# *Mindstorms* Revisited: Making New Construals of Seymour Papert's Legacy

Meurig Beynon

Computer Science, University of Warwick, Coventry CV4 7AL, UK

**Abstract.** Seymour Papert's *Mindstorms* is a seminal text in educational technology. Its subtitle: *Children, Computers and Powerful Ideas* reflects Papert's broad visionary ambition, yet the cover of its second edition is headlined: *ALL ABOUT LOGO - HOW IT WAS INVENTED AND HOW IT WORKS*. Notwithstanding the enormous practical impact of LOGO on educational applications of computers, we should not forget Papert's declaration regarding the computer-inspired revolution in learning he envisioned: "I do not present LOGO environments as my proposal for this ... [they are] too primitive ... too limited by the technology of the 1970s ...". This paper revisits the challenge implicit in this declaration, appraising the extent to which today's technological advances can realise Papert's vision, and proposing an alternative model for his central conception of "an object-to-think-with, that will contribute to the essentially social process of constructing the education of the future".

**Keywords:** Seymour Papert, Mindstorms, constructionism, making construals, LOGO programming, collaborative learning.

## 1 Introduction

Seymour Papert's *Mindstorms* [13], first published in 1980, made a seminal contribution to the field of educational technology. It stimulated research into the potential role of computer in learning that has embraced programming, microworlds and educational robotics. The spirit of *Mindstorms* is well-represented in the principal themes of the Edurobotics conference: how, by building on Piaget's pedagogical theory of constructivism and Papert's principles of constructionism, contemporary technologies such as digital fabrication and robotics can promote skills in creative thinking, collaboration, and problem solving in Science, Technology, Engineering, Arts and Math (STEAM).

*Mindstorms* is a visionary book. In the Introduction, Papert envisaged how 'ubiquitous computing' and 'an increasing disillusion with traditional education' can 'come together in a way that would be good for children, for parents and for learning ... through the construction of educationally powerful computational environments that will provide alternatives to traditional classrooms and traditional instruction'. It might be argued that these prophesies are now being realised in a culture in which 'apps' and 'MOOCs' abound. In a recent blog entitled "How to Teach Computational Thinking" [22], Stephen Wolfram makes the case for the Wolfram Language as a

transformative tool that makes computational thinking readily accessible and in the process opens up new teaching strategies across the STEAM disciplines. In a similar spirit, the *Maker Movement Manifesto* [6], promotes 'making' as an activity accessible to all, observing that: "The tools of making have never been cheaper, easier to use, or more powerful." ... "It may take some practice to get good at some kinds of making, but technology has begun to make creating easy enough that everyone can make". The implicit message is that modern technologies and practices are bringing Papert's vision into reality.

This paper reflects scepticism about such claims and questions whether – despite the practical progress that has been made towards exploiting computers in education in recent years – Papert's agenda in *Mindstorms* has yet been properly addressed. In truth, Papert's vision in *Mindstorms* went beyond realising new educational practices based on new technologies. His primary goal was a deeper understanding of how building with computers might contribute to learning that was based on key ideas set out by Piaget [15]. When Papert asserts [13:p11] that he sees the Turtle as "a valuable educational object [whose] principal role here is to serve as a model for other objects, yet to be invented" and that his "interest is in the process of invention of 'objects-to-think-with', objects in which there is an intersection of cultural presence, embedded knowiedge, and the possibility for personal identification", he is not merely making a disclaimer regarding his advocacy of LOGO. He is posing a challenging fundamental question about the relationship between computer-based activities and learning. 'Inventing an object-to-think-with' is not the same thing as 'writing a computer program' or 'fabricating a digital object'. Papert is not primarily concerned with making it easy to frame a computation or create a digital artifact; his interest is in the way in which learning is associated with these and other processes of construction. He proposes to exploit the computer in devising objects-to-think-with as a means to address this meta-agenda. When discussing "LOGO's Roots" in Chapter 7 of *Mindstorms*, Papert's focus is not on the qualities of LOGO as a programming language, but on how far it provokes learners to reflect on the basis for their knowledge and so enables the computer to liberate 'epistemological aspects of Piaget's thought' [13:p156]

The three main sections of this paper examine Papert's agenda in more detail with reference to motifs that run through *Mindstorms*. The first reviews Papert's proposals for educational use of computers from a 'computational thinking' [20] perspective. The second discusses Papert's pedagogical perspective in conjunction with reflections by the psychologist Charles Crook on the role of computers in collaborative learning and identifies perceptions of the challenges involved that they hold in common. The third relates *Mindstorms* to an approach, developed by a group of researchers working under the guidance of the author over many years, that aspires to a broader vision for computing than computational thinking affords. Its key concept, that of 'making construals', is well-matched to the core challenges identified by Papert and Crook.

## 2 *Mindstorms* from a Computational Thinking perspective

One of the most startling sentences in *Mindstorms* [13:p12] reads: "Readers who have never seen an interactive computer display might find it hard to imagine where this can lead." It reminds us what foresight Papert showed in relation to the technologies of the 1970s. It also highlights a fundamental shift in perspective on computing that is easily overlooked in hindsight.

The computational foundation for computing was established long before it was appropriate to view the computer and its associated technologies as a coherent source of experience. Computation in its strict mathematical sense is an abstract concept whose relationship to experience is indirect. Most of the algorithmic activity that goes on inside the computers around us cannot be directly experienced – only a minute part of their state-changing activity is manifest at the surface through interface devices. What is so radical and far-sighted in Papert's treatment of the computer in *Mindstorms* is that it is focused on the direct experience that computing technology enables. This emphasis on the 'computer' as a concrete specific physical artifact rather than on 'computation' is flagged explicitly in its subtitle. It is a perspective that sets our experience of the computer in the same context as our everyday experience of any other object. In this respect, it differs from addressing the computer as a technical device, as an engineer might, as Papert's deprecation of BASIC as a programming language for "the computerists" [13:p35] reflects.

For Papert, the semantic frame of interest is much broader than a conventional computational model can support. His stance on knowledge encompasses the process by which consensus is reached from personal understanding. By way of illustration, a paper on epistemological pluralism, co-authored by Sherry Turkle [17], describes how learners aspire to interact with programming languages and robots in ways far outside the scope of what their designers intended: "Lisa, 18, a first-year Harvard student in an introductory programming course … wants to manipulate computer language the way she works with words as she writes a poem." … "[Alex, 9] turns the Lego wheels on their sides to make flat "shoes" for his robot and harnesses one of the motor's most concrete features: the fact that it vibrates" … to make it "travel". Turkle and Papert allude to the idea that "Computers provide a context for the development of concrete thinking" but that "The practice of computing provides support for a pluralism that is denied by its social construction". Equally relevant is the fact that the traditional conceptual framework for computing is ill-suited to thinking of the computer in concrete terms as 'a source of experience'.

In *Mindstorms* and *The Children's Machine* [14], Papert identifies many of the considerations beyond the scope of traditional models of the computer that might enable us to see computers "as providing contexts for the development of concrete thinking". When he contends that "[computers] should serve children as instruments to work and think with" [14:p168], he proposes an engagement between a person and a computer resembling that between a scientist and musician and their instrument in which behaviours are crafted responsively moment-by-moment. When referring to "externalising intuitive expectations" [13:p145], to "thinking [that] is always vaguely right and vaguely wrong at the same time" [14:p167], and tolerance towards "false theories" [13:p132], he invokes contexts in which pragmatic human judgement has an essential role. Though such interactions and interpretations play a significant part in

many practical applications of computing in the modern world, it is to say the least problematic to accommodate them within the rule-based system of thought that is so prominent in computer science.

Since the publication of *Mindstorms*, there have been many attempts to bridge the gap between the computational foundation of computing and the computer as a source of concrete experience. For obvious reasons, the thrust in software development practices has been towards this objective. LOGO itself has had myriad incarnations [1] that reflect such trends in software development, including the development of object-oriented and multi-agent variants such as Imagine LOGO and NetLogo. Three contemporary developments are illustrative of different approaches that have taken inspiration from *Mindstorms*: the SCRATCH environment [30], the Wolfram Language [22] and Bret Victor's *Learnable Programming* [19].

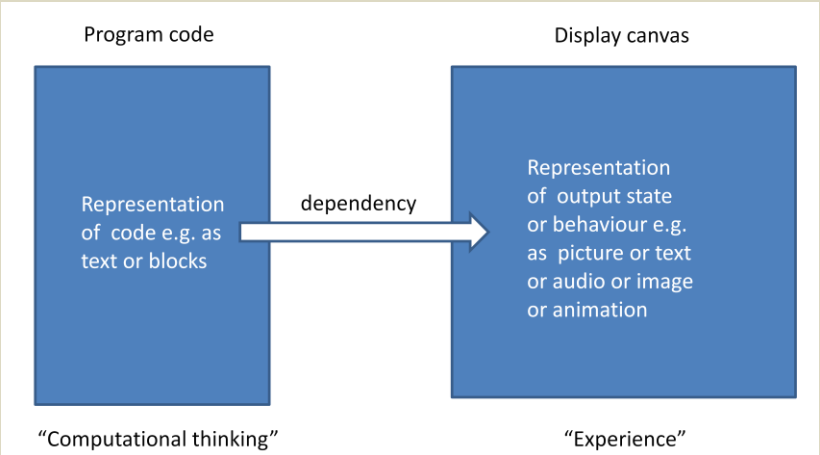


**Figure 1: From computation to experience**

In their different ways, SCRATCH, the Wolfram Language and Learnable Programming are concerned with making connections between 'program code' and the experience of the programmer / learner. Figure 1 depicts the generic framework within which the most elementary applications of all three may be conceived. Associating 'computation' with 'experience' is an idea that would be out of place in a formal approach to giving semantics to interaction with the computer. The 'dependency' between the program code and the display canvas refers to the pragmatic connection that the learner perceives between 'changes to the code' and associated 'changes to the ouput'. The presumption is that this connection, which is to be established primarily by modifying the program code and observing the effect, can be crafted in such a way that it can be reliably learnt.

In SCRATCH [30], the program code is represented visually by blocks that can be snapped together to form stacks of instructions for execution. The simplest blocks represent primitive instructions of the kind that are familiar from LOGO, but much more sophisticated instructions, such as might control a video for instance, are also

available. The behaviour associated with the program code is animated on the display canvas.

In the Wolfram Language [22], the program code takes the form of functional expressions using a range of powerful preprogrammed operators and the display canvas takes the form of a traditional output window in which the values of these expressions (which may have rich spatial and temporal characteristics) are displayed. A functional expression illustrated in [22] simultaneously displays maps of the vicinity of the Eiffel Tower at various different scales, for example.

Victor's proposals for Learnable Programming [19] highlight the subtlety of the considerations that determine how effectively the correspondence in Figure 1 can be applied in learning. Victor's work gives key insights into the way in which this correspondence functions. He shows how, even in crafting behaviours, the correspondence in Figure 1 has to be first apprehended as a connection 'in immediate experience'. He also highlights the fact that this correspondence can be invoked in two contexts that would be deemed quite distinct when considering program code in a conventional perspective: in 'designing the program' and in 'simulating its execution'. He also shows how the correspondence in Figure 1 is intimately tied up with the intended informal meanings of variables. On this basis, Victor deprecates environments for learning programming (such as [10]) in which the learner is encouraged to experiment with unlabelled variables in order to identify their significance.

Excellent illustrations of these principles can be found in Victor's talk on "Inventing on Principle" [18]. In one demonstration, Victor shows how an image of a tree with blossom can be manipulated through live editing of its script. In another, he shows how the motion of a cartoon figure can be specified using a similar technique. In both cases, the counterpart of the 'program code' in Figure 1 is a JavaScript program whose 'display canvas' realisation takes the form of an associated webpage.

SCRATCH and the Wolfram Language can both be seen as illustrating the power that new technology can potentially bring to Papert's vision. The world-wide dissemination of the SCRATCH culture might be taken as vindication of the idea that learners can use the computer to share and express their ideas as envisaged in *Mindstorms*. Wolfram's striking illustrative examples of interdisciplinary applications of computational thinking demonstrate Papert's power principle [13:p54]: "[the computer] must empower the learner to perform personally meaningful projects that could not be done without it". Wolfram's strategy of supplying the learner with a catalogue containing several thousand meaningful functions whose significance is to some degree implicit in their name (cf. [21]) brings to mind Papert's notion of learning to program as like learning a natural language. But Wolfram's critique of LOGO and SCRATCH ("while the turtle (or cat) is quite cute (and an impressive idea for the 1960s), it seems disappointingly narrow from the point of view of today's understanding and experience of computation" [22]), is two-edged. On the one hand, it reflects a valid contemporary concern about the expressive limitations of SCRATCH. On the other, it suggests a shallow appreciation of the deeper motivations behind the invention of the Turtle as a pedagogical device. And though the Wolfram Language includes pre-constructed functions of unprecedented expressive power, it is unclear to what extent it brings new insight to the processes by which such functions are constructed – a central epistemological focus of attention in *Mindstorms*. On this

basis, it seems unlikely that either of these approaches to programming is delivering all that Papert would expect of objects-to-think-with.

Victor's experimental work goes much further in attempting to link programming to experiential roots and in the process discloses challenging hitherto unexplored issues. His discussions highlight the potential for developing the embryonic framework depicted in Figure 1 to advance Papert's core agenda significantly. Victor is discriminating in his invocation of empirical processes in programming, insisting that the context for the correspondence in Figure 1 is critical. Since he considers this correspondence as it relates to correlating program code with state rather than simply with behaviour, he enhances the scope for expressing both declarative and procedural knowledge. This has important implications for conveying designs as well as executable programs. It also helps to refine concepts that Papert introduced casually without adequate explanation. An experienced teacher familiar with the form that debugging typically takes in the initial stages of learning will have reservations about Papert's observation that "Experience with computer programming leads children more effectively than any other activity to 'believe in' debugging" [13:p114]. Victor is able to make a distinction between 'debugging' as it often presents itself to the novice in conventional programming context as an uncomfortable encounter with meaningless state and 'debugging' as taking a meaningful step towards new understanding of the domain or application. Where Papert conceives associating LOGO commands with physical actions as 'learning the language of the Turtle', the introduction of declarative elements gives scope for more versatile and expressive connection with natural language. Thinking of programming in the setting of Figure 1 also helps to appreciate why it may be necessary to look beyond 'program code' as a metaphor when developing objects-to-think-with. Figure 1 can hardly be applied to Lisa and Alex's unconventional perceptions of what the computer might be able to deliver [17], for instance. The notion of "thinking [that] is always right and vaguely wrong at the same time" [14:167] as it might apply to 'writing a poem' implies an ambiguity of interpretation that is outside the scope of program code. Likewise, it is hardly conceivable that a Lego robot with vibrating flat wheels would follow a trajectory that could be reliably associated with a specific piece of program code. The extent to which Papert's vision transcended a 'computational thinking' paradigm is the theme of the next section.

## 3 *Mindstorms* from a pedagogical perspective

For Papert, the fundamental motivation behind using the computer to create 'objects-to-think-with' is a better understanding of learning. He attributes his theory of learning to aspects of Piaget's work that have been underplayed. Papert identifies this as a 'knowledge-based' rather than a deductive theory [13:p166] that draws upon Piaget's epistemological studies of how children develop knowledge.

In the Preface to *Mindstorms* and his discussion of LOGO's roots in Piaget and AI [13:Chapter 7], Papert describes two ways in which the computer can help to build on Piaget's work. Referring to his childhood experience of playing with gear mechanisms, and the positive impact this subsequently had on his interest in algebra,

Papert sees computer technology as a way of implementing similar models that promote learning. In this context, Papert stresses the 'affective' component, complementary to the cognitive aspects studied by Piaget, which this can bring to a child's assimilation of knowledge. They may develop an enthusiasm for geometric exploration from working with LOGO, for instance. A second way in which computer technology can be applied is in attempting to construct "machines to perform functions that would be considered intelligent if performed by people" [13:p157] in the spirit of AI.

In distinguishing his approach from a deductive theory, Papert refers to his work with Marvin Minsky on *The Society of Mind* [12]: "the subjective experience of knowledge is more similar to the chaos and controversy of competing agents than to the certitude and orderliness of p's implying q's" [:p172]. Though Papert refers to the possibility of making computational models that can to some degree emulate a child's subjective understanding (as in his discussion of how a child may come to understand that the same quantity of liquid may be held in containers of different height [13:p167]), such implementation is to be viewed as a way of obliging us to think deeply about the processes that inform human intelligence and thereby gaining insight. In a similar spirit, Papert is not concerned with using the computer simply as a way of illustrating an established theory, observing that Piaget's work calls into question the idea that teaching a child the "correct" theory is superior as a learning strategy [13:p133]. Papert's lack of interest in starting from the kind of comprehensive understanding of the agency in a situation that is presumed when framing a computational process is what distinguishes his perspective from that of a professional programmer. It is the pragmatic exploratory activities that may lead the learner to such an understanding that motivated Papert, and, where the computer plays a role in developing an object-to-think-with, the development is – at least in aspiration – more closely aligned with *bricolage* [14:p143-146]. That is to say, its guiding principle is the crafting of experiences by whatever means come to hand.

Making connections between different aspects of their personal experience is central to Piaget's and Papert's notion of how children learn. Most importantly, these are of their essence personal connections: cf. Papert's account of "encouraging [Debbie] to make connections between different elements of what she already knows ... they have to be *her* connections" [14:p165]. Though *Mindstorms* makes no explicit reference to William James's theory of knowledge [9], here, and elsewhere, there is synergy with the Jamesian notion that all knowing is rooted in personal connections in experience that are themselves experienced. It is to associations of this nature that Papert alludes when he asserts that "learning to control the Turtle is like learning to speak a language" [13:p56] and to the way in which the Logo environment entices learners into "imagining themselves inside the system" [14:p199]. A related sentiment is evident in Papert's appeal for "syntonicity" in learning activities [13:p63]: performing tasks that resonate with other aspects of experience and being emotionally in harmony with one's environment. And, as Papert remarks, the most effective learners are those who exhibit "a tendency to see things in terms of relationships rather than properties" [14:p199].

Papert's perspective on learning, as set out in *Mindstorms*, is complemented by the pedagogy of 'constructionism' that he later elaborated in *The Children's Machine* [14: Chapter 7]. Constructionism is the core theme of a series of international conferences

to which many of Papert's students and co-researchers are major contributors. The nature and scope of Papert's vision for constructionism, which remains controversial, was the theme of a core discussion at the most recent Constructionism conference in Bangkok, Thailand [29]. Relevant questions concern: How effective a role has Logo played in education? What is the relationship between constructionism and other disciplines: to what extent is it associated with learning mathematics, computer programming or computational thinking? Is it appropriate to interpret constructionism so broadly that it encompasses craft-based activities and music-making where the computer's role is no longer prominent or may even be absent? To what degree is the impact of constructionist ideas on education being inhibited by innate societal pressures (cf. Papert's allusion to "a permanent dilemma faced by anyone who wishes to produce radical innovation in education" [13:p140])?

There are many indications in *Mindstorms* about what Papert considered crucial to the full realisation of his vision for learning. In reviewing these, it is helpful to bear in mind that Papert was primarily interested in how the computer could transform the experience of the learner, and was not narrowly committed to any particular technical approach to computing *per se*. In this respect, his perspective is similar to that of Charles Crook, a psychologist whose primary interest is likewise in the impact of computers on the learning experience. In *Computers and the Collaborative Experience of Learning* [2], Crook questions whether constructionism has delivered to its promises (e.g. "The Logo experience reveals that even the most engaging and ingenious computer environment can fail to support pupils' learning" [2:p110]) but also stresses several themes that are represented in *Mindstorms*. These include four key interrelated ideas to be amplified in the discussion which follows:

- the critical importance of being able to exploit the computer as a means to create common knowledge,
- the great yet-to-be-realised potential of the computer in this respect,
- the recognition that thinking of 'programming the computer' is not an appropriate way to conceive this role, and
- the vital need to develop a richer conceptual framework in which to address such concerns.

**Creating common knowledge**: The emphasis that Crook gives to intersubjectivity and its role in instruction [2:p101] may initially seem to be at odds with Papert's stance. The high profile that Papert gives to personal experience, and to learning as something that is ultimately a matter for the individual, is potentially misleading. When considered alongside his forceful criticism of school practices, it may be construed as suggesting that children learn best independently without the need for instruction and perhaps even without reference to their broader social context. In fact, in his account of constructionism [14: Chapter 7], Papert emphasises the public nature of construction and its role in externalising the learner's thinking and promoting engagement within the wider social context. A concept such as "they have to be *her* connections" should not be interpreted as denying a role for 'instruction' but as echoing Piaget's dictum (as cited in [2 :p17]): "each time we prematurely teach a child something [s]he would have discovered for himself, the child is kept from inventing it and consequently from understanding it completely" [15].

**Potential for the computer**: That Papert is profoundly concerned with the potential role that the computer can play in moving from the subjective to the objective realms of experience is most evident in relation to learning mathematics: "[The computer] is unique in providing us with the means for addressing what Piaget and many others see as the obstacle which is overcome in the passage from child to adult thinking. I believe that it can allow us to shift the boundary separating concrete and formal. Knowledge that was accessible only through formal processes can now be approached concretely. And the real magic comes from the fact that this knowledge includes those elements one needs to become a formal thinker." [13:p21] His aspirations for programming as a way of promoting epistemological reflection are broader than this – they relate to the way in which interaction with computers promotes communication and negotiation of meaning more generally. And whilst Crook looks critically at the ways in which computing technology is being deployed in education, he also suggests "that it has a special potential for resourcing the social construction of shared knowledge" [2:p189].

**Beyond conventional programming**: In his analysis of interactions with computers, Crook identifies loss of context as especially problematic for the case of activities supported by computers [2:p105]. His concern is one facet of the way in which programming practices encourage the learner to abstract elements from the situation to which their programs refer and so invite the pedagogist to 'isolate the pupil-technology component of the interaction' [2:p107]. Crook also identifies the rule-based nature of programming activity as potentially diminishing the quality of human interactions in the educational context: "... the meaning of some teaching utterance is rarely to be located in, or made manifest through, its simple surface features – as if such meaning were something to be generated by a rule-bound system of the sort that computer-programmers would seek to construct" [2:p119]. It is quite apparent that Papert [14:p171] is open to much broader interpretations of programming than a computer scientist might endorse. Programming is typically promoted as a discipline that teaches children the need for absolute precision in thought and expression, since there can be no negotiation of meaning with a computer. Yet, in [14:p171], Papert poses the question: "is there such a thing as 'the concept of programming', or is programming something that can be constructed in radically different ways?" and subsequently goes on to propose a notion of programming, quite alien to traditional computer science, that is "inherently biased towards evaluation not by 'is it right?' but 'where can it go from here?'" [14:p173].

**A richer conceptual framework**: For Papert, the computer is the basis for means of communication that can extend, and in certain contexts perhaps supersede, language. Citing Timothy Gallwey's popular book *Inner Tennis* [3] as a source of inspiration, Papert observes: "people need more structured ways to think and talk about the learning of skills. Contemporary language is not sufficiently rich in this domain." [13:p98] In analysing interaction with computers in a collaborative learning context, Crook identifies ways in which the presence of computing artifacts can reduce the need for explicit verbal communication. This leads him to observe that studying discourse in order to clarify that shared knowledge is in place is problematic [2:p181], highlighting the need "to develop a conceptual vocabulary for talking about cognition as distributed, or shared achievement" [2:p138].

## 4 Making Construals

The need to give a good account of the connection between the formal and the experiential aspects of computing, as highlighted by Papert in *Mindstorms*, is evident in many applications. Educational robotics epitomises the challenge of reconciling a computer science perspective, rooted in computational thinking, with an engineering perspective, rooted in physically-based experimental practices. The same concern arises in general in software development, especially where the context for the development is novel and entails close integration with physical devices and systems. Blending the formal abstract framework in which computer programs are conceived and the pragmatic empirical setting of engineering raises fundamental questions about the role of mathematics and logic in relation to experience (see for example the eminent software consultant Michael Jackson's analysis of what we can expect of program verification [7]). These concerns have been the principal motivation for the Empirical Modelling research programme [26,27] over the last thirty years.

Papert, following Piaget, is primarily concerned with the process of construction by which a child's subjective empirical understanding of the world comes to be connected with objective understandings that underlie mathematics, science and natural language. The notion of construction also has broader relevance to areas such as software development, where the perspectives of many different kinds of agent, both human and automated, have to be analysed and orchestrated. Empirical Modelling proposes principles and practical techniques that can underpin a conceptual framework for 'constructivist computing' [27:#100]. Central to this is an experientially-guided approach to creating 'objects-to-think-with' that is characterised as 'making construals'. Disseminating the concept and culture of making construals is currently the theme of the EU Erasmus+ CONSTRUIT! project [24].

A full discussion of Empirical Modelling (EM) and making construals is far beyond the scope of this paper. More background on EM in relation to software practices may be accessed from an online repository of EM papers [27:#114,#121]. EM publications on educational technology include doctoral theses by Roe [16] and Harfield [5]. Introductions to the concept of making construals [27:#128] and to the online environment that has been developed for this purpose in the CONSTRUIT! project [27:#134] are also available. This section will outline how making construals relates to Papert's agenda in *Mindstorms*. Illustrative examples relating to this theme will be accessible online via an associated webpage [25].

Informally, 'making a construal' means figuring out how you think something works. The activity plays a fundamental role in everyday life, especially when we encounter unfamiliar situations, as in trying to make sense of a new place, culture or language. The personal and provisional nature of our construals is characteristic, as is the fact that they are associated with a specific moment and context and are typically liable to change. Papert's reference to 'a notion of programming' that is "inherently biased towards evaluation not by 'is it right?' but 'where can it go from here?'" [14:p173] hints at a similar frame of reference, where attention is focused on the conceiving the next step in the current situation.

Physical media often play a part in making construals. We may refer to a map, or draw up our own informal sketch map to explain where we think we are. In interpreting a construal, or assessing whether it is an appropriate construal, we may

interact within the external situation and with the construal we have made of it, whether in our imagination or in the physical world. The physical artifact we create in this context might qualify as what Papert identifies as ‘an object-to-think-with’; we can identify such an artifact with a construal by taking account of the thought we invest in interacting with it. The appropriateness of a construal is to be gauged in a pragmatic way – we are concerned with whether it is good enough to serve a purpose (cf. Papert’s reference to "thinking [that] is always right and vaguely wrong at the same time" [14:p167]) rather than in some sense ‘absolutely correct’. We are quite accustomed to accepting that our construals are personal and subjective and that this is particularly significant when communicating with novices and children (cf. Papert’s concern for appreciating "the non-obviousness of what we consider to be obvious" [13:p41]). This principle is illustrated in a Shopping construal [25] which integrates adult and child perspectives on a shopping scenario.

Making construals is well matched to Papert’s need for creating “a dynamic model of how intellectual structures come into being and change” [13:p166]. The adopted term ‘construal’ was previously introduced by David Gooding to describe the way in which the celebrated experimental physicist Michael Faraday first made sense of electromagnetic phenomena (cf. [4, 27:#114]). Such an application of making construals resonates with Papert’s motivation for introducing the concept of a microworld in which the laws of physics could be freely postulated [13:p138]: “The propositional content of science is certainly very important, but constitutes only a part of a physicist's body of knowledge. It is not the part that developed first historically, it is not a part that can be understood first in the learning process, and it is, of course, not the part I am proposing here as a model for reflection about our own thinking.”. A construal of the most basic concepts of linear algebra [25] serves as a simple illustration of Papert’s constructivist vision: it models the transition from concrete empirical observation of the canvas to the abstract formal structure of a 2-dimensional linear space.

In his discussion of collaborative learning [2:p188], Crook identifies the need “to understand more about how structural details of educational software support or constrain the possibility of collaborative work”. Crook’s motivating concern is how the use of the computer in an educational setting can foster intersubjectivity, an objective that is pivotal in Piaget’s constructivist outlook on child development and to Papert’s goals for constructionism. From an EM perspective, the aspirations for constructionism are not well-served by educational software that is conceived in conventional programming terms. For instance (cf. [27:#132,#111,#080]), the idea of construction demands a blending of the roles of the designer, teacher and learner of educational software that orthodox programming principles obstruct, but for which making construals is far better suited.

In practice, the influence of Papert’s concern for “programming inherently biased towards evaluation not by 'is it right?' but 'where can it go from here?'” [14:p173] is seen in the style of programming that has emerged in educational technology (cf. Section 2 above). Figure 1 reflects the way in which the direct but informal connection between suitably presented forms of program code and what the learner experiences takes precedence over the formal analysis of sequences of instructions. The program code is presented not simply as a closed representation of ‘the correct behaviour’ but as an invitation to ‘what if?’ experiment. The same mental model also

applies directly to educational software based on spreadsheet and dynamic geometry principles [31,28].

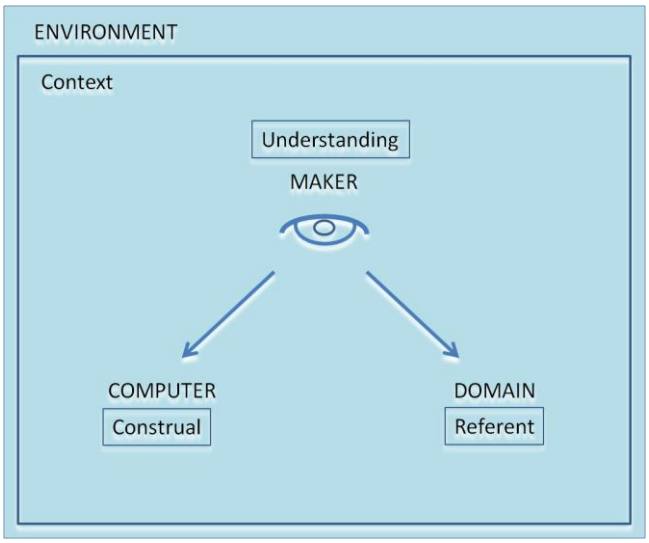


**Figure 2. Making a Construal**

Figure 2 depicts the generalisation of Figure 1 that is associated with making construals. Note that making a construal does not necessarily have to involve a computer; it relies on an mode of interpretation and three basic concepts that are suggested naturally by Papert's question *'where can it go from here?'*. The interpretation of Figure 2 relates to a specific moment in the experience of the maker (what Papert is informally alluding to as 'here'). In this moment, in the spirit of James's radical empiricism, the maker seeks to experience a connection between the two aspects of experience that are being offered, on the one hand, by the construal and, on the other, by its referent. This connection is robust if the construal and its referent can be correlated in such a way that there is a close correspondence between what the maker encounters in the referent and in the construal in three respects:

- the agency that is perceived to be at work ("**agents**"),
- the entities this agency can affect ("**observables**")
- how changes to these entities are deemed to be concomitant ("**dependencies**").

A good correlation guarantees that the construal can serve in the role of an object-to-think-with in relation to the referent, and that the agency attributed to the maker in the construal gives a good indication of "where it can go from here".

The building of a construal proceeds in parallel with developing understanding of the nature of its referent that is reflected in the identification and refinement of ever richer observables, dependency and agency. The most significant feature that distinguishes Figure 1 from Figure 2 is the reference to a 'Context' for the making that also evolves in parallel. In programming, the aim is to establish a context in which the observables, dependency and agency considered can be abstracted and viewed in computational terms – a familiar process of simplification through model-

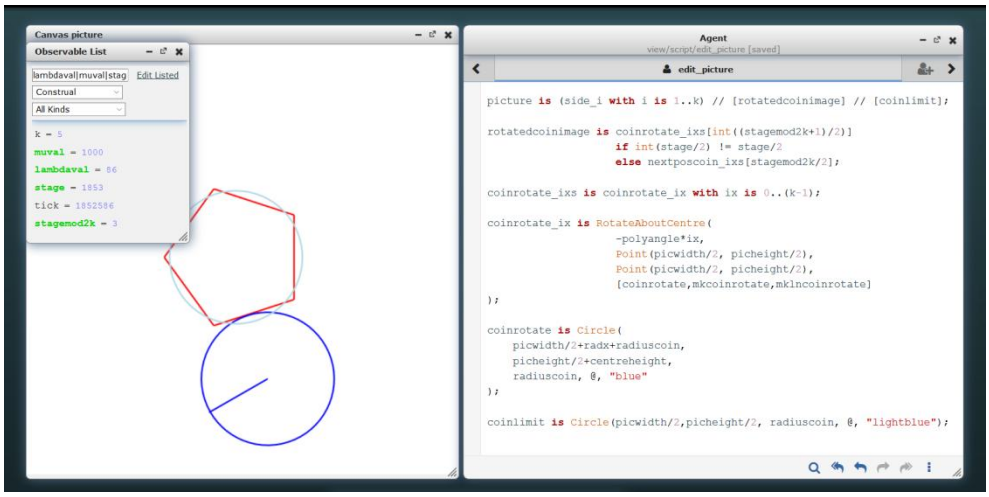


**Figure 3: A construal for Papert's penny rolling puzzle (see [25] and [13:p150])**

building that is characteristic of theoretical science as and when robust construals exist. When making a construal, the context in Figure 2 is typically more fluid and volatile, in keeping with the underlying spirit of uncertainty, exploration and experiment.

In using the computer to make a construal, the current configuration of observables and dependencies is framed as an acyclic network of definitions (or "script"). A typical agent action involves assigning a new value, or giving a new definition, to an observable. Figure 3, a screenshot from the current online environment, is a sketchy indication of how a construal is made: in this case, the subject of the construal is the object-to-think-with that Papert associates with a penny rolling problem posed by Martin Gardner [13:p150]. Papert conceives this object-to-think-with, in which an approximating regular k-polygon of the same circumference is substituted for the stationary penny, to account for the fact that when one penny is rolled around another it undergoes *two* complete revolutions. The extract from the script includes an observable rotatecoinimage to represent the rolling coin. This is based on observing the coin in two different phases: as it rolls along a side of the pentagon, and as it turns at a corner. The observable coinrotatecoin_ixs is a list of all k possible side rolling configurations of the coin, as derived from the template observable coinrotate. The way in which the configuration of the rolling coin is correlated with passing time (as recorded in the tick observable) can be inferred by inspecting key observables that are displayed in the Observable List at the top left. They include the observable stagemod2k which determines which phase of the coin motion currently applies. In keeping with the intended agency that Papert invokes in interpreting his object-to-think-with, the number of sides of the polygon can be freely changed dynamically (even whilst the animation is in process) so that, as k increases, it converges to a circle of the same radius as the rolling penny (as depicted in Figure 1).

In his *Talks To Teachers* [8], William James emphasises the distinction between understanding the basic psychological principles of learning and the art of teaching.

He is at pains to point out that in order to be an effective teacher, it is neither necessary nor sufficient to be familiar with the underlying principles of learning. In a similar way, making construals is presented as a way of understanding what is involved in developing educational software that is to be distinguished from the practical skills needed to create a specific educational application in the most direct and effective manner. The purpose of making construals in the framework of Figure 2 is not to displace other simpler approaches to programming specific educational applications but to clarify the fundamental principles on which these operate and in the process bring coherence to the semantic and epistemological principles involved. The most significant potential impact of introducing this new perspective is to give clear expression to the great varieties and subtleties of meaning that inhabit the space in which construction occurs. In practical terms, this can add new depth to Papert's analogy between learning to 'instruct' the computer and learning to speak a language, providing a shared interactive artifact that can enrich the conversations between the human agents who contribute to design and problem-solving in many different roles. This is a function that cannot in general be served by conventional programs.

A key point is that making construals is a way of exploiting the computer that transcends a pure computational framework. In *Mindstorms*, Papert advocates formulating computational models not necessarily because they are the most appropriate models, but because they oblige us to reflect deeply upon our understanding and the roots of our knowledge (cf. McCarty's account of computer-based modelling in the humanities [11]). He also recognises the potential for learning through interacting with concrete physical objects, both as a precursor to understanding abstract structures (as in the 'gears' of his childhood), and as a way of giving concrete expression to abstract concepts. Learning of the former kind features in 'computer science unplugged' [23], where activities that do not involve the computer are developed to teach computational thinking. Learning of the latter kind, where abstract principles are given concrete practical expression, features in educational robotics. In both these settings, the primary emphasis is on the quality of the interactive experience of concrete artifacts that computing technology enables. It is in just such contexts that the notion of making construals is most appropriately invoked. This is illustrated by a construal of giving change at [25] and by Arduino-based construals in which observables have direct physical counterparts [25].

## 5 Conclusion

This paper pays tribute to Papert by making connections between many of his key insights concerning 'the art of intellectual model-building' and 'making construals'. It also raises some challenging questions.

In interpreting 'constructionism', is has been natural to focus on the idea that the learner takes responsibility for constructing 'objects-to-think-with', and to infer from this that learners must acquire the requisite computing skills. This paper consolidates on previous critiques of 'constructionist' practices [27:#132,#111,#080] to argue that making construals is better suited to this role than conventional programming. It would be facile to suggest that making construals is an easy skill to acquire, however,

as a closer inspection of the construal depicted in Figure 3 indicates. Indeed, not only would it be challenging for a casual learner to make such a construal, but it is questionable whether the educational value of Papert's object-to-think-with, as set out in [13:p150], is sufficiently enhanced to justify the technical investment in its construction. The merits of using the computer to make such a construal relate to a much broader role and agenda.

In understanding how best to use the computer to support learning, (cf. Section 3 above), Papert and Crook's concern is to exploit the computer to its full potential as a way of capturing and communicating common knowledge within a well-conceived supportive conceptual framework in a collaborative learning context. If we liken the environment for making construals to a musical instrument, to expect learners to build construals is perhaps only as realistic as expecting the audience at a concert to be able to compose a violin concerto, play the soloist's part and conduct the orchestra. In exploring this metaphor, the expression 'making a construal' has a helpful ambiguity. To make a construal is not necessarily to create a physical digital artifact – it is to apprehend a connection in your experience. This is something we are capable of doing however limited our technical competence, as is suggested by the choice of the term 'maker' in Figure 2 to refer to the role of every human agent involved.

The goal of making construals is no more and no less than trying to clarify and communicate working understandings. It is a venture in collaborative learning for which there can be no guarantee of success. We should not judge the quality of a construal by how easily it can be constructed by a learner, though we are concerned with how readily the learner can make insightful connections with other experience of the learning domain. To revisit the analogy with music-making, the quality of a musical composition such as a concerto is evaluated in terms of the degree of satisfaction it gives to all who participate in its rehearsal and – the audience members, the soloist, the instrumentalists, the conductor, the critics. A successful musical composition is appreciated by all participants in different ways and degrees according to their level of familiarity and expertise. Appropriating Papert's own word, such success is achieved through 'syntonicity' – each person bringing different agency and observation to bear in such a way that they are able to make and share exceedingly rich connections.

This paper itself makes its own contribution to a wider tribute to Papert in just such a way by highlighting the diverse participants in educational technology who have been able to elaborate on, and celebrate, the products of his vision and imagination.

**Acknowledgments.** I am indebted to all the members of the CONSTRUIT! project team and to Dimitris Alimisis in particular for inviting me to prepare this keynote paper. The CONSTRUIT! project has been funded with support from the European Commission under the Erasmus+ programme (2014-1-UK01-KA200-001818). This publication reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

## References

1. Boytchev, P.: Logo Tree Project, http://elica.net/download/papers/LogoTreeProject.pdf

2. Crook, C.: Computers and the Collaborative Experience of Learning, International Library of Psychology, Routledge, 1996
3. Gallwey, W.T.: Inner Tennis : Playing the Game. New York: Random House. 1976
4. Gooding, D.: Some Historical Encouragement for TTC: Alchemy, the Calculus and Electromagnetism, Paper presented at Thinking Through Computing, November 2-3, 2007. go.warwick.ac.uk/em/thinkcomp07/gooding2.pdf
5. Harfield, A.: Empirical Modelling as a new paradigm for educational technology. PhD thesis, University of Warwick, UK, 2008, go.warwick.ac.uk/em/publications/phd/ant/
6. Hatch, M.: 2014. The Maker Movement Manifesto, McGraw-Hill Education. Available at: http://cds.cern.ch/record/1643837 [Accessed July 8, 2015].
7. Jackson, M.: What Can We Expect from Program Verification? IEEE Computer, Volume 39 Issue 10, October 2006, p65-71
8. James, W.: Talks To Teachers on Psychology: and to Students on some of Life's Ideals, Henry Holt and Company, NY, 1899
9. James, W.: Essays in Radical Empiricism, London: Bison Books, 1912/1996
10. Khan Academy: https://www.khanacademy.org/computing/computer-programming
11. McCarty, W.: Humanities Computing, Palgrave Macmillan, 2005
12. Minsky. M.: The Society of Mind, Simon & Schuster, NY, 1985
13. Papert, S.: 1993. Mindstorms: Children, Computers and Powerful Ideas, Harvester Wheatsheaf, UK (2nd Edition)
14. Papert, S.: 1993. The Children's Machine: Rethinking School in the Age of the Computer, Harvester Wheatsheaf, UK
15. Piaget, J.: 1970: Piaget's theory. In P. Mussen (ed.) Carmichael's manual of child psychology, NY Wiley, p715
16. Roe, C.: Computers for Learning: An Empirical Modelling perspective. PhD thesis, Universty of Warwick, UK, 2003, go.warwick.ac.uk/em/publications/phd/croe/
17. Turkle, S. and Papert, S.: Epistemological Pluralism and the Revaluation of the Concrete. In Constructionism, I. Harel & S. Papert, Eds. (Ablex Publishing Corporation, 1991), p161-191
18. Victor, B.: Inventing on Principle, Invited Talk at Canadian University Software Engineering Conference (CUSEC). 2012, http://worrydream.com/InventingOnPrinciple

Victor, B.: Learnable Programming, http://worrydream.com/LearnableProgramming/

19. Wing, J.: Computational Thinking. CACM Viewpoint, March 2006, pp. 33-35
20. Wolfram, S.: The Poetry of Function Naming, Blog post October 18th 2010 At url: http://blog.stephenwolfram.com/2010/10/the-poetry-of-function-naming/
21. Wolfram, S.: How to Teach Computational Thinking, Blog post September 7th 2016. At url: http://blog.stephenwolfram.com/2016/09/how-to-teach-computational-thinking/
22. Computer Science Unplugged, http://csunplugged.org/
23. CONSTRUIT!: construit.org
24. Edurobotics 2016 resources: go.warwick.ac.uk/em/construit/year3/events/edurobotics
25. EM website: go.warwick.ac.uk/em/
26. EM papers: go.warwick.ac.uk/em/publications/papers/ (each paper is indexed by a 3 digit code #XYZ to be appended to the url)
27. GeoGebra: https://www.geogebra.org
28. Proc. Constructionism 2016, Bangkok, Thailand. Sipitakiat, A., Tutiyaphuengprasert, N. (eds.)
29. http://e-school.kmutt.ac.th/constructionism2016/
30. Scratch - Imagine, Program, Share: https://scratch.mit.edu/
31. Spreadsheets in Education (eJSiE) electronic journal, Bond University School of IT, Australia, http://epublications.bond.edu.au/ejsie/1.