Understanding Children's Programming as Poor Learning Environments

Jochen Rick
College of Computing / GVU Center
Georgia Institute of Technology
Jochen.Rick@cc.gatech.edu

Abstract

Poor learning environments (PLEs) encourage personal commitment and deep understanding. In this paper, I argue that children's programming should be understood from the viewpoint of PLEs.

1. Introduction

I find that environments designed with a poor aesthetic are useful for encouraging personal commitment and deep understanding; I call these poor learning environments. In this paper, I provide some justification as to why PLEs are worthwhile and examine the fundamental characteristics of PLEs. Then, I show that programming environments meet these characteristics and are thus naturally PLEs. Finally, I argue that understanding children's programming from this perspective is useful.

2. Poor Learning Environments

There exists a correlation between theatre and education. With this correlation, theories in either domain can be translated into corresponding theories in the other domain and inform the practices of that other domain. Because theatre has several evolutionary advantages over education, mapping theatre theories into the education domain is particularly informative.

I focus on the theatre theory of Jerzy Grotowski, whose work revolutionized the way many thought about theatre [2]. I translate his theory of a poor theatre into a theory of poor education; in particular, I find that Grotowski's poor aesthetic is useful for designing learning environments that encourage personal commitment and deep understanding.

In many modern theatres (both community and professional stages), the performance of a play has been trimmed to its surface elements. With around four weeks of rehearsal, the actors barely have time to memorize their dialogue and the scenic action. Almost all of the deep work on character, plot, and theme has been ignored in favor of the surface level elements that are the least common denominator for a performance to take place. Grotowski posits that this alarming trend is due to the modern stage trying to compete with its successful spin-off, the screen of the cinema and television. He finds that the stage has been trying to compete with the screen on exactly the qualities that the screen will always beat the stage—in richness.

Grotowski's remedy is simple: "If it [the theatre] cannot be richer than the cinema, then let it be poor." Instead of trying to compete on surface elements, Grotowski shifts the focus of the stage theatre back to the actor and the essential power of the actor to convey emotions, feelings, and ideas. As such, he asserts that the stage does have a place without having to compete with the screen on its terms.

Just as Grotowski finds that the modern stage has deteriorated because of its focus on surface elements, I find modern education to be lacking. Because of the screen, the stage has tried to satisfy viewers with ever decreasing attention spans that demand action and immediate gratification. Similarly, as the amount of knowledge in the world has increased (in some fields dramatically), more and more is seen as common knowledge for every student to have a glimpse of. Instead of deeply understanding domains, today's students are being exposed to so many fields that no field can be covered thoroughly enough to be deeply understood. Instead of really understanding systems and how separate elements work together, students are bombarded with surface level knowledge that is easy to test. In theatre, the emphasis of the screen on surface qualities, such as looks and special effects, has encouraged stage performances to try to match those qualities. In education, quiz shows, like Jeopardy™, encourage the notion that what makes a person "smart" is being able to answer many questions across numerous fields. This enforcement of surface-level traits causes society to forget what is really important—communication of ideas and deep understanding.

My remedy for education is the same as Grotowski's remedy for theatre: let it be poor. I claim that Grotowski's sentiment should be applied to computing environments too; we should create poor computing environments.

3. Characteristics of a PLE

Let me explain what I (and Grotowski) mean by "poor." Commonly, "poor" is another way of saying "bad," and "rich" another way of saying "good." Webster's dictionary defines "poor" as "lacking material possessions." So, what possessions do poor computing environments lack? Simple, they lack the possessions that rich environments have.

Poor environments are *empty*; they lack content. It is up to the learner to create the content. Actively creating the content is necessary for learners to engage the domain. Creating a simulation is poor compared to seeing someone else's simulation. An empty environment that allows you to create simulations is preferable to a collection of simulations full of content. Having to discover the principles of a field is poor compared to having them available to look up. An empty environment that allows you to discover is preferable to a full one that shows. Collaborating with others to create an artifact is better than examining a fully formed artifact. Creating content is preferable to viewing content.

Poor environments are *dumb*; they lack intelligence. Deep understanding requires mastery; at the end of the learning experience, the learner should have some mastery of the domain. To support this, it is useful if the environment is dumb—simple enough to understand. Since the environment captures the essence of the domain, the learner must have a real understanding of the environment to understand the domain. A simulation that follows dumb (understandable by the learner) rules is preferable to one that acts without clear rules (as if it had smarts). An exploratory tool is only helpful as a tool if it behaves in a predictable (dumb) manner. Computer programs seem smart when you do not understand what they are doing. A good learning environment needs to be understandable; it needs to be poor.

Poor environments contain *multiple representations*. When the domain is in its simplest form, it is possible/useful for it to be split into multiple representations. Most domains are complex: many factors contribute to the overall effect. Still, there is usually a fundamental concept that is necessary for better understanding of the domain. A simulation with too many secondary factors can obscure the fundamental one. Learners need simplified environments with multiple representations of the fundamental concept.

4. Children's Programming

Programming environments are inherently poor. They are empty; it is up to the user(s) to create (i.e. program) something. They are dumb; programming languages obey a number of simple logical rules. When users are able to use these rules, they can create complex programs to explore various concepts. They contain multiple representations; at minimum, programming environments have two representations—the source and the running program. Moving between these representations allows the users to gain an understanding of the whole that is more than the sum of its parts.

Since programming environments meet the three criteria of PLEs, they encourage personal commitment and deep understanding. Thus, it seems only fitting that programming environments have been used in learning situations. As children program in these environments, they are able gain deep understanding in a way that is personally meaningful.

5. Discussion

Understanding children's programming as PLEs is useful in several ways. First, it provides a theory that justifies why children should program.

Next, the poor design aesthetic provides a design methodology for creating children's programming environments. It is nearly impossible to design a poor environment with a learner-centered design process. You cannot simply start with the needs of the learner and create such an environment. Because the poor design aesthetic forces you to remove features and reduce the learning environment to its essential nature, the focus of the design becomes the environment (and not the learner). The environment, reduced to its essentials, can only address a few domains. This is not to say that learners should be ignored. But, part of the design process involves finding the audience, rather than starting from learner needs. So, a designer of children's programming environments needs to understand the affordances of the technology and the learner's needs.

Finally, children's programming provides concrete evidence of useful PLEs. Like end-user programming, children's programming environments specialize on a domain [3]. Logo specializes (or at least used to specialize) in geometry [4]. StarLogo allows children to explore decentralized thinking [5]. MOOSE Crossing provides a meaningful context for students to engage in creative writing [1]. All of these are PLEs, yet they all specialize in different domains. Thus, children's programming demonstrate how learning environments can be designed with a poor aesthetic, but still address more than one domain.

6. References

[1] A. Bruckman, "Community Support for Constructionist Learning", *CSCW* 7, 1998, pp. 47:86.

[2] Grotowski, J., *Towards a Poor Theatre*, Methuen & Co Ltd., London, England, 1968.

[3] B. Nardi, *A Small Matter of Programming: Perspectives on End User Computing*, The MIT Press, Cambridge, MA, 1993.

[4] S. Papert, *Mindstorms: Children, Computers, and Powerful Idea*, Basic Books, Inc., New York, NY, 1980.

[5] M. Resnick, *Turtles, Termites and Traffic Jams: Explorations in Massively Parallel Microworlds*, The MIT Press, Cambridge, MA, 1994.