Designing Digital Technologies for Layered Learning[1]

Ken Kahn[1], Richard Noss[1], Celia Hoyles[1], and Duncan Jones[2]

[1]London Knowledge Lab, Institute of Education, 23-29 Emerald Street,
London WC1N 3QS, UK
{K.Kahn, R.Noss, C.Hoyles}@ioe.ac.uk
http://www.lkl.ac.uk
[2]firbush@hotmail.co.uk

Abstract. Designing digital technologies for deep learning is a highly non-trivial enterprise. In this paper, we discuss one approach we have adopted that seeks to exploit the possibility of affording diverse layers of engagement that exploit the interconnectivity available on the web. In a nutshell, we describe a system that offers learners the possibility of engaging with difficult scientific and mathematical ideas without the necessity of interacting with complex layers of symbolic code – while making that interaction available at all times.

1 Introduction

Our prior research concerning computer games [1] has indicated that games designed for learning authored by students tend not to suffer the difficulty noted with educational games in general around poor production values. In the case of constructed games, even if they are relatively simple and crude, students tend to become dedicated to their creations. However acquiring the skills to make computer games requires a major investment in time and effort. Here, we present some novel design research that will illustrate an approach that aims to reap the advantages of learning by programming games without the disadvantages of having to know *a priori* how to program.

Our starting point derives from the principle that a powerful way for students to learn mathematics is through their long-term engagement in collaborative projects for which they take responsibility individually and collectively (see, for example, [2]). From this basis we have developed the following principles for designing for this learning:

---

[1] This paper is a greatly extended version of Kahn, Noss, Hoyles and Jones (in press): Designing for diversity through web-based layered learning: a prototype space travel games construction kit. Submitted to the ICMI 17 study group on *Technology Revisited*.

- sequenced activities that can engage students at a *variety* of levels in what we name *layered learning*;
- *flexible* tools that have adjustable parameters, can be combined in different way and can also be programmed, and thus allow students to investigate each activity for themselves[2];
- *collaborative interactions* as part of the activity sequence through which students discuss their emerging ideas in the context of their game design;
- *multi-player game playing* either at one computer or over the web.

The second design requirement is worthy of particular attention, as it implies that designers can both empower and constrain students by offering a set of programmable components or modules that can be customised to suit diverse students' goals, as well as tuned to the knowledge domain. Programmable modules have the added advantage of providing a consistent way to combine and modify tools, in the control of the learner.

There have been numerous attempts to design a programming-based approach to learning over the years. The most successful have achieved tangible learning outcomes across various topics (music, mathematics, language, physics (some of these are discussed in [3])). They have also provided important pointers to the possibilities of learning that transcends the procedural and superficial by encouraging a playful – yet mindful – spirit of enquiry on the part of learners, aiming to break down the curricular silos that so often characterise traditional schooling. For the most part, what has been missing has been generalised success in tapping into students' own interests on a wide scale and engaging them in debate, investigation and production. To achieve this aim, we have built and tested a Space Games Construction Kit (SGCK) along with a narrative 'metagame' to assist learners in navigating their way through the sequence of activities of design and game playing. This work can be seen as a continuation of the research we have undertaken within the WebLabs Project [4]. In WebLabs children constructed models by assembling program fragments and then published their models on the project web site. These reports were then read and commented upon by other students who downloaded and ran the models. The programs were constructed in ToonTalk [5], and this enabled students to build programs from scratch. In contrast, in the SGCK the programs were pre-built and are in a dialect of Logo called Imagine Logo [6]. So, whereas the students in the WebLabs project needed to master the programming system (see Harel [7] and Kafai [8] for studies of children engaged with game construction through programming from scratch], those engaged with the SGCK could interact without any programming knowledge if they so desired, and were unable to construct programs from scratch or make major program alterations.

[2] In the current research aiming to involve groups spread geographically it was important that the tools could be accessed through a web browser - although this is not a 'principle'.

2 Description of the Game Environment

The game environment consists of three major components: the game construction kit, the metagame, and usage scenarios. We will describe these in turn.

2.1 The Space Travel Games Construction Kit

We began by considering the classic computer game of *Lunar Lander*. In the process of trying to land upon the moon, a player engages with the laws of motion, playing in a virtual world where the laws of gravity and momentum are not obscured by friction or atmospheric drag as they are on Earth. The Space Games Construction Kit can be used to build a variety of games similar to *Lunar Lander* (see Sherin [9] for a different approach to a similar problem). The construction kit provides small program fragments together with tools for customising and composing them. Thus the software extensions allow *layered* exploration appropriate for populations of students with differing backgrounds and ages; that is, users can choose how far to delve into the workings of the tools.

The innovative aspects of the software include:

1. A child-friendly interface for the composition and parameterisation of pre-built program fragments;
2. An underlying computation model that has been simplified and made composable by building upon multiple independent processes. This is critical for the deeper levels of engagement where students inspect and edit program code;
3. An underlying physics model based upon conservation of momentum that is simpler than the one commonly used based upon first and second derivatives of position and the first derivative of momentum. Yet it is capable of modelling the same phenomena;
4. The concept of a "metagame" where games are made within an overarching narrative game structure.

2.2 The Narrative Metagame

The metagame starts with the player being hired as a game developer at a game company. The player receives help from a team of simulated experts including a programmer, a scientist, a historian, an assistant game designer, and an animator. A player may skip over all this and jump into game making, but the metagame motivates

the scenarios within a project format, as well as providing structure, background information, guidance, and a gradual introduction of new features and capabilities.

The metagame embodies the design of a learning sequence. Learners are presented with a goal and need to interact with their virtual teammates in order to acquire both the required components and the knowledge to proceed. The response of each teammate to a visit by the player is scripted but also depends upon both the current state of the game being constructed by the player and the history of the player's interactions with all the teammates. This gives the player freedom to visit the teammates in any order and with any frequency. Furthermore, each game component has an associated help button. When a component's help button is pressed, the player is informed which teammates have something to say about it. For example, the programmer, the scientist, the historian, and the game designer all have a unique perspective when giving an explanation of the component which implements gravity.

The first task in the metagame is to acquire program fragments and artwork to build a very simple game based upon an astronaut who is adrift and needs to reach her space ship. At first this is accomplished using only components involving horizontal motion. Next the game maker is faced with designing conditions to separate player success and failure. The next task is to give the ship a vertical velocity, and then to add vertical acceleration to the astronaut. By combining the horizontal and vertical components the game elements can programmed to move diagonally. The final game in this sequence involves making the game into a cooperative game between two players over a network. Although the metagame guides the learner along a designed path, he or she is free at any point to invent and play games other than those in the metagame's sequence. To proceed, to the next phase, however, the required game making tasks need to be performed.

The next part of the metagame is to build a Lunar Lander game. Here gravity is introduced. Learners are faced with making decisions such as what speed constitutes a safe landing. Next we introduce player-configurable gauges which help in the task of landing safely on the moon without using too much fuel. The next challenge is to construct the best autopilot program. This is accomplished by recording a good "manual" landing and then editing the parameters of the recorded landing to produce an optimal landing. A two-player version is then introduced where players can compete to land with the safest landing speed using the least amount of time or fuel.

2.3 The Activity Sequence

The students in the first part of our study were given the following written instructions. The second group of students instead relied upon the virtual teammates of the metagame that provided guidance and background science and mathematics.

Phase 1: an exploration of game making.

- *Moving in outer space:* An astronaut is adrift and needs to reach her space ship. Build a game by acquiring program fragments and artwork, which can be accomplished using only components involving horizontal motion that is achieved by 'throwing' rocks (previously collected by the astronaut).

- *Agreeing upon some constraints:* for example, will the astronaut get back safely? Is she going too fast when she hits the spaceship?
- *Creating a new game:* Now invent a new game. For example, the space ship has started to move off in a vertical direction. Can the astronaut reach it now? The final challenge in this sequence involves making the game into a cooperative game between two players over a network.

Phase 2: Building and playing a Lunar Lander game. To proceed to this phase, all the required game-making tasks in Phase 1 need to be performed. Here gravity is introduced.

- *Landing on the Moon:* for example, what speed constitutes a safe landing? At this point we introduce dynamic configurable gauges to measure speed, acceleration, mass and other parameters which monitor and graph these values and can help in the task of landing safely on the moon. Added challenges are introduced: e.g. do not use too much fuel!
- *Using the Autopilot:* The settings of a 'manual' landing consists of all the variables and how they are changed can be captured by an autopilot. The next challenge is to construct the best autopilot program, by recording what is deemed a good landing and then tweaking the parameters of the recorded landing to produce an optimal landing.
- *Multiplayer game over the net:* Players can compete for example to land with the safest landing speed using the least amount of time or fuel. Students are also challenged to invent new games to play.

3 The Potential Layers of Learning

Both the metagame and the construction kit were designed to support *layered learning*. At the first layer, engagement is mainly through reading, watching and making conjectures based on observation and for example describing a motion and reflecting on it. The tools for this layer are basic, perhaps only a handful to control a simulation on video, or the timing of movement. At a second layer, the learner can begin to manipulate behaviours that define various sorts of motion and predict and test out the effects of different values. At a third layer, the learner might explore further how variables relate to each other, for example position, velocity and acceleration, by reference to the values set by sliders. And finally, a fourth layer that engages with these relationships either by modifying existing programming code or by writing new programs or fragments of programs.

In a follow-up study we provided 'behaviour gadgets' that support rotation, grasping, releasing, and springs. Using these building blocks, students can build a detailed model of the astronaut in space rotating her arm while holding a rock and then releasing it. This enables the students to take the act of throwing, that previously was an atomic action, and explore a deeper layer involving additional concepts such

as angular momentum. Students can delve into deeper layers of learning in different domains such as physics, mathematics, computation, or game design.

4 Game Construction: the Interface

When the player is ready to build the game, a game panel with images of the astronaut and lander is presented (see Figure 1). Beside it is a control panel with a start button. Pushing the start button initially does nothing, since the game elements have yet to be given programs. The control panel also has a button that causes the *behaviour gadgets panel* to appear (Figure 2). It contains behaviour gadgets that consist of one or more *code boxes*. A picture can be given a behaviour by placing a behaviour gadget on its back. The behaviour can be altered by setting sliders on the gadget's *settings* page. The code boxes of a behaviour gadget can be removed, whereupon they expand to display the code that implements the behaviour. Portions of the code that can safely be edited without programming expertise are colour highlighted.

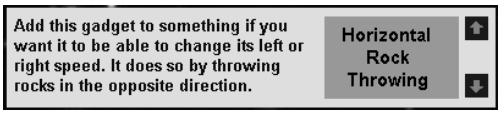

Fig. 2. Behaviour gadgets can be dragged from the Behaviour Gadgets Panel. Initially, the behaviour gadgets panel contains only a horizontal velocity gadget and a horizontal rock throwing gadget. As the metagame progresses, the behaviour gadgets panel acquires more elements

Fig. 1. The initial game construction page

A behaviour gadget contains buttons for setting parameters and obtaining help from the virtual teammates. It contains at least one code fragment. Removing a code fragment from a behaviour gadget causes it to expand to full size as illustrated in Fig. 3.

```
HORIZONTAL VELOCITY CODE
; The following causes my position to be updated by my velocity.
; It updates 100 times a second by 1/100th of the velocity.
; The velocity is how much it moves in a second.
; This code only applies to motion to the left or right.
repeat_every 1/100
  [change_my_horizontal_position_by
    my_horizontal_velocity/100]
```

Fig. 3. The Horizontal Velocity Code Box after removed from the Behaviour Gadget

As a layered learning design, our software includes code fragments that students may, but need not, read and edit. To encourage delving deeper, some behaviours

cannot be altered by moving sliders. Instead, the student is encouraged to change the code to improve the game they are making. One example of this is the code box for running out of air, which is a part of the reached or missed ship gadget. The game is too easy (and boring) if one can very slowly drift back to the ship. Running out of air introduced an intrinsic time limit.

The total mass of rocks (i.e., total fuel), the largest rock (the maximum rate of fuel usage), and the rock velocity (the propellant velocity) can all be adjusted by moving sliders. As one does so, the system makes calculations to show derived values such as force and to perform unit conversions. These parameters reflect real engineering tradeoffs. For example, adding more rocks/fuel does increases the duration of manoeuvrability but at the cost of a greater total mass and hence a smaller acceleration from identical rock throws.

Limitations of space prevent us from illustrating a range of further panels, behaviours and other objects that control instrumentation (for example, gauges that monitor any of 13 values in graphical or numerical displays, including velocity, acceleration, remaining fuel, total mass, the application of thrust (by throwing rocks out in the opposite direction), autopilot facility (a recording of all the changes to thrusters made manually), and a two-player version of the Lunar Lander game typically involving a race to be the first to land safely on the moon.

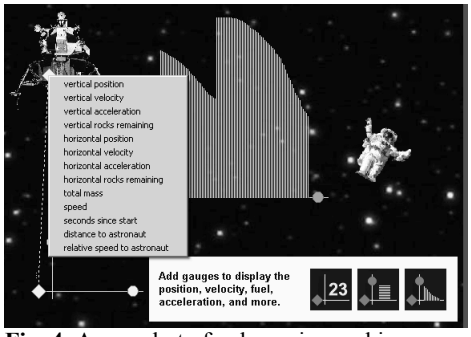

Fig. 4. A snapshot of a dynamic graphing gauge displaying the vertical position of the astronaut as she falls and the connection of a second gauge to attributes of the lander

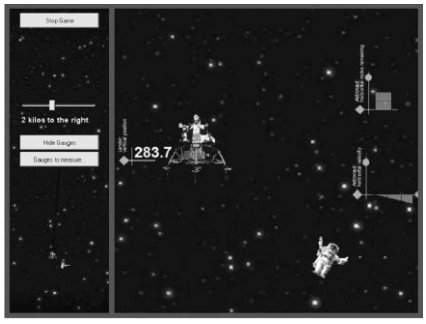

Fig. 5. A snapshot of game play with three active gauges

5 Some Illustrations of Learning

We now very briefly outline some of the learning issues that are emerging from the iterative design/test cycle with three groups of students: two drawn from a large, urban comprehensive school (one "Year 7" class aged 11-12; one "Year 8" class, aged 12-13) and a small group of 3 students (aged 12-14) from a second school in an after-school setting. We should stress that we are in the very early stages of working with students, and that up to now, we have considered students (and their teachers) as primarily co-designers of the emerging system, rather than as 'subjects' with whom to evaluate learning. The learning issues we outline here, therefore, should be regarded

as a tentative set of issues for further exploration, rather than a definitive list of 'learning outcomes'.

Developing understandings of Newton's third law: In the first task the Year 7 students began with a relatively basic knowledge of the physics concepts involved. For example, R suggested, "you could throw some rocks away and that would make her lighter so she would move." She was apparently unaware that the effect of gravity in space is negligible (in the setting in which the game takes place). Through the course of the activity and experimentation with the horizontal rock thrower R and her coworkers appeared to develop some appreciation that throwing a rock would develop an opposite movement proportional in velocity. Indeed, later in the session two of the students worked with the theory that "throwing larger rocks makes her move faster."

Minimising time (of astronaut to spaceship, or lander to moon) proved a motivating task, particularly for the Year 7 students. Most students used an iterative strategy, e.g., Tom and Alex were delighted to refine their strategy again and again by optimising the use of the horizontal rock thrower against the speed of reaching the spaceship.

Using the gauges: Throughout the sequence much use was made of the gauges and interpretation of their output. Students found the gauges relatively easy to put in place and interpret and tended to refer to them constantly as a guide to their use of the rock thrower or thrust. When landing on the moon some students applied far too much thrust causing the lander to move upwards and disappear off the screen. Reading the vertical velocity gauge which they had set up for the lander they predicted how the lander would "keep on getting slower until zero. Then it will fall back again because of gravity." In the two-player game the ability to attach gauges to the opponent's lander was a particularly successful feature, enabling one group to make a close comparison with the other and to adjust the strategy second by second.

Composing horizontal and vertical velocities: Coming up with the hypothesis that to achieve diagonal movement a combination of horizontal and vertical thrusts would be needed, appeared surprisingly 'obvious' to the students and was tested by, for example, using both horizontal and vertical rock throwers to the astronaut and using both simultaneously.

Gravity: None of the Year 7 students immediately made a connection between the rock throwing astronaut and the rock throwers for the lander, although this was evident for all of the Year 8 students. The Year 7s also only had a sketchier concept of gravity. Only two – Tess and Alex – volunteered that the lander game would be different from the astronaut in that there would be a gravitational pull near the surface of the moon.

Collaboration, competition and motivation: Beating previous best scores proved highly motivating, especially for the team of Year 7 boys. Collaboration centred around agreeing what the two teams should have in common; the total mass of the projectiles, an agreed safe landing action, a value for gravity and the vertical starting position of their landers, for which they sought and found a new gauge, previously not used. Attempts to minimise fuel use became more sophisticated. The boys realised that with their agreed safe landing speed of 30 metres per second, they needed only to keep just below this figure to ensure a safe landing and minimal fuel consumption. Before this they had been trying to reduce the velocity to the minimum regardless of fuel use.

In summary, the competitive element of the two-player version appeared as a considerable motivation to the students: they enjoyed seeing the opposition's ship on their screen and being able to monitor its progress through gauges.

6 Reflections on Design

Overall the software did allow access to diverse students at many layers of learning, it stimulated huge interest and discussion (both inside and outside classrooms) and students used quite sophisticated ideas in pursuit of their game making and playing. The students came up with many ideas for improvement. For example, they suggested the need to reduce the amount of reading required in the initial stages and simplify some terminology that was too complex in places. Further suggestions included that a choice should be offered between reading and listening to instructions and that more complex instructions might be communicated through demo buttons or tutorials – for example showing them how to set up a gauge or to use a behaviour gadget. Perhaps the most interesting suggestion was that the software might be structured into what some students described as "levels", such as those commonly found in computer games.

Where the lunar lander software failed to meet the initial expectations was in giving students easy access to the programming code and in creating situations where they would want to analyse and adjust that code. It would seem possible that if the software were remodelled into a series of levels – corresponding in some way to the previously defined expected layered learning model – analysis and use of the relationships inspectable in the programming code or as recorded by the autopilot could become not just a real possibility but an integral part of the game.

When designing a toolkit to be used to construct scientific models (and games based upon these models) one needs to determine the underlying ontology of the system. How should time be modelled? How should concurrent processes behave? What are the primitive notions of motion and space and which concepts are to be constructed from those primitives? Are the usual ways of conceptualising the laws of motion that are based upon algebra and calculus optimal for computational modelling?

We chose to build upon a discrete conception of time and forces, by providing behaviours that create independent simultaneous computational processes that implement horizontal and vertical velocity, horizontal and vertical thrust, and gravity.

We created new computational primitives upon which everything else rests. One is called *repeat_every*. Its first argument defines the frequency with which the second argument is executed. For example, constant gravity near the surface of the moon is implemented by the following code fragment:

```
repeat_every 1/100
     [change_my_vertical_velocity_by
            value_of_slider_for_gravity/100]
```

This causes the vertical velocity of the associated object to be increased by the value of gravity (or decreased if gravity is negative as it usually is). To approximate the continuous change of velocity this program is run every 1/100th of a second. The concept of a sampling rate is relied upon here.

Note that simultaneous with the execution of the gravity code fragment there can be other processes that are also incrementing or decrementing the vertical velocity, perhaps due to the use of thrusters (or rock throwing). The relative ordering of the execution of these processes on a sequential computer will have no material effects.

We chose to make the notion of position be the only computational primitive for modelling motion. So when the code implementing the vertical velocity runs, it relies only upon this primitive to change the vertical position by the product of the current vertical velocity and the elapsed time (1/100th of a second in this example).

```
repeat_every 1/100
     [change_my_vertical_position_by
       my_vertical_velocity/100]
```

Alternatively, we could have chosen velocity as the only primitive notion. Then the updating of position would still be a function of the current velocity but in an opaque manner. If both position and velocity were primitive notions, then there could be confusion if both were used as control variables. When any part of the program changes the position, should the velocity change accordingly? Does velocity measure something or specify something? Doing both can lead to confusion. It also hides the mechanism relating them.

While position is the computational primitive underlying the construction kit, it is velocity that is the physical primitive. It is implemented in terms of position and time but other behaviour gadgets are expressed in terms of changes in velocity. It is interesting to note that perhaps speed (velocity without concern for direction) is the human psychological primitive. Piaget, for example, suggested that speed is primitive and that time is not, but instead understood in terms of speed [10].

Another epistemological question is how to model forces. Should forces define acceleration, which in turn defines velocity, which defines position? A sequential program that models all the processes could be built in this way, but it would lack the modularity and composibility of the concurrent processes on which we rely. Consider the difficulties that would arise if one process implemented gravity by setting the acceleration to the appropriate value, while a thruster process implemented force by adding to or subtracting from the current acceleration. Clearly, the order in which the gravity process and the thruster processes run will drastically affect the model.

One reason we chose to build upon the conservation of momentum, rather than F = ma, is that it provides a concrete and discrete way to think about forces. We suggest that this approach is likely to be more accessible than the alternative, which relies upon a notion of continuous rates of change. It also makes the mechanism underlying rocket thrust transparent. Throwing a one kilogram rock once per second is the same mechanism as real rocket thrusters that “throw” a trillion trillion molecules (“rocks”) per second. Force is the derivative of momentum and as such seems a more complex and difficult notion than the discrete change in momentum that underlay our approach.

Our de-emphasis of the concept of force is consistent with modern physics. Nobel laureate Frank Wilczek, recently wrote in "Whence the Force of F = ma?" [11]:

> … the concept of force is conspicuously absent from our most advanced formulations of the basic laws. It doesn't appear in Schrödinger's equation, or in any reasonable formulation of quantum field theory, or in the foundations of general relativity…
>
> … If F = ma is formally empty, microscopically obscure, and maybe even morally suspect, what's the source of its undeniable power?

Our perspective on forces is closer to diSessa's concept of momentum flow [12]. Momentum flow is an alternate way to conceptualise elementary mechanics where one considers momentum as something that acts like a conserved fluid that flows within and between objects. Formally, it models the same phenomena as the framework based upon modelling forces and accelerations. Pedagogically, it leads to a promising alternative way of thinking about momentum and forces.

The prototype demonstrates the promise of these ideas in learning Newtonian mechanics. Currently, we have sketched a plan for how the current prototype could be extended to include universal gravity (i.e. orbital mechanics) and atmospheric drag. In a subsequent project we are exploring other aspects of mechanics such as the conservation of angular momentum. Clearly these ideas can be applied in many other areas in physics (e.g. special relativity). Some areas, however, such as quantum physics, are particularly challenging.

Games based upon ecology and animal behaviour fit this framework well. Behaviour gadgets could give the elements of the game the ability to hunt or evade predators, eat or starve, reproduce and so on. These behaviours could be customised and combined to create a wide variety of animal behaviours and ecologies. [4]

It is plausible that the range of phenomena modelled by the NetLogo [13] and StarLogo [14] communities can fit into our framework. By doing so they would benefit from the structure and guidance of metagames for making games and the support for component composition and parameterisation (behaviour gadgets). It would be interesting to explore whether many popular commercial games such as *Civilization*, *SimCity*, or *The Sims* could inspire game making games for learning about history, government, or psychology.

7 Future Research Directions

In addition to more systematic and varied field studies of student use of the Space Travel Games Construction Kit, we have considered about twenty enhancements to the construction kit and the metagame. We also designed several other mini space travel games that could be strung together with the ones described here to provide a richer and deeper experience. One promising direction for future research is to explore monitoring or debugging tools to help students understand the execution of their programs in detail.

Acknowledgement

We acknowledge the funding of the BBC, and the helpful comments of colleagues in the London Knowledge Lab, notably of Gordon Simpson and Diana Laurillard. Ivan Kalas designed and implemented the user configurable gauges. Peter Tomcsanyi and Ivan provided invaluable advice and technical support.

References

1. Noss, R. & Hoyles, C.: Exploring Mathematics through Construction and Collaboration. In K.R. Sawyer (ed.) *Cambridge handbook of the Learning Sciences,* CUP, Cambridge. (in press)
2. Harel, I., & Papert, S. (eds.): Constructionism. Norwood, Ablex Publishing Corporation, New Jersey (1991)
3. Noss, R. & Hoyles, C.: Windows on mathematical meanings: learning cultures and computers. Kluwer, Dordrecht (1996)
4. Simpson, G., Hoyles, C. and Noss, R.: Exploring the mathematics of motion through construction and collaboration, Journal of Computer Assisted Learning 22:2 (2006) 114-136
5. Kahn, K.: ToonTalk: An Animated Programming Environment for Children, Journal of Visual Languages and Computing, 7(2) (1996) 197—217
6. Blaho, A., Kalas, I.: Object Metaphor Helps Create Simple Logo Projects. Proceedings of EuroLogo, University of Sofia, Bulgaria (2001)
7. Harel, I.: "Children as software designers: a constructionist approach for learning mathematics", The Journal of Mathematical Behavior, 9 (1) 4 (1990) 3-93
8. Kafai, Y.: Children as designers, testers, and evaluators of educational software, The design of children's technology, Morgan Kaufmann Publishers Inc., San Francisco, CA (1998)
9. Sherin, B.: A comparison of programming languages and algebraic notation as expressive languages for physics, International Journal of Computers for Mathematics Learning, (2001) 1-61
10. Piaget, J.: The child's conception of time, (Translation: Pomerans, A), New York, Basic Books (1969)
11. Wilczek, F.: Whence the Force of F=ma? I: Culture Shock, Physics Today, October, (2004)
12. diSessa, A.: Momentum flow as an alternative perspective in elementary mechanics, American Journal of Physics, 48, (1980) 365-369
13. Wilensky, U.: NetLogo, http://ccl.northwestern.edu/netlogo, Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL (1999)
14. Resnick, M.: Turtles, Termites, and Traffic Jams: Explorations in Massively Parallel Microworlds, Cambridge, MA, MIT Press, (1994)