Algebraic Equations as a Creative Medium

Ken Kahn[1] and Niall Winters[2]

Abstract

We describe MoPiX, a software construction kit for making graphics, animations, simulations, and games using equations. Children 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 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.

Keywords Software construction kits, equational programming, MoPiX

1. INTRODUCTION

Children have been creating graphics, animations, simulations, and games using child-friendly programming systems since the late 1960s. Tools commonly used today include LOGO, Smalltalk/Squeak, Alice, AgentSheets, Scratch, and ToonTalk [1]. The work reported here follows in this tradition except that equations, rather than computer programs, are used to describe dynamic behaviours. Alternatively, MoPiX can be described as using algebra as a programming language.

For many children 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. Children come to see equations as a way to express animations and games. Furthermore, children can *use* equations before acquiring an understanding of how to create or manipulate them. A young child's 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 child who delves deeper into the *meaning* of equations will acquire greater expressive power by customizing equations and creating new ones.

2. DESIGN OF MOPIX

The overall goal is for learners to iteratively construct their own game environment in which characters and events are determined by the use of applied mathematics. To do this they progress

[1] ToonTalk@gmail.com, University of Oxford, Oxford, UK.
[2] n.winters@ioe.ac.uk, London Knowldege Lab, London, UK.

through a number of steps and share the constructions they create. The games are built by following a construct-animate-investigate-collaborate cycle, based on constructionist principles [2].
A growing body of research indicates that playing computer games can lead to serious learning and that learning can be deeper and more richly interconnected if game playing is combined with game making [3, 4]. With MoPiX a game component can simply be an agreed upon goal or challenge or, more ambitiously, can be embedded in the software.

While it is possible for students to use MoPiX individually, a particular feature is facilitation of student-student interaction and collaborative problem solving among small groups of students. Student constructions are sharable between mobile devices (such as ultra-mobile tablet PCs) and inter-group communication is facilitated by the open nature of knowledge sharing. Different students within a group can explore their creativity by working on different aspects of a scene, which can then be combined.

Furthermore, the design of MoPiX aims to provide a multi-semiotic set of representations that will offer a rich system of potential meanings. These range from visual simulations of motion that should connect to everyday physical experience to more-or-less conventional mathematical representations using algebraic and graphical forms. The linking of mathematical to visual representations is intended to support learners in using mathematics and the principles of laws of motion as means of making sense of the world.

3. EQUATIONS IN MOPIX

MoPiX is built around the surprisingly expressive equations of the form

$$f(object_i,t) = \ldots$$

These equations always define functions of two arguments: an object and the time. For example,

$$x(object_1,t) = t$$

states that $object_1$ moves to the right at a constant speed. And

$$y(object_1,t) = t \cdot 2$$

states that it should also move upward at double speed. Combining the two equations causes the object to move upwards at a 30° angle.

MoPiX consists of the following components:

- An engine that computes the value of functions for any object at any time using the equations the user has associated with objects.
- An animation engine that, for any integer t, displays all the objects with the state of their display attributes at time t. There are display attributes for the position, orientation, size, red, green, and blue colour components, transparency, and shape. Furthermore, there are pen attributes that specify attributes of a trail, if any, an object leaves as it moves. Typically the real-time elapsed between t and t-1 is small enough that a high frame rate is achieved resulting in smooth animation.

- A work area where users add and remove equations to and from objects
- An equation editor
- A library of equations
- Import and export of equations in the XML-based MathML standard to a web service repository

When MoPiX is launched within a web browser, the user sees a large construction area called the stage. See Figure 1. The 'Stage' tab enables one to add new objects and alter them interactively. Equations can be added to objects. The equations can be created in the 'Equation Editor' tab, by combining expressions, and by sending them from the library.

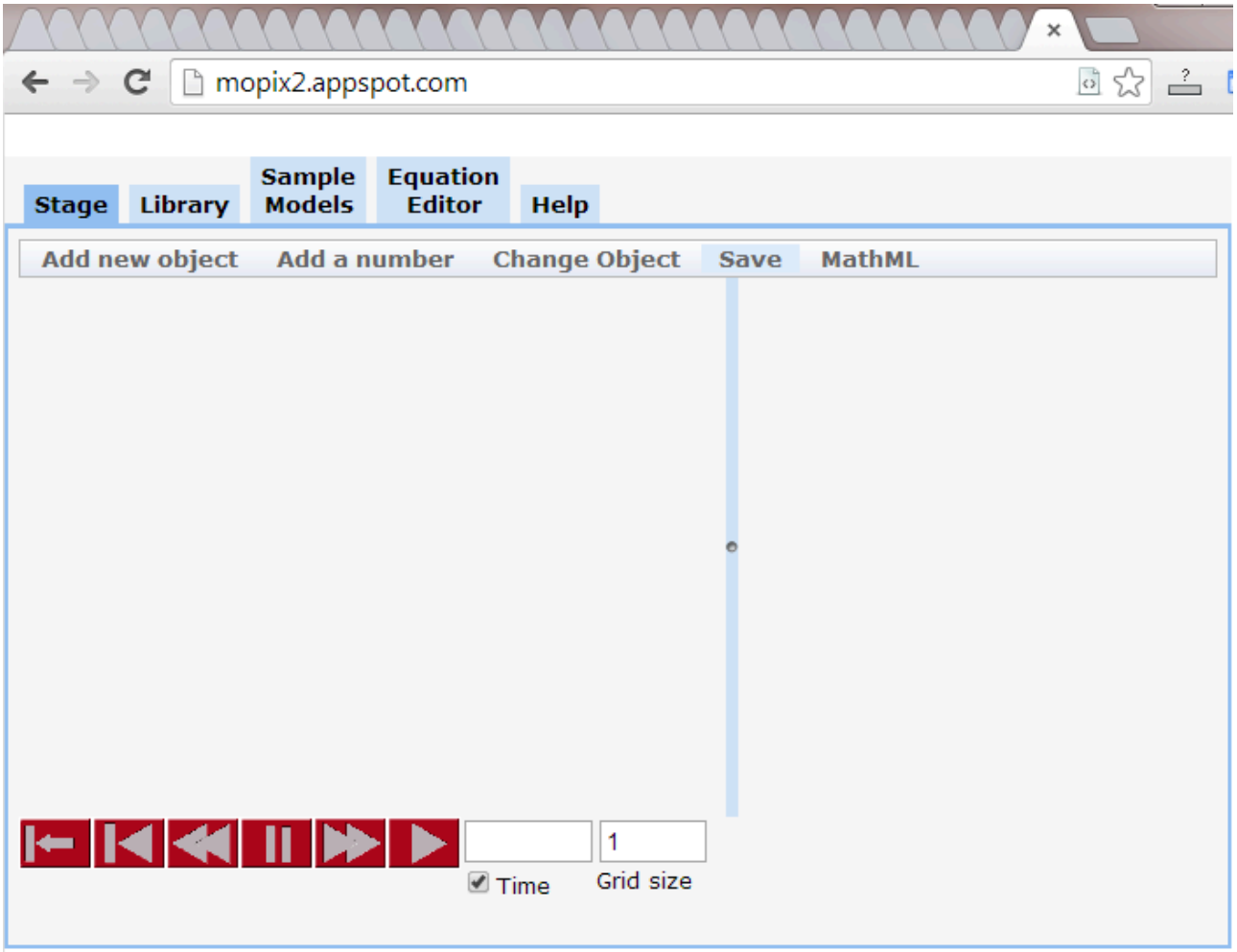

Figure 1 - Initial workspace

Fresh equations are typically a template for an equation in that they don't specify which object they refer to. The template variable ME is used to reference objects. When an equation is dropped on an object, the identifier ME is replaced by the identifier of the object. In Figures 2 and 3 we see the result of adding equations for colour, rotation, and height to the square on the left and equations for colour, width, and height to the circle on the right. The two tiny circles at the upper left have equations that cause them to move as the rotation angles of the square and circle change. They both leave a trail behind thus producing a graph specified equationally.

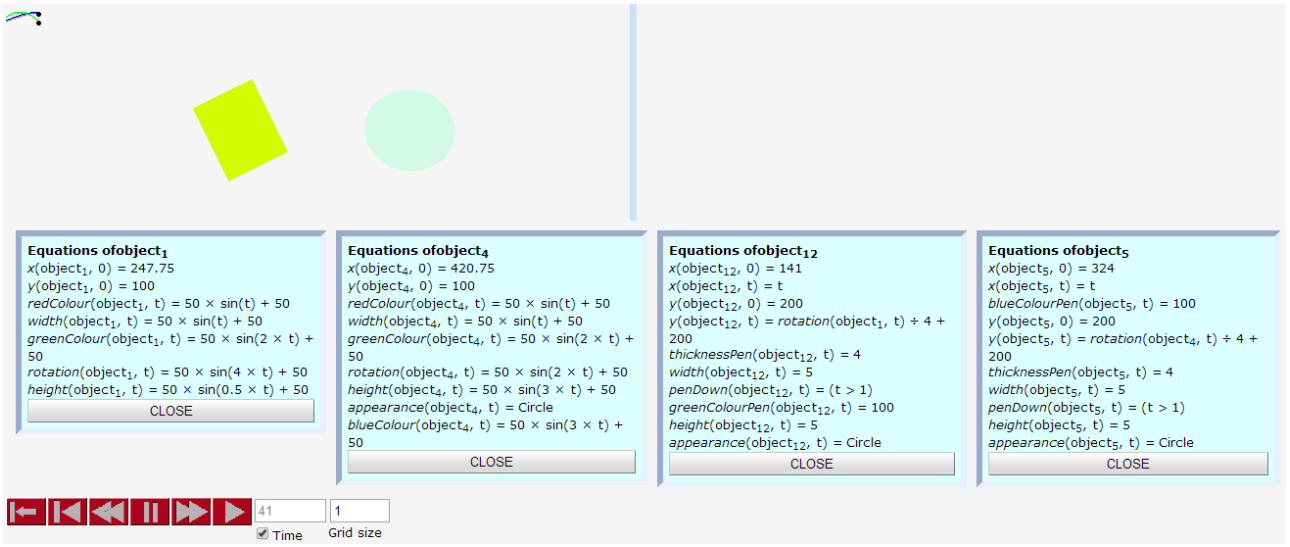

Figure 2 - Two objects changing in a playful manner with two objects graphing them at time 41

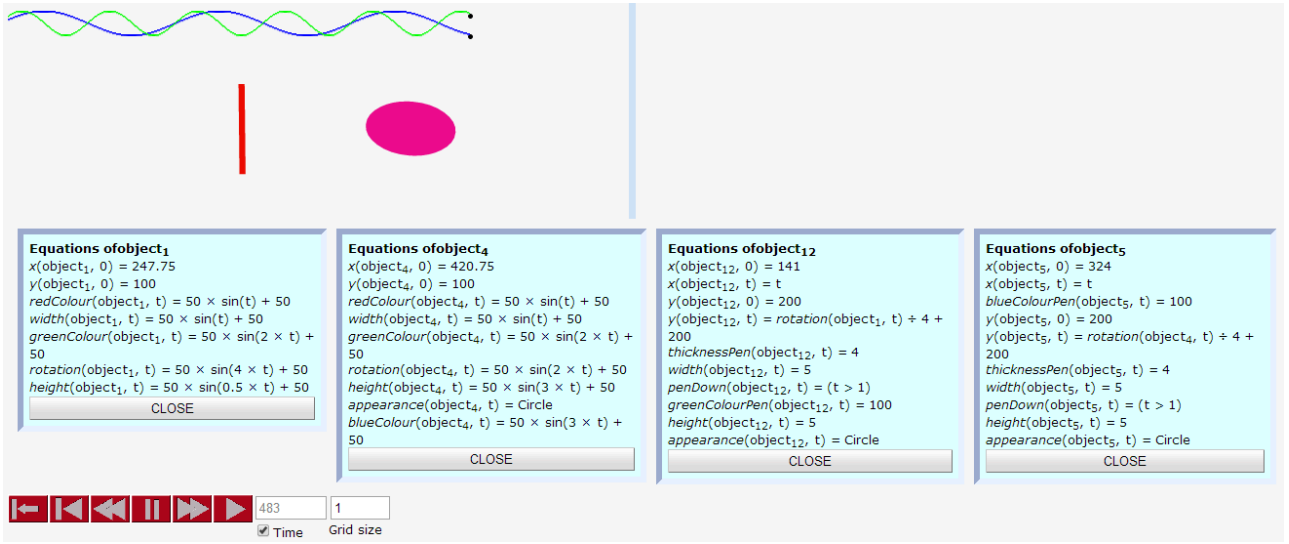

Figure 3 - Two objects changing in a playful manner with two objects graphing them at time 483

The library of equations is a web page that surrounds equations with explanatory rich text and links. Equations display a menu of operations when clicked upon. Third parties can produce other libraries relying only upon HTML and MathML.

An equation to implement (horizontal) acceleration could be defined as

$$x(ME,t) = x(ME,0) + v_x(ME,0) \cdot t + \tfrac{1}{2} \cdot a_x(ME,t) \cdot t^2$$

where ME is a template variable that is replaced by the identifier of the object the equation is associated with.

A more modular and pedagogically useful alternative is to break the equations into a set of difference equations. The following equations also implement acceleration (at integer time values)

$$x(ME,t) = x(ME, t\text{-}1) + v_x(ME,t) \qquad (1)$$

$$v_x(ME,t) = v_x(ME,t\text{-}1) + a_x(ME,t) \qquad (2)$$

The difference equations, unlike the closed form, can be paraphrased so as to be understandable by less-advanced students:

(1) "The (horizontal) position of an object is the sum of its previous position and its current velocity."
(2) "The (horizontal) velocity of an object is the sum of its previous velocity and its current acceleration."

We have begun to explore how to make MoPiX suitable for less mathematically advanced students. An experimental version of MoPiX (available by *adding ?hideEquations=1* to the URL) can display equations as icons, generated natural language, or as conventional mathematical notation, or any combination of these display formats. Students can begin by combining pre-built behaviours displayed as icons and later open them up to see and edit the equations. Equations can live on web pages with supporting rich text including animations of how they work in isolation. When equations are well-known, e.g., the laws of mechanics, they can have links to background resources such as Wikipedia pages. This approach to providing high-level components for modelling scientific phenomena was inspired by the WebLabs Project [10].

Note that the difference equations are also more modular: equation (1) can be used without equation (2). Furthermore, equation (2) can be enhanced to model bouncing while modifying the closed form for bouncing is extremely complicated.

Equations can be defined so that an object's attributes depend upon the attributes of others. In this way composite objects can be constructed where, for example, a hand is constrained to be at the end of an arm. Two objects can interact via equations that reference them both. One student used equations to define two bouncing balls and a square that was always at the average position of the balls. It also left behind a trail. See Figure 4.

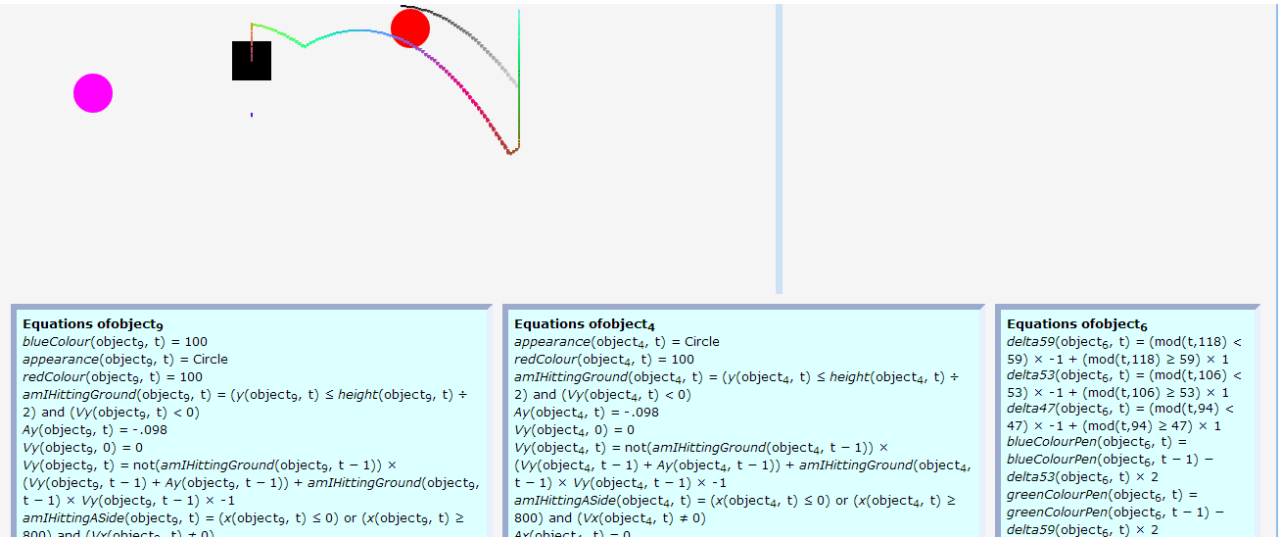

Figure 4 - A square that leaves a colourful trail at the centre of two bouncing balls

4. STRENGTHS AND LIMITATIONS OF EQUATIONS

Equations have many advantages as a means of expressing models:

- Ease of combining things. There are no order dependencies or complex interfaces between components.
- Algebra is an international language so there is minimal need to localize for different languages or cultures.
- Equations (particularly if in a standard format such as MathML) can interoperate with many mathematical tools. For example, an equation solving tool could provide predictions about when two objects will collide.
- Algebra, unlike computer programming, is (already) an important part of the school curriculum.

Equations also have some disadvantages relative to general programming languages for modelling:

- Conditionals and iteration are harder to express. Conditional statements are expressed by multiplying by expressions that evaluated to 0 (false) or 1 (true). An enhanced MoPiX could instead support directly curly bracket conditionals. Iterations not tied to time would require a generalisation of MoPiX where other variables could take the role of *t*.
- State change is represented by variables that have different values at explicitly referenced points in time. In some situations this leads to awkward expressions. It also presents runtime performance challenges to the implementation.

5. SCENARIOS OF USE

Morgan and colleagues used MoPiX in teaching students in a London tertiary college introducing them to a mathematical treatment of motion [5, 6]. In addition to using equations to model various situations they also used MoPiX equations to generate real-time graphs as the simulation ran. Moustaki and colleagues introduced MoPiX to 17-year old students in Athens [7]. These students also focussed upon Newtonian mechanics but in a more playful creative manner. They made shapes change colour depending upon its position and eventually built an interactive juggling game.

6. POSSIBLE DIRECTIONS OF FUTURE RESEARCH

An unexplored area of future research is how to support the debugging of models. Staying within the framework of equations one can monitor or graph any attribute. We have yet to design other tools for debugging. Perhaps a supportive community of users will emerge where the experts will help those with buggy models.

MoPiX is focused upon algebraic equations. The general idea could be applied to using Calculus as an expressive medium. Differential equations could augment the user's expressive vocabulary while creating models.

7. CONCLUSIONS

Our initial experiences confirm that, indeed, algebraic equations can be used as an expressive medium to create animations, simulations, and games. Equations can be combined in creative and surprising ways to achieve ends that matter to children. Perhaps they will acquire a positive attitude towards algebra after experiencing equations as empowering and maybe learn algebra while having fun.

8. AVAILABILITY OF THE SOFTWARE

MoPiX is freely available at http://mopix2.appspot.com. It is implemented in Java and uses GWT (http://www.gwtproject.org/) to translate the client code to JavaScript in order to run in any browser. Source code is under an open source license and is available at https://code.google.com/p/modelling4all/source/browse/#svn%2Ftrunk%2FMoPiX.

9. ACKNOWLEDGMENTS

Dusanka Nikolic was a member of the MoPiX development team. Candia Morgan and Richard Noss helped develop the initial ideas and was involved with activity designs. MoPiX was developed as part of the ReMath – Representing Mathematics with Digital Media project [8], funded by the EU under the Framework 6 programme.

References

[1] M. Guzdial, "Programming Environments for Novices", *CS Education Research*, edited by Fincher, S. Petre, M. 2004.

[2] I. Harel and S. Papert, eds. *Constructionism*. Norwood, 1991, NJ: Ablex.

[3] Y. Kafai, *Minds in Play*, 1995, Lawrence Erlbaum Associates.

[4] K. Kahn, R. Noss, C. Hoyles, and D. Jones, "Designing Digital Technologies for Layered Learning". *Informatics Education, The Bridge between Using and Understanding Computers, Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, 2006: 267-278

[5] C. Morgan and J. Alshwaikh, "Learning about motion in a multi-semiotic environment", Research in Mathematics Education, Volume 11, Issue 1, 2009, Taylor & Francis

[6] C. Morgan and J. Alshwaikh,. "Mathematical activity in a multi-semiotic environment", *Proceedings of CERME 6*, January 2009, Lyon France

[7] F. Moustaki, G. Psycharis, and C. Kynigos. "Making sense of structural aspects of equations by using algebraic-like", *Proceedings of CERME 6*, January 2009, Lyon France

[8] ReMath Project, [online], Available: http://remath.cti.gr

[9] B. Sherin, "A comparison of programming languages and algebraic notation as expressive languages for physics", *International Journal of Computers for Mathematics Learning*. 2001, 6, 1-61

[10] G. Simpson, C. Hoyles, and R. Noss, "Designing a programming-based approach for modelling scientific phenomena", 2005, *Journal of Computer Assisted Learning* 21 (2), 143–15