Towards Making a Computer Tutor for Children of All Ages

Yoshiki Ohshima, Alessandro Warth, Bert Freudenberg
Aran Lunzer, Alan Kay

Presented at the PX/16 workshop,
co-located with ECOOP 2016 in
Rome, Italy July 18 2016

VPRI Technical Report TR-2016-002

Viewpoints Research Institute, 1025 Westwood Blvd 2nd flr, Los Angeles, CA 90024   t: (310) 208-0524

Towards Making a Computer Tutor for Children of All Ages

– A Memo –

Yoshiki Ohshima
Human Advancement Research Community
Y Combinator Research
yoshiki.ohshima@ycr.org

Alessandro Warth
Human Advancement Research Community
Y Combinator Research
alex.warth@ycr.org

Bert Freudenberg
Viewpoints Research Institute
bert@freudenbergs.de

Aran Lunzer
Viewpoints Research Institute
aran@acm.org

Alan Kay
Viewpoints Research Institute
alan@vpri.org

ABSTRACT

One of the primary goals of our research group is to improve education through computing. We are interested in unleashing the power of the computer to create automated "intelligent" tutor systems (ITS). This paper presents ideas that may guide the design of such a system, targeting the problem of computer-programming education in particular. We also outline a research and development plan to build this system. While this plan is just a straw-man (there is a lot of uncertainty), our hope is to get a discussion started on this important topic.

CCS Concepts

•Human-centered computing → Interaction paradigms; Collaborative interaction; •Computing methodologies → Artificial intelligence;

Keywords

Programming Education, Intelligent Tutor System, Multimodal interaction

1. INTRODUCTION

The computer has become an essential part of our civilization. However, the way people use this decades-old technology still has not unleashed its full potential. As one of the authors said nearly 20 years ago, and we still believe holds true: "The Computer Revolution has not happened yet"[15].

People are often stuck with mundane applications provided by others. This is problematic because they cannot look under the hood [14] to understand how things work, or modify applications to suit their own needs. Another problem is that applications are often mere simulations of old media such as paper print. We expect that treating the computer as its own medium (and taking advantage of its dynamic nature) will trigger a revolution that rivals the one that was brought on by the invention of the printing press.

What can we do to accelerate the revolution? From our observation, the computer revolution is intertwined with the education revolution (and vice versa). The next steps in both are also highly overlapped: the computer revolution needs a revolution in education, and the education revolution needs a revolution in computing.

We think that, for any topic, a good teacher and good books can provide an "above threshold" education. For computing, one problem is that there aren't enough teachers who understand the subject deeply enough to teach effectively and to guide children. Perhaps we can utilize the power of the computer itself to make education better? We don't hope to be able to replace good teachers, but can the computer be a better teacher than a bad teacher?

Why is this a good time to start this project? One main factor is that computers are getting more powerful, and in some domains we can say they are getting smarter [8]. Also, a new generation of children is growing up in an environment where computers exist as a commodity. We would like to seed the idea that such devices are much more capable than a TV that sometimes responds to your finger flicks.

While CAI (computer-assisted instruction) [5] is applicable to many different domains, the domain we are particularly interested in is early computer programming. In our experience teaching Etoys, we have observed that children

PX/16, July 18 2016, Rome, Italy

 ISBN 978-1-4503-4776-1/16/07. . . $15.00

DOI: http://dx.doi.org/10.1145/2984380.2984383

aged 11 and above are capable of making plans and writing non-trivial programs.

While there have been some attempts to provide remote tutoring [9] and self-teaching online courses, we would like to provide an environment that can motivate not only the top 10% of students, who tend to be strongly auto-didactic anyway, but also the next 80%, with potentially lower levels of inner motivation.

Our vision is to have a computer interface where the computer can *observe* the user's actions, along with eye gaze, hand movement and so on, so that it can encourage, suggest, guide, and teach the user. In other words, we would like to bring the vision of computer-assisted education to the next level.

2. APPLICATION TO PROGRAMMING EDUCATION

Our focus is on computer programming. We have experience in tile-based (often referred to as block-based) programming with children, and believe it has some advantages for early learners. One reason is that tiles have a lower entry barrier when compared to text-based programming; for a start, the user never has to worry about syntax errors. Another reason is that it helps to have instructions from the tutor in kinesthetic terms; in a tile-based program, we observed that users can easily understand instructions of forms such as "move this tile to there". In contrast, in a text-based program, the tutor often dictates what the program should look like including cryptic symbols and unfamiliar words (e.g., `fn`, `lt`, &, `**`, #!, etc.).

From our experiences with Etoys and other systems, we learned that most children can follow our instructions to create our standard "drive a car" example [7] (see Figure 1 for a typical project). The drive-a-car project involves creating a car widget and a steering wheel widget, and writing a short program involving those widgets. This example proves to be an effective starting point, as it is engaging and introduces many powerful concepts such as turtle geometry, iteration and conditionals.

We hope to create a system that enables children to learn and apply these powerful ideas even when they don't have us there to guide them.

2.1 Ideas

There are many challenges in arriving at this vision. In this section, we introduce some of these challenges and propose ideas for addressing them.

For CAI, it is beneficial to know what the user is trying to do. Consider our drive-a-car example: even when projects the learners make vary in details, they are still recognizable as drive-a-car attempts. Today a human can certainly recognize variations and unfinished drive-a-car projects as such. We anticipate that a computer algorithm will be able to do the same and, for a project that is not working, be able to suggest steps to fix it. Alternatively, as shown by Brown et al., the system may have a database of common user mistakes and advice for correcting them. [12]

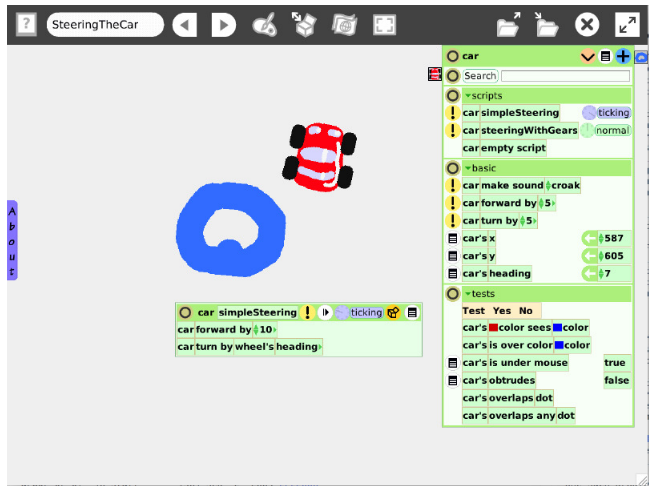

Figure 1: The Drive A Car example in Etoys.

How do we sustain the learner's focus and motivation, and thus make the entire user experience more fruitful? This will require some experiments with pedagogy on the computer. We believe that the situation where the tutor sits next to the learner is ideal; how would we make a simulation of such a tutor? Our current thinking is to have a computer avatar or an agent on screen that helps the user.

Although on-screen agents have had a troubled history (a famous example being Microsoft's "Clippy" Office Assistant), we still see potential in some existing attempts to incorporate such agents in an educational setting. Wang et al. created a system with eye tracking and multiple empathic characters reacting to the user's eye gaze and interactions [22]. Cycorp experimented with a computer agent in a learning-by-teaching setting, in which a human learner is asked to teach elementary mathematics to a computer-generated avatar who is (i.e., pretends to be) struggling with mathematics concepts [16]. (See Section 3.2 for more discussion.)

We could experiment with different kinds of agents. It might be more effective for the tutor to appear to be the same age as the user. Maybe using a non-human creature would avoid the "uncanny valley" problem. How about having multiple agents interacting with the user and each other, where the user is not the only one who is learning? If the user sees an agent make mistakes, she might not worry about making mistakes too.

We could add telemetry sensors to measure the user's activity and state of mind. There are studies that show correlation between pupil dilation and the subject's focus [13]. We might use that kind of information to gauge the user's attention level. Furthermore, we would like to sense where the learner's finger is pointing, and where she is looking. These are things that a human tutor sitting next to the user can easily see; we believe a computer could provide better help by taking these signals into account.

The availability of new kinds of display and user interface technologies provides other opportunities for improving computing education. In this project, we would like to show more context and information, things that are typically transmitted via different modes of communication. A live tutor sitting next to the learner may explain things by drawing diagrams, making hand gestures, and so on. The CAI version of such interactions may involve a head-mounted display or a wall-sized high-resolution display to show programs, data, diagrams, etc. in the appropriate context, enabling the user to put her peripheral vision and spatial memory to good use [21].

Furthermore, researchers might come up with entirely new types of programming systems. In a system where programming is done by assembling physical blocks in an immersive environment or even in the real world, for example, a good simulation of an experienced programmer standing next to the user could be especially effective.

Voice conversation is an important mode of interaction in a typical human-to-human tutoring scenario. The arrival of mainstream commercial "AI" applications such as Siri and Amazon Echo is proof that the state of art in speech technology is coming along. However, the user experience in these systems leaves a lot to be desired. This is especially problematic in a tutoring setting, where we would like to create an atmosphere that is conducive for learning. For example, people tend to get annoyed when a computer agent projects a know-it-all attitude but has actually misunderstood the user's intent. Additionally, because there is constant interaction between the user and the system, even a rare mistake by an agent in a long session may break the user's attention. A challenge, therefore, is to create an atmosphere where the user's "flow" is maintained.

Fortunately, we don't have to do this work in a vacuum. We can build on research from communities such as Artificial Intelligence in Education [1] and Computer Science Education[3]. From the programming language research field and the human computer interaction field, we can learn from research that is concerned with assessing student programs and providing solutions based on common errors such as [20] and [11].

3. A HYPOTHETICAL PLAN

This is a long-term project with many moving parts and parallel development efforts. While there is a lot of uncertainty, we think it is fruitful to start the conversation by laying out a hypothetical plan. The following is such a plan, which is divided into three phases. We expect each of these phases to take 2-3 years.

3.1 First Phase

Some of our collaborators have worked on existing education software systems, such as Etoys, Scratch and Snap!. We plan to experiment using these systems in the following educational scenarios:

- **Live Person:** Initially, a live tutor will interact with the learner from a different room. Our collaborators are teachers with good track records, so we will try to capture their pedagogy in these sessions and observe what works.

  One twist we can try is to have the live person pretend to be an artificial tutor, an experimental technique used for the talking typewriter by O. K. Moore et al [18] and later referred to as "Wizard of Oz". Some of the authors were involved in the implementation of an interactive attraction for Disney theme parks, where a live cast member operates a computer-animated Stitch character (from the film *Lilo and Stitch*) with a game pad and uses a high-pitched voice to act as Stitch to talk to park visitors [4]. For young children, this was a very believable and captivating experience. Even adults were often left wondering whether there was a human behind it, and even when they knew that there was a human, it was still an enjoyable experience. We could build upon this experience to design an avatar.

- **Remote Connection:** We plan to use a VNC-like connection to share the user's screen and cursor position, similar to the Nebraska system on Etoys or the online support of the Python Tutor [9].

  We expect to use laptops for our initial experiments. Tablets are also a possibility, but the lack of a pointing device might make it difficult for the learner and tutor to direct each other's attention to a specific location.

  Based on our colleagues' experience with person-to-person interaction in classrooms, the tutor should never "steal" the mouse or keyboard in an attempt to be overly helpful [2]. We will take these insights into account when we design the system.

We will record as much of the interaction between the tutor and the learner as possible (e.g., their conversation, what is on the screen(s), the program state). We plan to analyze this information to better understand what works and what does not work, and make improvements to the system as we continue on to the next phases.

3.2 Second Phase

We will implement software assistants and gradually move toward more automated interaction with the user. We also plan to leverage more telemetric data such as eye gaze, finger positions, and hand movement. Humans can often detect whether the other person is focused, or if her mind is wandering. Boredom may be inferred based on the rate of clicks and eye gaze patterns [22]. If a human tutor's suggestions and actions would be dictated in part by the perceived state of mind of the learner, we may be able to provide similar automated reactions from an agent on screen. We will also explore ways to display the tutor's fingers on the learner's screen without much distraction.

We will also explore the possibility of training an AI program to recognize features of user-created projects. For example, if our system recognizes that a given project has some "drive-a-carness", it may be able to do a better job of helping its author. This help may come as a suggestion of next steps for a partially finished project or identifying bugs (similar to program repair ideas [23]).

Another area we plan to explore in the second phase of our project is expressiveness. We would like to enable users to create more advanced projects. As projects become more diverse, the users' questions will inevitably go beyond the capabilities of the system. Eventually, the support of a human will be needed in such cases (as stated above, we cannot replace good teachers with a computer). We can try a hybrid approach where a human tutor oversees multiple learners simultaneously. In this setting, unlike Codeopticon [10], we can still have automated responders handling the initial conversation with users, but a question that is too difficult for an agent may be elevated to a human tutor.

We aim to design and build a programming language and environment that is fit for our purposes, based on what we learn in this phase.

3.3 Third Phase (and Beyond)

We hope to have reliable AI features as well as better voice recognition and generation by the time we get to this phase of the project. That being the case, the early part of programming education can be fully automated, and we can provide a one-to-one learner to (automated) tutor ratio.

We hope that our new programming system will be used to develop new curricula, with enough intelligent agent support to help children around the world even if they lack access to knowledgeable adults.

4. CONCLUSIONS

Educating the next generation is humanity's most important endeavor, as solving other problems depends on it. We believe that true computer literacy will give us a good boost, when the new generation can use computers for making better things as well as better decisions, arguments, etc.

In this paper, we laid out a 10-year project plan. In the first phase, we will experiment with existing systems and curricula to gather data on effective tutor and learner interaction, and to gain experience in designing the next generation of intelligent tutoring systems. In the middle phase, we will try to build a new system from those experiences, and gradually move toward computer-agent-based interaction. In the final phase, we will experiment more with the new system equipped with the agents.

Along the way, we would like to design an entire system, including the programming language and programming environment, with the primary goal of supporting education. This system will have to provide a low entry barrier for the user. On the implementation side, it will have to be built with a solid foundation for networking and communication, and support for logging user actions and program state.

We hope that our system (and systems like it) will motivate new research by the Programming Experience community that targets budding programmers, i.e., how can we tailor the PX to make things easier for the tutor and learner? Both would certainly benefit from better program visualizations [19], live programming features [17], back in time debuggers [6], etc.

Finally, we posit that the new generation of computer-assisted instruction systems can be applied to teach *any* subject, anywhere in the world; from The Three R's (reading, 'riting, and 'rithmetic), to advanced science and mathematics and beyond.

Some people object to the idea of their children being taught by computers. Some may say that it is a form of cultural invasion. Some may say that education is not only about learning mechanical skills but also something about providing the moral compass, which should be done through human interaction.

We embrace these genuine concerns, and accept that our learning environments must mesh with the core values of the surrounding society—against the backdrop of our strong belief that giving all children the opportunity to learn is non-negotiable.

A related future project might be to help teachers who need to teach programming in classrom but do not have much programming experience. In other words, a bad teacher might become a better teacher with some dynamic assistance from computers in the classroom.

This is a big challenge that involves new technologies and research. The AI part alone will need substantial advances from today's capabilities. We view this as a long-term project, with many and diverse collaborations, that will bring us closer to our goal of meaningful education for children of all ages and all countries.

Acknowledgments

We would like to thank the reviewers and attendees of Programming Experience '16 workshop (PX/16) for providng us insightful comments.

5. REFERENCES

[1] International Artificial Intelligence in Education Society. `http://ijaied.org/`.

[2] Personal communication with Kazuhiro Abe. The idea is also described in: `https://github.com/yasslab/scratch_tutorials`.

[3] Special Interest Group on Computer Science Education. `http://sigcse.org/`.

[4] Stitch's Photo Phone. An experimental interactive attraction. A 2003 Thea Award Winner: http://theming.directory/thea/.

[5] R. C. Atkinson and D. N. Hansen. Computer-Assisted Instruction in Initial Reading: the Stanford Project. Technical Report 93, Institute for Mathematical Studies in the Social Sciences, 1966.

[6] R. M. Balzer. EXDAMS: Extendable Debugging and Monitoring System. In *Proceedings of the May 14-16, 1969, Spring Joint Computer Conference*, AFIPS '69 (Spring), pages 567–580, New York, NY, USA, 1969. ACM.

[7] B.J. Allen-Conn and K. Rose. *Powerful Ideas in the Classroom*. Viewpoints Research Institute, 2003.

[8] D. S. et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529:484–489, 2016.

[9] P. J. Guo. Online Python Tutor: Embeddable web-based program visualization for CS education. In *Proceedings of the 44th ACM Technical Symposium on Computer Science Education*, SIGCSE '13, pages 579–584, New York, NY, USA, 2013. ACM.

[10] P. J. Guo. Codeopticon: Real-Time, One-To-Many Human Tutoring for Computer Programming. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology, UIST 2015, Charlotte, NC, USA, November 8-11, 2015*, pages 599–608, 2015.

[11] B. Hartmann, D. MacDougall, J. Brandt, and S. R. Klemmer. What would other programmers do: Suggesting solutions to error messages. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 1019–1028, New York, NY, USA, 2010. ACM.

[12] John Seely Brown, Richard R. Burton, Catherine Hausmann, Ira Goldstein, Bill Huggins, and Mark Miller. Aspects of a Theory for Automated Student Modelling. Technical Report 3459, Bolt, Beranek and Newman, Inc., 1977.

[13] D. Kahneman and J. Beatty. Pupil Diameter and Load on Memory. *Science*, 154(3756):pp. 1583–1585, 1966.

[14] A. Kay. Opening The Hood Of A Word Processor, 1984.

[15] A. Kay. The Computer Revolution Hasn't Happened Yet, 1997. A talk given at the OOPSLA '97 conference. Movies are available online.

[16] D. Lenat and P. Durlach. Reinforcing Math Knowledge by Immersing Students in a Simulated Learning-By-Teaching Experience. *International Journal of Artificial Intelligence in Education*, 24(3):pp. 216–250, 2014.

[17] S. McDirmid and J. Edwards. Programming with Managed Time. In *SPLASH Onward!* ACM, October 2014.

[18] O. K. Moore and A. R. Anderson. Some Principles for the Design of Clarifying Educational Environments. Technical report, Pittsburgh University Learning and Development Center, 1968.

[19] J. Ou, M. Vechev, and O. Hilliges. An Interactive System for Data Structure Development. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pages 3053–3062, New York, NY, USA, 2015. ACM.

[20] R. Singh, S. Gulwani, and A. Solar-Lezama. Automated Feedback Generation for Introductory Programming Assignments. In *Proceedings of the 34th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '13, pages 15–26, New York, NY, USA, 2013. ACM.

[21] B. Victor. Seeing Spaces. `http://worrydream.com/#!  /SeeingSpaces`.

[22] H. Wang, M. Chignell, and M. Ishizuka. Empathic Tutoring Software Agents Using Real-time Eye Tracking. In *Proceedings of the 2006 Symposium on Eye Tracking Research & Applications*, ETRA '06, pages 73–78, New York, NY, USA, 2006.

[23] W. Weimer, T. Nguyen, C. Le Goues, and S. Forrest. Automatically finding patches using genetic programming. In *Proceedings of the 31st International Conference on Software Engineering*, ICSE '09, pages 364–374, Washington, DC, USA, 2009. IEEE Computer Society.