their own resources (Users Add Value)

A key to supporting learner and teacher use of MoPiX 2.0 is providing them with appropriate tools for development. This places the focus of MoPiX 2.0 on content creation and use, adding value to it as an application/web service. Teachers and learners are encouraged to add their own resources to those already available. This is achieved through *Holistic Integration* and the provision of *ActiveSheets* (see below).

Design decisions that support this principle

- *Layered Learning* is defined as the ability of learners to “choose how far to delve into the working of the tools” (Kahn et al, 2006). In our implementation layered learning is supported by having a shared repository of models, which can be used as the *building blocks for larger models*. To support this practice, learners are required to ‘drill down’ (i.e. inspect the layer of) each model. Moreover, through this process a teacher can view - and begin to understand - how students are working with their models. When developing activities, content can be customised to a particular student’s needs, with appropriate layers defined as is necessary. For example, the content could be layered: equations can be found and used while optionally hiding some of the technical details. In the MiGen project enhancements to MoPiX 2.0, equations can be replaced by rich text and used as such but can also be viewed as ordinary algebra (or as natural language).
- *Holistic integration* MoPiX has been designed in such a manner so as to support students and teachers to easily add and share resources. The key point here is that we wish to *make it easy* for users to engage thus lowering the barriers to participation. The first design decision we made in support of this was what we have termed *holistic integration*.



Screen shot of split screen of the stage and an active lesson sheet

Holistic integration means that each component of the system resides in the browser window. Moreover, anything to do with the system can be implemented in it. An important example of this concept is the shift from paper-based activity sheets to what we have termed *Activesheets*.

- *Activesheets* contain live data. Activities are composed (by teachers mainly but potentially by students) in HTML and are located under the ‘XX’ tab in MoPiX. Important all activities are editable by both students and teachers. Furthermore, the key to Activesheets is the notion of live content. Live Content are equations and models embedded within Activesheet. In order for learners to use the content they simply send it to the MoPiX stage. One potentially interesting functionality of Activesheets is to enable a teacher to provide a carefully designed subset of the potentially huge library of equations and models to both focus the learners and to support them. Additionally, Activesheets are also important for ‘holistically integrating’ documentation and tutorials. They live on the web and benefit from Web 2.0 in ways paper-based activity sheets (Passivesheets! are not likely to do.
- *Collaboration example* Take the fireworks example [add details of the example here]. The Firework activity is designed to support collaboration. Different students can take on different roles. For example, learner 1 might use equations to determine the appearance of the model while learner 2 can focus on controlling the motion of the fireworks. The Web 2.0 aspect of this example is the fact that the model and its equations live on the server and are shared by students facilitates the collaboration. One of the nice architectural aspects about this is that it *scales*: Imagine if a teacher had a class of 30 students and asked them to build fireworks in MoPiX 2.0. Then the peer-production aspects of Web 2.0 could make for a Wikipedia-like process of *loosely coordinated co-construction* of different elements and behaviours to make a class-constructed fireworks simulation.

Aggregating Learner Data (Network Effects by Default)

Only a small percentage of users will go to the trouble of adding value to your application. Therefore: Set inclusive defaults for *aggregating user data* as a side-effect of their use of the application. Students and teachers that use MoPiX but don’t rate, tag, or discuss things still provide valuable data about what is popular, where users are getting stuck, etc.

Design decisions that support this principle

- *Model Recording*: Recording of all models developed: ability to record, step back in time. Thus we are not aggregating across students (although we can do) but instead are aggregating across time. In the next iteration of MoPiX 2.0, teachers and learners can do model building and quit. Then, when you return (either to the same browser with cookies or via a bookmarked URL) their history is recreated (if they choose ‘Restore previous session’). This has the potential to be very useful to teachers in that they can *see the entire history* even if learners moved between computers and browsers.

Low barriers to adoption (Some Rights Reserved)

In MoPiX 2.0 we made a very conscious effort to make barrier to adoption low. Teachers can easily customise the interface. This includes ‘formatting’ for different tasks (e.g. Pond tiling or Newtonian Mechanics), including for different languages. MoPiX 2.0 is modularised so as such customisations can be done with little or not technical support. Furthermore, as previously mentioned, models build in MathML can be imported/exported.

Design decisions that support this principle

- *Open model of development* MoPiX 2.0 was designed so as resources are made available to everyone, free of charge. Both the finished outputs (e.g. resources, student model and activities) and the *processes* by which they were developed (e.g. software code or discussion on curriculum development) are publicly available. MoPiX 2.0 code is available from the Google code repository.

Incremental updating (The Perpetual Beta)

When a user begins to use a Web 2.0 service they follow a link in their browser. The browser then retrieves the client part of the program (as JavaScript) and connects to the programs that run on the servers of the service provider. Because both the client and server parts of the program reside on the service provider’s servers, they can be updated frequently (Flickr deploys new releases (up to) every 30 minutes” - http://blogs.warwick.ac.uk/chrismay/entry/deploy_every_30/). “The smaller and quicker the releases, the less chance of regression, the faster features get to users, and the sooner feedback comes back to the team. Basically, they [Flickr] release pretty much every feature and bug-fix as soon as it's complete – they don't really bother with 'batching' releases like we do.”

[http://blogs.warwick.ac.uk/chrismay/entry/deploy_every_30/] Furthermore, the desktop problems of incompatible file formats doesn’t arise because all data is kept on the servers (though it can be exported in content MathML in the case of MoPiX 2.0).

Design decisions that support this principle

- *Update often* MoPiX 2.0 is constantly updated. This offers the potential for teachers to be brought in as key players in the development process. In the current development model, teacher feedback and inputs was provided by the ETL Athens, who were deeply involved with deployment in schools.

Modular and Loose Coupling (Cooperate, Don't Control)

“Web 2.0 applications are built of a network of cooperating data services. Therefore: Offer web services interfaces and content syndication, and re-use the data services of others. Support lightweight programming models that allow for loosely-coupled systems” (O’Reilly, 2006)

Design decisions that support this principle

- *Lightweight model of development* The model is modularised and distributed (see Winters, Mor & Pratt, 2008).
- *ActiveSheets with Live Equations*: These can be used to support teachers to create learning activities
- *Component-based*: e.g. Leverage the freely available HTML editor

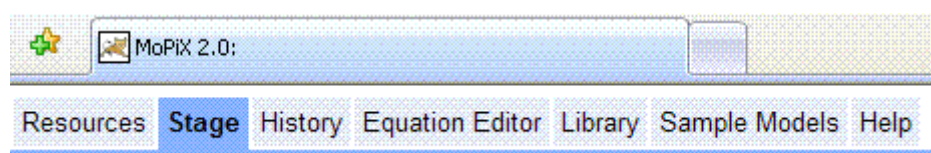
Fit in with the overall learning eco-system (Software Above the Level of a Single Device)

MoPiX 2.0 was designed to work across handheld devices (with less functionality) and PCs, leveraging internet servers. Moreover, because it is web-based, students and teachers can use it across schools and home contexts. Furthermore, there are no installation issues, so fitting it in with existing school eco-systems is easily afforded (unless the MoPiX 2.0 URL is banned).

Design decisions that support this principle

- No plug-ins by definition
- Integrate across devices: PC, Mobile, Game devices

4) Putting the principles into action: description of the final form of MoPiX 2.0



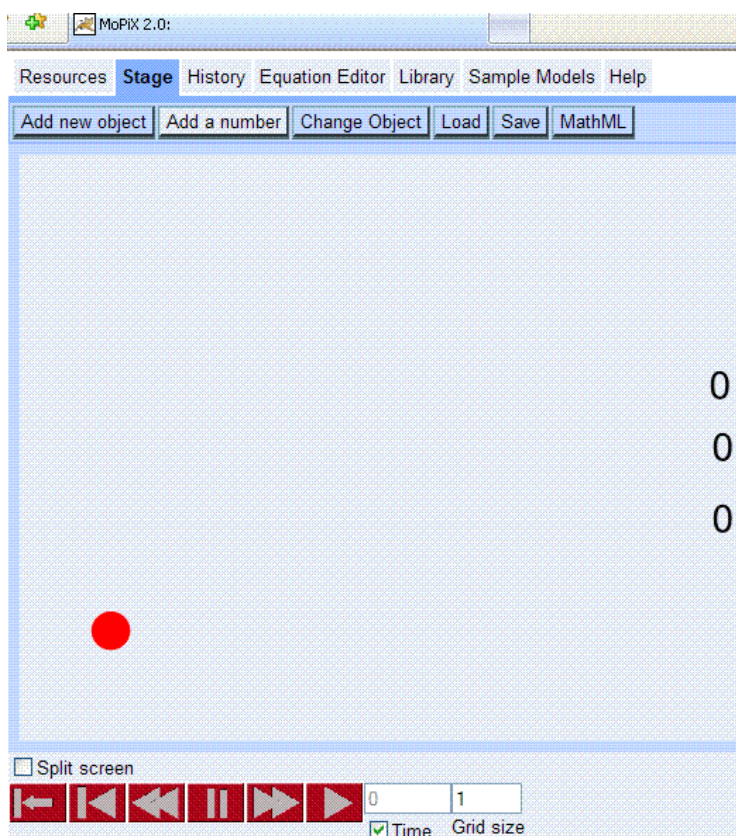
Tabbed environment of MoPiX 2.0

MoPiX 2.0 Environment consists of several different parts:

- *Resources* - offer different information and a possibility of exploring models and equations.
- *Stage* - the construction and play area. Models are built and run in this part.

It is divided into two adjustable panes, left hand side for construction and running and the right hand side one for bringing/keeping and [manipulating the equations](#).

Set of buttons on the top of the Stage controls the addition of new objects, changing an object's size, orientation and colour (in addition to the way of doing it by adding the equations onto an object) as well as loading/saving models (See [Saving, Retrieving Models and Equations](#)) and MathML manipulation (See [Viewing, Editing and Saving MathML](#)).

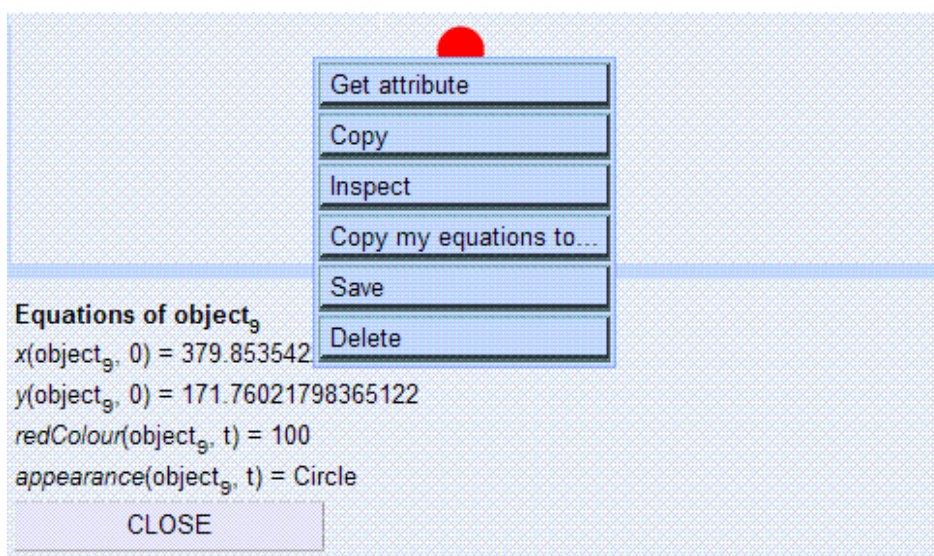


MoPiX 2.0 Stage with loaded model

The set of commands at the bottom of the Stage controls playing and monitoring of models (See [Playing Animations](#))

Object Manipulation

Double-click on an object on the Stage brings up a menu of options for Object Manipulation.

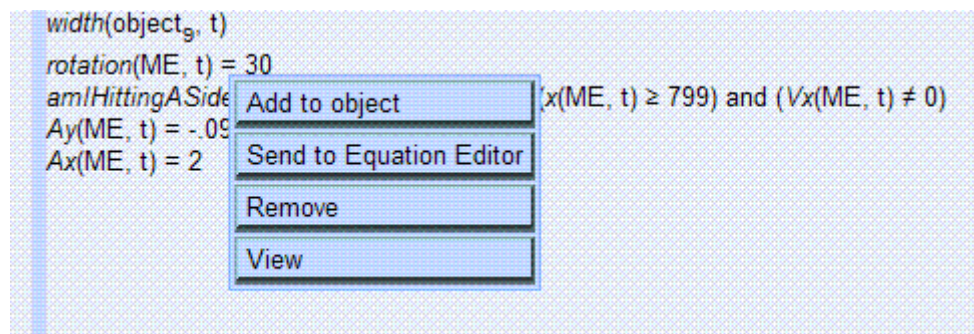


Object Manipulation Menu

An object can be copied, saved or deleted from the Stage. Object's equations can be copied onto another object and an object's attributes can be shown. If an object is inspected, the lower part of the Stage opens showing all the belonging equation.

Equation Manipulation

Left click on an equation brings up a menu of options for Equation Manipulation.



Equation Manipulation Menu

An equation can be added to an object, sent to the Equation Editor (See [about Equation Editor](#)) removed from the Stage or object and viewed in different formats.

- *History* - keeps a list of all the moves.
- *Equation Editor* - the Editor for creating, opening and editing equations. See [about Equation Editor](#)
- *Library* - it is a set of equations based upon MoPiX 1 library. Consists of different sections of equations grouped on their functionality basis: controlling the appearance, position, orientation, size, colour, and trail of the objects.

Sample models - a set of models for the off line use.

The **equation editor** provides a way of creating new equations. It is situated in the *Equation Editor* tab of the MoPiX 2.0 environment. The Editor consists of a button which starts new equation. The equation to be created needs to be analyzed before expanding its left and right text fields: it needs to be seen as a binary tree structure, in order of precedence of its operators. For example, to create an equation like this:

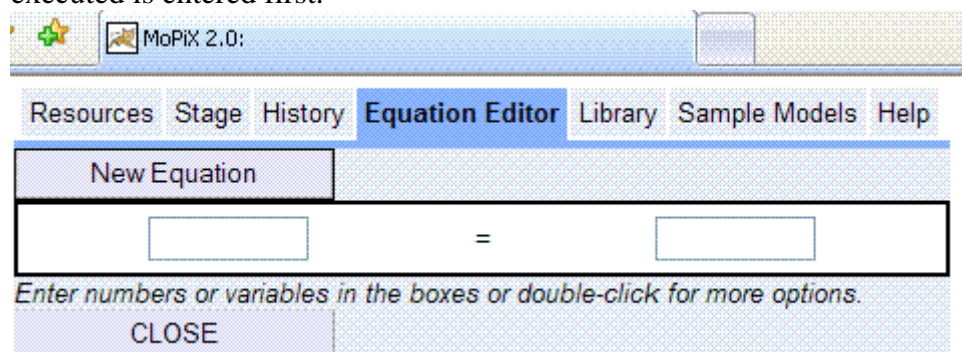
$$\text{redColour}(\text{ME}, t) = \text{redColour}(\text{ME}, t-1) - \text{delta47}(\text{ME}, t) \times 2$$

One should follow this set of actions:

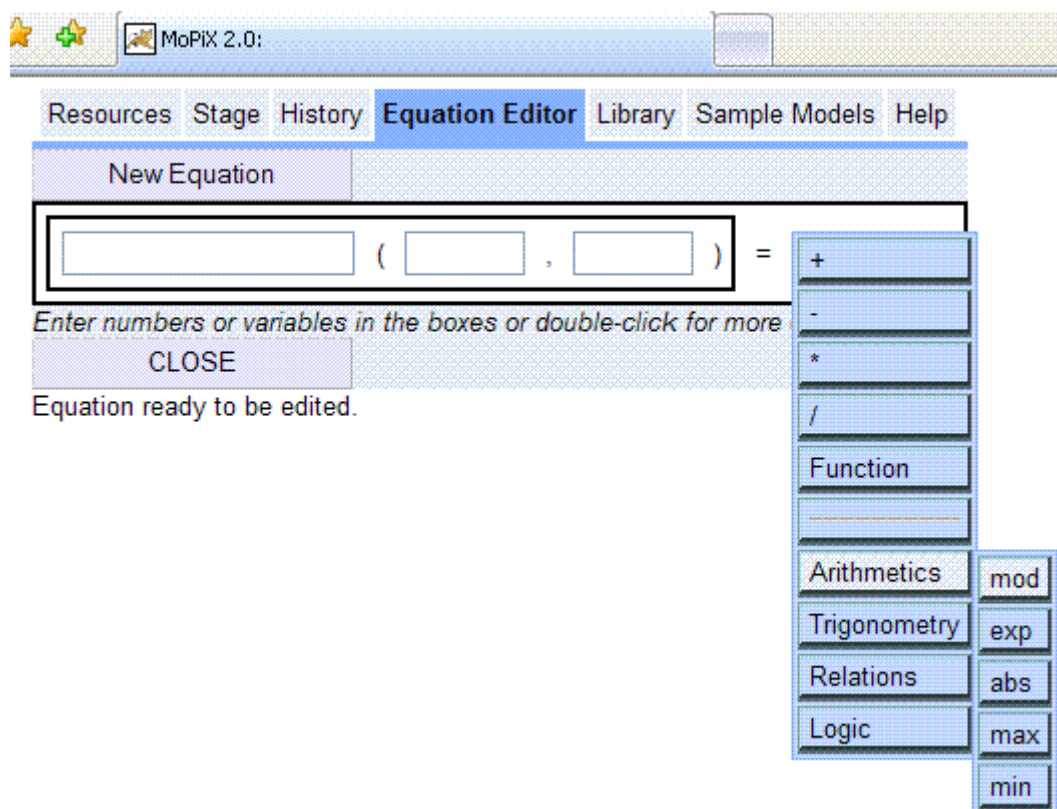
- Click on the New Equation button to start new equation
- Double-click on the left text field and choose *Function*, fill in the values for the function name, object name and time

- Double-click on the right hand side of the equation and choose '-' sign from the operators list
- Double-click on the left hand side of the minus sign and choose *Function*, fill in the functions values
- Double-click on the right hand side of the minus sign and choose '*' sign for the text field to expand further
- Expand the left hand side of the times '*' sign into function and fill the right hand side of the '*' operation with the number

In other words, the operations are entered in such a way that the last one to be executed is entered first.



The following picture shows the cascading menu of different operations to be chosen after a double-click on a text field.

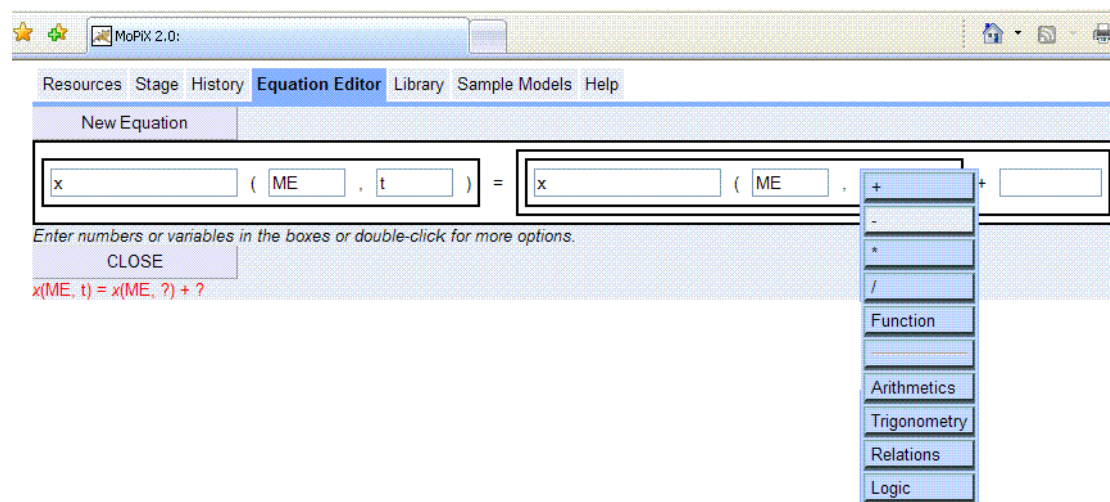


Creating New Equations

The Equation Editor supports creation of user's own equations by opening of a template for an equation clicking on the New Equation button. By positioning the cursor on one of the text fields and double-clicking, a menu of all the operations available appears. When an operation is chosen, a new pair of text fields opens with the desired operator in between (in case of binary operations), a text field with the operator (in case of unary operation) or a function template with text fields for the function name, object name and time.

This can be continued further, until the complete equation has been created. While it is being created, the equation is also displayed on the screen.

Once completed by the editor, it is ready to be sent to the *Stage* where it can be used to define a property or behaviour of an object. Being on the *Stage*, the created equation can be reused many times or sent back to the Editor to be opened and changed.



Supported Operations

The Equation Editor supports the following:

- *Basic arithmetic operations*: addition, subtraction, multiplication, division;
- *Relational operations*: less than, greater than, less then or equal to, greater than or equal to, not equal;
- *Logical operations*: and, or, not;
- *Other arithmetic operations*: absolute value, remainder.
- *Trigonometric operations*: sin, cos, tan;

User Defined Functions are supported in the form of:

functionName (Object name, time)

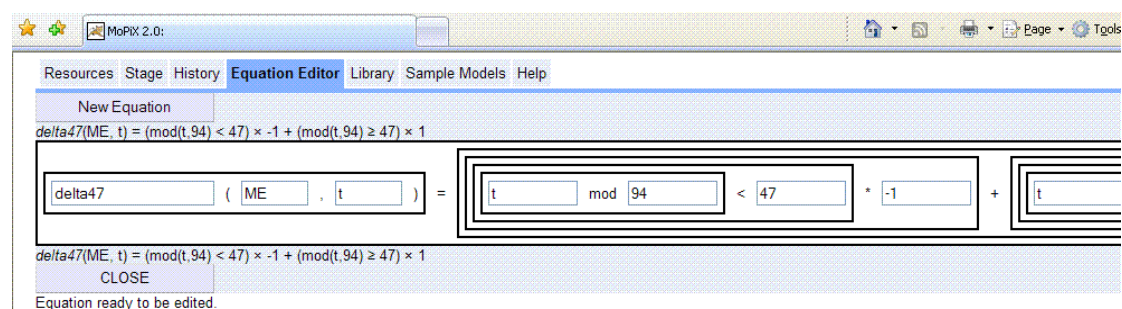
where Object name refers to the object on the Stage currently dealt with (ME) or any other object in relation with the it (OTEHR1, OTHER2, ..., OTHER9). The functions are functions of time and the second argument specifies particular time instance.

Opening Existing Equations

Library , *Stage* and *Resources* tabs of the MoPiX 2.0 environment are all sources of 'live' equations that can be opened by left-clicking and choosing *Send To Equation Editor* from the menu.

The equation can be further edited in the same ways as explained in the previous sections i.e. using expanding and deleting features.

In this way, existing equations can be changed by replacing their parts with the desired elements. The following picture shows an open equation.



Deleting Equations

The equations can be deleted partially or completely.

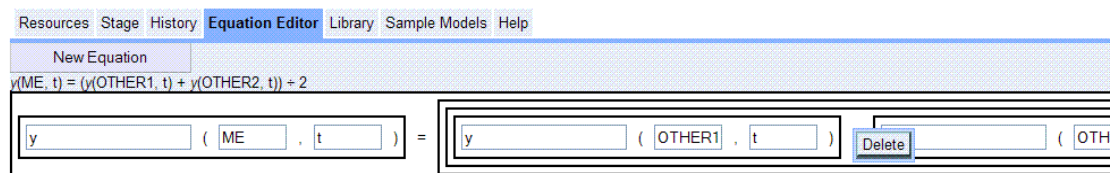
Partial deletion

Parts of an equation can be deleted by left-clicking on an operator and choosing *Delete*. In this way, the operator together with the two belonging operands are removed and replaced with an empty text field. The rest of the equation can further be edited in the same ways as described in the previous sections.

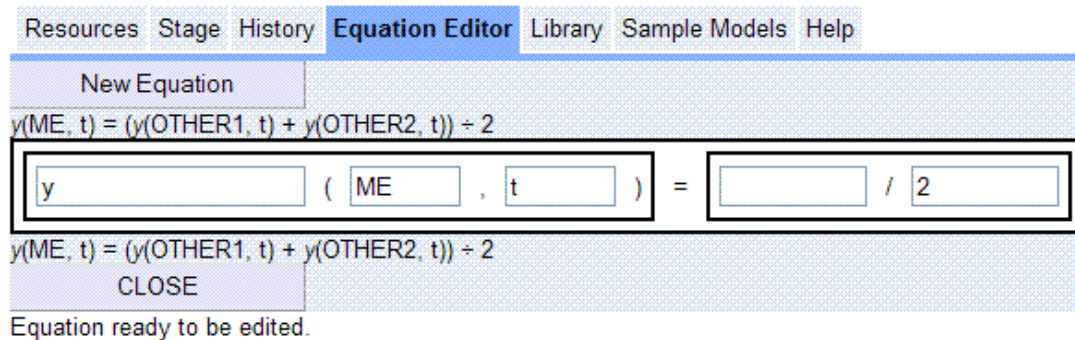
Complete deletion

Complete deletion can be done either by clicking the **CLOSE** button or applying the partial deletion on the '=' sign of the equation.

The following picture shows partial deletion applied to an operator of an equation.



The following picture shows the effect of the above partial deletion.



Creating your Model

Creating a new Model

Elements of your model begin as black squares. Click on the *Add new object* button on the Stage as many times as you need to get the objects. They have no behaviour, colour, or other attributes other than the initial position. When you run your model each element or object i is displayed where its attributes are defined as:

- $x(i, t)$ – the horizontal coordinate of the centre of object i at time t
- $y(i, t)$ – the vertical coordinate of the centre of object i at time t
- $width(i, t)$ – the width of object i at time t
- $height(i, t)$ – the height of object i at time t
- $rotation(i, t)$ – the number of degrees object i at time t is rotated
- $redColour(i, t)$ – the percentage of red in the colour of object i at time t
- $greenColour(i, t)$ – the percentage of green in the colour of object i at time t
- $blueColour(i, t)$ – the percentage of blue in the colour of object i at time t
- $transparency(i, t)$ – the percentage of opacity of object i at time t (100 is fully opaque, while 0 is fully transparent)
- $appearance(i, t)$ – the shape of object i at time t (currently only Circle and Square are supported).
- $x0(i, t), x1(i, t), x2(i, t), \dots, x359(i, t)$ – the horizontal coordinate of a point on the perimeter of object i at time t ($x0$ represents the x coordinate of point on the perimeter at a heading of 0 degrees from the centre, $x90$ is x coordinate of the point on the perimeter reached by a ray emanating from the centre at an angle of 90 degrees, and so on)
- $y0(i, t), y1(i, t), y2(i, t), \dots, y359(i, t)$ – the vertical coordinate of a point on the perimeter of object i at time t ($y0$ represents the y coordinate of point on the perimeter at a heading of 0 degrees from the centre, $y90$ is the y coordinate of point on the perimeter reached by a ray emanating from the centre at an angle of 90 degrees, and so on)
- $penDown(i, t)$ – only when it has a value of 1 will object i at time t leave a trail when it moves
- $thicknessPen(i, t)$ – if $penDown(i, t)$ is equal to 1 then the trail drawn when object i at time t moves has a thickness controlled by this function

- *redColourPen*(*i* , *t*) – if *penDown*(*i* , *t*) is equal to 1 then the trail drawn when object *i* at time *t* moves has a colour whose red component is the percentage specified by this function
- *greenColourPen*(*i* , *t*) – if *penDown*(*i* , *t*) is equal to 1 then the trail drawn when object *i* at time *t* moves has a colour whose green component is the percentage specified by this function
- *blueColourPen*(*i* , *t*) – if *penDown*(*i* , *t*) is equal to 1 then the trail drawn when object *i* at time *t* moves has a colour whose blue component is the percentage specified by this function
- *transparencyPen*(*i* , *t*) – if *penDown*(*i* , *t*) is equal to 1 then the trail drawn when object *i* at time *t* moves has a percentage of opaqueness specified by this function

In addition to the objects you drag out to the construction area there are special built-in objects:

- *mouse* – is an object that is not displayed whose *x*(*mouse*,*t*) and *y*(*mouse*,*t*) correspond to the position of the computer mouse at time *t*
- *a*, *b*, *c*, ... *z*, *A*, *B*, *C*, ... *Z*– are the names of objects that are not displayed whose *keyDown*(*t*) is equal to 1 if and only if the keyboard button labelled *key* is depressed at time *t* (*keyDown*(*mouse*,*t*) is defined similarly to be 1 when the left mouse button is depressed at time *t*)

All of the functions controlling the appearance, position, orientation, size, colour, and trail of objects can be defined directly to have a value such as:

rotation(*ME*,*t*)=*t*×2

which specifies that object to which we refer as *ME* should rotate at 2 degrees per time unit.

Alternatively, functions can be defined in terms of other functions such as:

$$x(\text{object } i, t) = x(\text{object } i, t-1) + Vx(\text{object } i, t)$$

where *Vx* is a function of objects and time that has no built-in meaning to MoPiX and must be defined by other equations.

Partial functions can also be defined, for example,

$$Vx(\text{object } i, 3) = 5$$

Defines a value of 5 for *Vx* of object *i* only at time 3.

All undefined values are assumed for convenience to have the value of zero in MoPiX.

Playing Animations

Animations are displayed when you run your model.

The following set of commands situated on the bottom of the Stage is used to play and monitor animations:

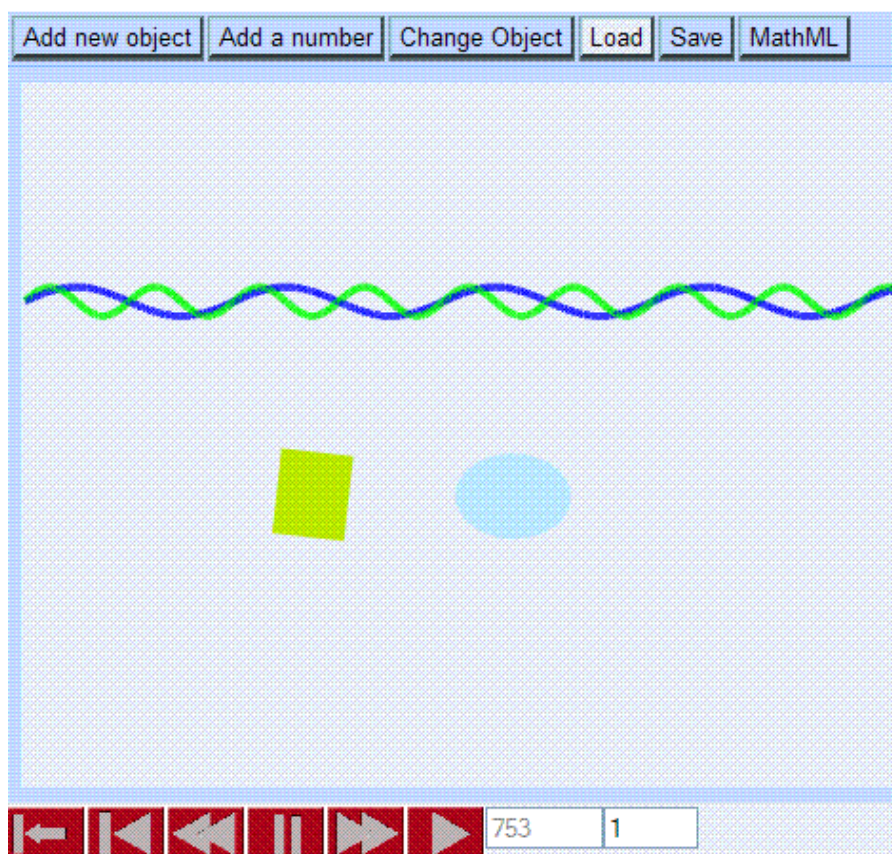


From the left hand side to the right, the commands are as follows:

- *time reset to 0* - resets the time to 0 and your model to the initial state
- *run backwards until time 0* - running the model backwards to the initial state
- *run backwards one step until time is 0* - the model can be monitored one step at a time, in this case going backwards
- *stop running* - pausing your model
- *run forwards one step* - the model can be monitored one step at a time, in this case going forwards
- *run forwards until stop* - Play command: when the Play command is pressed, the time t is repeatedly incremented by 1. After each increment of t , the display is updated.

The box with the time can be edited if the check box is clicked. One can also explore the temporal nature of a model then by moving objects (or changing other attributes) and MoPiX will find the nearest time that matches.

The Playing area on the Stage can be adjusted by dragging away the right-hand side pane. Also in the *Grid* box underneath, one can change the grid size (the number of pixels per unit of distance) by entering a number.



Playing the Two Animated Objects with Graphs model

Viewing, Editing and Saving MathML

The button *MathML* on the Stage opens a window pane with the current model's MathML XML (all the objects on the stage). The options are offered underneath. The usual right-click menu options: select, cut, copy, paste and delete are all allowed.

You can edit the XML and when you close the window the model is updated appropriately. (Please be careful since the error handling is weak.) Alternatively, you can copy and paste the contents into a file or email message. To restore the model at another time or on another computer you can copy the saved XML to the clipboard and paste it into the XML window.



Showing MathML of the Growing Rectangle model

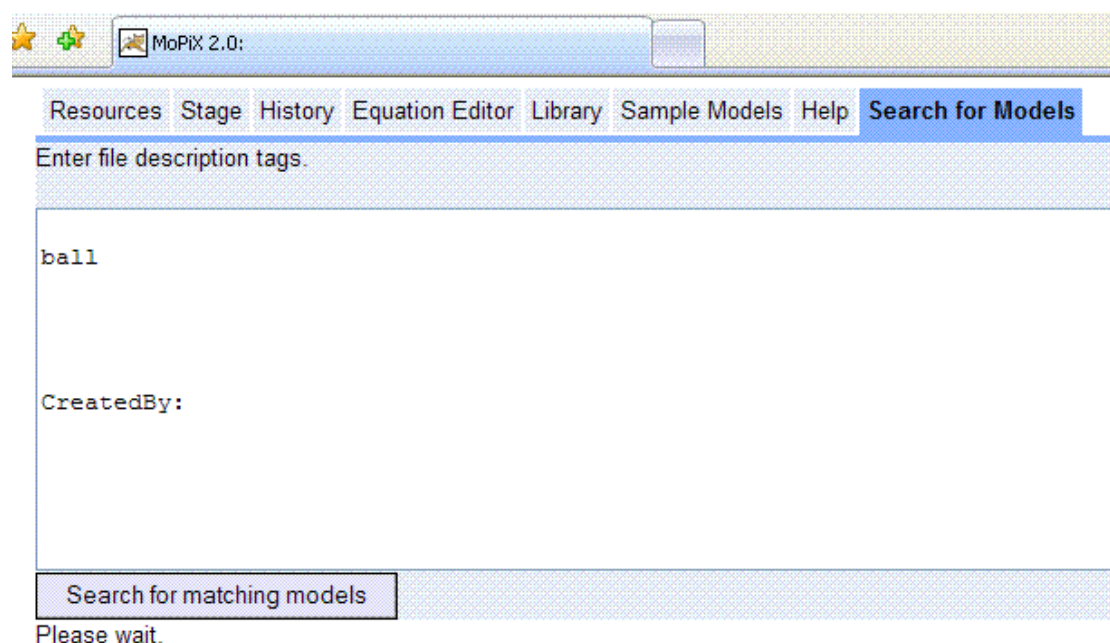
Saving, Retrieving Models and Equations

To Save a model along with its associated equations click on the *Save* button on the Stage. Dialog text boxes open requiring a User Name. Press Enter when this has been completed and a description window opens. A short description of the created model should be entered followed by OK. The status line at the bottom of the stage will show the progress of the uploading.

To Load the saved work by any user, click on the *Load* button on the Stage and after entering the User Name, enter the description (key word or a longer description) and, if desired, the author of the work in the CreatedBy: line.

The tag "Type:" is also added that can have the value "model", "object", or "equation". The tag "CreatedOn:" is added with the value of the time and date the model was first created. The tag "LastModifiedOn:" is added with the value of the time and date the model was last updated.

The status line will report on the progress on the downloading.



Searching for models with tag 'ball' created by all users.

5) Future Work

Making MoPiX 2.0 a web-based application enables many promising areas of future development. It could be embedded in a Web 2.0 site where teachers and students could share models, sets of equations, tightly integrated lesson plans, tutorials, puzzles, and surrounding material at a web site. The same site could directly (or indirectly via integration with other sites) provide tools for discussing, rating, reviewing, and tagging the contributed materials. The ease with which users can explore and then build upon the creations of others has the potential of leading to a vibrant supportive community.

Another avenue of future exploration is the mining of the usage traces of all the online users. One could develop tools to target improvements to the system to where it is most needed based upon data mining the server's databases. The same databases could enable the community to become "self-aware" in the sense that they can see which models are frequently downloaded or embedded in richer models, which equations are used in a large variety of models, which lesson plans are popular and so forth. The databases could be used to provide teachers with high-level summaries of what their students are doing.

Because every user action is communicated to the server it should be easy to support real-time collaborative creation and exploration activities. For example, Google Docs currently enables users to collaboratively create in real-time spreadsheets, presentations, and word documents. We can imagine students using a future version of MoPiX to collaboratively create and edit equations and models.

MoPiX is best suited for students with some familiarity with algebraic notation. However, it could be enhanced for by younger students by providing iconic and natural language veneers to expressions and equations. Students could start with the

power of equations while postponing their exposure to complex mathematical notations.

Finally MoPiX could be extended to cover a richer subset of mathematics. An example would be the addition of the Sigma operation to specify the sum of a large number of elements.

6) References

Mor Y. and Winters, N. (2007) Design approaches in technology enhanced learning, *Journal of Interactive Learning Environments*, 15(1), 61-75

O'Reilly, T (2005)
<http://www.oreillynnet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html>

Winters, N. Mor, Y. and Pratt D. (2008) The distributed developmental network – d2n: a social configuration to support design pattern generation, In P. Goodyear & S. Retalis (Eds). *Technology enhanced learning: Design Patterns and Pattern Languages*. Rotterdam: Sense Publishers.

Annex 1

The internal review process of DEL16

The review process used four researchers of different Remath teams. Each person was asked to review three contributions which give two reviews for each contribution, according to the following list:

Giampaolo Chiappini: Cruislet, Aplusix, Casyopée

Jean-Michel Gélis: Alnuset, Mopix, MaLT

Christos Markopoulos: Aplusix, Mopix, MaLT

Jean-François Nicaud: Alnuset, Cruislet, Casyopée

The indications to structure the contribution were:

- 1) Last evolutions of the DDA (4 to 6 pages)
- 2) History of the design and development of the DDA (1 to 3 pages)
- 3) Description of the final form of the DDA (5 to 8 pages)
- 4) Perspectives (1 to 3 pages)
 - a) In terms of developments
 - b) In terms of deployment

Below are the reviews.

1. Review of Giampaolo Chiappini

Observation on the structure of the three reports (Aplusix, Cruislet, Casyopée)

The first section of the Aplusix report and of the Casyopée report is respectively a “General presentation” of two pages and half (Aplusix), and an “Introduction” of two page and half (Casyopée). I note that in the Nicaud’s indication there is no reference for a section 0 with these characteristics. Moreover, in Cruislet report the section 3 (Description of the final form of the DDA, 5 to 8 pages) is substituted by a reference to the manuals of the DDA, that are detached from the report and are quite long (60 pages).

I think that all the reports should respect the same structure in terms of common sections and of their approximate lengths.

Perhaps, it could be useful to give the indication to insert in all the reports a brief Introduction. Moreover, I think that in the Cruislet report a brief description of the final form of the DDA is necessary (manuals cannot substitute this description).

Observations on the Aplusix report.

Except the above described observation concerning the first section of the report, I have not any relevant observation to do on this report.

Section “last evolution of the DDA”: No observations

Section “History of the design and development of the DDA”: No observation

Section “Description of the DDA”: No observation

The section “Perspectives”: No observation

Observation on the Casyopée report

Section “last evolution of the DDA”: make more explicit the modifications made in this version of Casyopée with respect to the previous version. Moreover, in the part “Better definition of the measure window” the authors write that some changes have been made to improve this window but in the following part they describe the buttons of the window’s interface without specify what are the new features. A brief introduction in which the authors explain the effective changes that have been made could probably clarify this part.

Section “History of the design and development of the DDA”: this part is constituted by a list in which the author explain the changes made at the DDA. Perhaps, in this section some contribution concerning the aspects, the ideas or the events that have contributed to modify the design of the DDA could be inserted and appreciated by the reviewers.

Section “Description of the DDA”: the authors explain the interface of the symbolic window. At the end of this part, the authors state that this window includes facilities to prove properties of functions with the help of theorems. Nevertheless, no indication is given to explain in which way these functionalities work (how it is possible to select theorems? In which way are they included in the window?...). The second part concerning the geometrical window is clear.

The section “Perspectives”: information concerning the deployment of the DDA is missing

Specific remarks

In the figure p.3 “expression” and not “expression”

In the text pag. 3 the triangles OMP and AMR are not rectangle triangles but equilateral triangles.

In the figure pag. 4 the axis (oj) is named i

At the end of pag. 3 the authors write “An algebraic proof can be made by choosing a variable related to M and exporting the functions corresponding to the two calculations into the symbolic window.” It could be useful to explain in which way it is possible to choose the variable and to export the corresponding function in the symbolic window.

The important note at the end of p. 6 could be inserted in the fourth point: perspective

Observations on the Cruislet report

Section “last evolution of the DDA”: No observations

Section “History of the design and development of the DDA”: Perhaps, in this section some contribution concerning the aspects, the ideas or the events that have contributed to modify the design of the DDA could be inserted and appreciated by the reviewers (This is the same observation of Casyopée report).

Section “Description of the DDA”: I think that a brief description of the DDA is necessary. This description has to be focused on what has been developed within the Remath project and this, eventually, has to be distinguished from what had been already developed

The section “Perspectives”: Information concerning the deployment of the DDA is missing

2. Review of Jean-Michel Gélis

Observations on Alnuset

AlNuSet report includes three parts.

The first one focuses on the latest evolutions of the DDA. Main rich improvements are clearly explained, that greatly extend previous capabilities.

The second part gives an insight this DDA’s history. It details different important and decisive stages such as integrating algebraic variables on existing representations. Others important elements are pointed out, e.g. improvements related to the different components.

The last part of the report describes the final form of the DDA. All its components are detailed with numerous figures and clear resolution examples that support the given explanations. Note that this part concerns not only the final form of the DDA, but also includes various others considerations such as the main guide lines that were respected when designing the different component of the DDA (see the three pedagogical necessities).

Finally, a last paragraph presents the perspectives and gives some insight into its commercial exploitation and their consequences.

Suggestions

This document is a rich one, a lot of dimensions are already explored and it’s not sure that it will be possible to take into account the following suggestions. My real interest for this DDA as a reviewer, and a scientific curiosity motivate the propositions below.

1) About technical points

Few elements are given concerning the algebraic calculus abilities available in the DDA. May be the ITD-CNR team encounter no important limitations due to calculus abilities, may be some difficulties had some important and interesting consequences in the final work.

For example, what about the determination of polynomial roots? It is mentioned that “*The solution of equations and inequations is not limited to a degree less than or equal to 4.*” and further “*the algebraic line component was designed to provide other two very important instrumented techniques for the algebraic activity, namely to find the roots of polynomial with integer coefficients and to identify and validate the truth set of algebraic propositions*”. Some questions may be asked: was it possible to find all the solutions of equations and inequations in all cases, what happens if they can’t be explicitly written, have these limitations some consequences on the working space and the possible expressions?

Another point is the pattern matching procedure that allows applying transformations rules. Is this point not a possible difficulty that might have important consequences upon the interaction with the student and his/her working context? For example, the rule $a^2 - b^2 \rightarrow (a+b)(a-b)$ (that is given to illustrate user rules created from available transformation rules) can certainly be applied to $x^2 - 3^2$. The question is to know if this same rule can be directly applied to $x^2 - 9$, to $9(4x+1)^2 - x^2$ or to $5(4x+1)^2 - x^2$ in order to know the precise transformation that a student has to performed before using the rule he/she chooses.

Some precisions (even brief and general ones) about the two previous points will shed some light on the important work that was carried out and has possibly some consequence on the final productions and interactions. Authors refer several times to CAS methods that were perhaps partially used to succeed in implementing the required technical capabilities.

2) About a user community

Important decisions all along the DDA development have been made for didactical reasons. These reasons are visibly coming from the authors’ experience and analysis as mentioned several times in the document (as in this excerpt “...*could be very interesting and effective from a didactical point of view*”).

The report does not mention any other origins, except in the final perspectives paragraph where it is expected to produce “*didactical scenarios to support the familiarization with the system and its integration in the didactical practices*”, that suppose to work with users and certainly integrate their eventual propositions.

Perhaps the team has proposed the DDA to users (teachers, students, others researchers...) and perhaps remarks have modified or inflected the initial development. If this significantly occurred, it will be certainly interesting to know how some external propositions have been taken into account in the DDA.

Observations on Malt

The report has no introduction and begins directly with the last important evolutions of the DDA (such as the substitution of the 3impact game engine with the jMonkey 3d game engine and a new GUI). The project history is then described, 2 levels are distinguished. A third part explains the final form of the DDA and some perspectives end the report.

Comments and suggestions

Introduction

I agree with Jean-François, introducing the document would be better.

The first seven lines of the final part (Perspectives, in Terms of Developments) can be certainly used. The idea is to give some general elements about the main goal and the context of the project.

Progression and repetitions

The first two parts of the report (last Evolutions and History) are clear and precise.

The third one (Description) includes some repetitions. Its first three points were already partially explained above, sometimes in the same words and with the same figure. Perhaps it is possible just to refer to the right previous paragraphs (instead of all rewriting, except if the global consistency seemed to be lost).

The last point of this third part (Using the MaLT environment's Features) gives numerous examples which are very clear and no redundant. These examples are particularly relevant and they really clarify the presentation.

Users

The team has realized a very important technical work and the report focuses on this part of the project.

However, the final paragraph (Perspectives, in Terms of Deployment) mentions that MaLT was amply used by numerous users (including students and mathematics teachers). It would be very interesting to know if some users' remarks have been taken into account by the team or have inflected the orientations of the design.

Perspectives, in Terms of Deployment

This final paragraph reports a very important deployment for this year. It is possible to assume that these actions will be renewed for the next years.

The suggestion is to mention (if any) quite new deployment possibilities. For example, the team has made MaLT widely available. Consequently, interesting questions are raising, such as : has the team planned to collect users' remarks or suggestions, has it planned to follow and study their productions with the DDA, has it planned to eventually integrate some modifications ?

Observations on Mopix

The report is made of three main parts. After an introduction, the first important paragraph focuses on the DDA history - mainly for this last year- and gives numerous detailed technical elements.

The second one deals with perspectives and rationales. The main idea is "*combining Web 2.0 principles with socio-constructivist approaches to learning*". The main principles of Web 2.0 are clearly explained and design decisions supporting these previous principles are presented (see holistic integration, ActiveSheets...).

A third important part describes the final form of MoPiX 2.0. A wide range of available capabilities are explained, with many relevant pictures. A final paragraph presents the future work.

Suggestions

The following paragraphs propose some possible and non-crucial ideas to facilitate the understanding of the described work.

Presentation order

The first suggestion is related to the presentation, more precisely to the order of its different parts.

The proposed plan is a logical one, which begins to explain the Web2.0 principles and the design decisions that support them, before describing in details the final and resulting form of the DDA. On the one hand, this presentation is relevant for a reader who already knows MoPiX with enough details.

On the other hand, a reader who has only a vague idea of the precise productions of MoPiX, will put no matter behind the detailed principles. For such a reader, a description of the DDA has to come before the explanations of the Web 2.0 principles. If he/she is aware of what MoPiX is able to do, he/she will better appreciate the quality of this DDA that respects and supports Web 2.2 expounded principles. It's possible to think that a reader used to MoPiX may also draw some advantage of such a presentation.

Some examples

Another suggestion is to add some short examples to illustrate what MoPiX may product. This will certainly allow getting a better idea of all the possibilities of this DDA. Some successive pictures of a given context might be provided (with its corresponding equations), each picture displaying the configuration for a particular value of the time parameter.

The document mentions several times some recurrent examples, coming for example from the Newtonian Mechanics field or related to fireworks. Some pictures (probably a small set of successive configurations) of what can be produced with MoPiX will certainly help the reader to understand the interest of this DDA with respect to the design principles, for example in the context of collaborative learning.

It will be certainly also be useful to give a short and precise example of such a collaborative learning (in the same previous contexts) that would illustrate the richness and interest of the work done by users.

Users

Another point can be proposed. There are few mentions of real users in the context of mathematics education. It will be very interesting to know if some observations or remarks of users (pupils, teachers, researchers) had some significant importance when designing MoPiX, for example by changing a development priority or enhancing the importance of a given principle.

The final paragraph mentions the possibility as future work of proposing MoPiX to students familiar to algebraic notation as well as younger ones. It is possible to imagine that an analysis of the productions of groups of students or of teachers will be of a great interest to measure the efficiency of MoPiX in the context of mathematics education for example.

3. Review of Christos Markopoulos

Malt

1) Last evolutions of the DDA (4 to 6 pages)

OK

2) History of the design and development of the DDA (1 to 3 pages).

Although the development of the three Variation tools (the 1d Variation Tool (1dVT), the 2d Variation tool and the Vector Variation Tool) is associated with the three version of the DDA, a clear chronological presentation of the development of the DDA from the first version towards the last one is appropriate.

3) Description of the final form of the DDA (5 to 8 pages)

OK

4) Perspectives (1 to 3 pages) in terms of developments& deployment

OK

Remarks:

I think that an introductory section of a brief presentation of the DDA would be appropriate as well as a clearer chronological presentation of the development of the DDA from the first version towards the last one

Mopix

1) (Introduction)

OK

2) (Development History)

In this section after a short summary of the history of development the last evolutions of Mopix are presented.

3) (Perspective/Rationale)

This section includes justifications concerning the design and the development of the DDA.

4) (Putting the principles into action: description of the final form of MoPiX 2.0)

OK

5) (Future Work)

OK

Remarks:

In the case of Mopix , due to the major evolutions of the DDA, it is difficult to follow the last changes in comparison with the previous versions of the DDA. Moreover, the proposed structure of the deliverable (at least in terms of headlines) should be used.

Aplusix

1) General presentation (Introduction)

OK

2) Last evolutions of the DDA (4 to 6 pages)

OK

3) History of the design and development of the DDA (1 to 3 pages)

OK. There is a clear chronological presentation of the development of the DDA from the first version towards the last one.

4) Description of the final form of the DDA (5 to 8 pages)

OK

5) Perspectives (1 to 3 pages) in terms of developments& deployment

OK

Remarks:

In the case of Aplusix there is not any justification for the choices that have been taken concerning the design as well as the development of the DDA. I consider that the presented evolutions of each DDA should be clearly justified in terms of theoretical considerations. In the case of Mopix and Malt such kind of considerations is provided.

4. Review of Jean-François Nicaud

AINuSet

1) Last evolutions of the DDA (4 to 6 pages)

OK

2) History of the design and development of the DDA (1 to 3 pages)

OK

3) Description of the final form of the DDA (5 to 8 pages)

OK

4) Perspectives (1 to 3 pages) in terms of developments& deployment

OK

Micellaneous

I think that “popup menu” concerns a menu which appears when one clicks on the right button of the mouse in a place where no menu is previously shown. In your case, there is a “File menu” in the main menu which is not a popup menu and there are “popup menus” in fig. 4 and 5.

Cruislet

1) Last evolutions of the DDA (4 to 6 pages)

The reader who has not Cruislet in mind cannot understand several things like the notion of “avatar” and “avatar tab”. Please add an introduction before “last evolution” describing the bases of Cruislet.

2) History of the design and development of the DDA (1 to 3 pages)

OK

3) Description of the final form of the DDA (5 to 8 pages)

Please, write at least 4 specific pages for this section.

4) Perspectives (1 to 3 pages) in terms of developments& deployment

The “deployment” relates only past events. I think the reviewers would like to have idea about the future. Can you tell also what the economical model is for Cruilset? Is it, or will it be, free, purchased, by subscription?

Size

The overall size is small. If we change the margins to 2.5cm and the line spacing to 1, we get 6 pages. Casyopée has 14 pages. Others have at least 20 pages.

I suggest you write a 2 pages introduction and a 4 to 5 pages “Description of the final form”.

Casyopée

1) Last evolutions of the DDA (4 to 6 pages)

OK

2) History of the design and development of the DDA (1 to 3 pages)

The written history concerns the last year when a global history (2 years) was expected.

3) Description of the final form of the DDA (5 to 8 pages)

OK

4) Perspectives (1 to 3 pages) in terms of developments& deployment

The two last paragraphs are devoted to perspectives with a limited scope as well for developments as for deployment.

I think that more elements can be said about development, like existing features that can be improved and new features that can be added. I suggest to move the “tutorial environment” of the last sentence towards the “development” perspectives. I suggest to replace the sentence “We expect then the development of a regional community, before extending wider” by something like “We plan to set of a regional community of users, and later to extend then community to France”, i.e., to say the same thing in a more active way.

5. Pages and style

Chapters	Casyopée	Aplusix	Mopix	Alnuset	MaLT	Cruislet
Last evolutions (4 to 6 pages)	5p	6p		8p	3.5p	6p
History: design & development (1 to 3 pages)	0.5p	1.5p	2p	3p	3.5p	0.5p
Description of the final form (5 to 8 pages)	5p	6p	12p	7p	15p	Annex: Manual: 39p Logo: 20p
Perspectives: Developments & deployment (1 to 3 pages)	1p	1p	1p	1p	1p	1.5p
Added chapters						
Introduction	2.5p	3p	0.5p			
Perspective/Rationale			7p			
Font	Times	Times	Times	Calibri	Verdena	Times
Size	12	12	12	12	10	12
Line spacing	1	1	1	1	1.15	1.5

6. Decisions

Add an introduction to Alnuset, MaLT and Cruislet.

Extend the Cruislet “Description of the final form”.

Use Times 12 font with line spacing between 1 and 1.2.

Use at least 2 levels of numbered headers: “1.” For first level and “1.1.” for second level.

Follow the reviewers’ suggestions as far as possible.