

Allocentrism and Computational Thinking

Carmen Petrick, The University of Texas at Austin, 1 University Station D5700,
Austin, Texas 78712-0379 USA, carmenpetrick@gmail.com
Matthew Berland, The University of Texas at San Antonio, 1 UTSA Circle,
San Antonio, Texas 78249 USA, matthew.berland@utsa.edu
Taylor Martin, The University of Texas at Austin, 1 University Station D5700,
Austin, Texas 78712-0379 USA, taylormartin@mail.utexas.edu

Abstract: IPRO is a virtual programming environment in which students use handheld devices to program virtual robots to play soccer. The mobile nature of IPRO allows students to directly embody their robots, meaning that they physically enact their programs as if they were a robot. The process of direct embodiment prompts students to interpret their program from an egocentric perspective (i.e. the robot's point of view). We conducted a study of high school students using IPRO, and in this paper, we present the case of how one student directly embodied her robot in order to develop her program. We found that the student used direct embodiment in order to make detailed decisions about her program code and that collaborating with teammates and using other resources helped her make general decisions about her program. We also found that the student alternated between allocentric and egocentric frames of reference while developing her program.

Introduction

Between motion-controlled video games, avatar-based social worlds, and ubiquitous mobile devices, high school students have become avid users of recent technological advances. However, when these students enter technology and engineering classrooms, they experience a significant disconnect from the highly collaborative, interactive devices and applications to which they are accustomed. Traditional computer programming instruction focuses on procedural skills, limits collaboration, and puts little emphasis on building conceptual understanding (Pea & Kurland, 1984; Robins, Rountree, & Rountree, 2003). Developments in the last decade have started to bridge the gap between how computer programming is taught and how students experience programming outside of school (Maloney et al., 2004; Wilensky, 1999). Changing the programming environment can alter the way students understand computer science, engage previously uninterested student populations, and change students' perceptions of the field.

IPRO (which stands for "*iPod Robotics*" and "*I (can) program!*") is a virtual programming environment designed to support students in developing computational thinking and programming skills in a social, mobile space. Students use handheld devices (e.g. the iPod Touch) to program virtual robots to play soccer. Rather than holding class in a computer lab, students work in a large open space and are free to move about the room. This setting promotes collaboration and the ability to use physical movements to intuit and interpret program code. Students can imagine they are each a robot, moving ways that a robot might move; we are calling this *direct embodiment* after work by Fadjo, Lu, & Black (2009). When students directly embody their robots and physically enact their programs, they naturally assume the robot's first-person perspective within the program, or what Berthoz (2000) calls an egocentric frame of reference. This point of view transforms the way information about their programs is perceived and interpreted. While it is possible to imagine a first-point of view without physically pretending to be a robot, students learning to program in a mobile environment can directly embody their robots as a way to use an egocentric frame of reference within their programs. As students collaborate with their teammates, they are more likely to transition to an allocentric, or third person, perspective. This paper presents a case study in order to explore in detail how one student's physical movements allowed her to directly embody her robot, easily access an egocentric frame of reference, and combine this with other resources to develop her program.

As research points towards cognition evolving from perception and action (Anderson, 2007; Varela, Thompson, & Rosch, 1993; Wilson, 2002), the connections between physical movement and cognition become clearer (Anderson, 2003; Seitz, 2000; Wilson, 2002). What we see, hear, feel, and do with our bodies helps us to understand and make sense of what is going on around us. Papert (1980) calls this "body syntonicity." He developed Logo and Turtle geometry specifically to draw upon students' understandings of their bodies and to motivate them with authentic problems that they could solve by drawing from their past experiences. He encouraged students to "play turtle" by directly embodying the Logo turtle and acting out the movements that they wanted to program the turtle to make. In our research, we build on Papert's idea of "playing turtle" by examining how, in our case, "playing robot" can help students access multiple frames of reference and advance their thinking. We hypothesize that alternating between frames of reference is important to the development of computational thinking. The research presented here builds upon the ideas of body syntonicity and, more broadly, upon the theories of embodied cognition (Anderson, 2003; Seitz, 2000) to create a mobile-social

programming environment that will facilitate students' abilities to physically embody their robots and enact their programs leading to new learning pathways. IPRO represents one of the many possible ways to help students learn to program through embodiment.

In our larger study, teams of high school students used the IPRO application to program virtual robots on mobile, handheld devices. This paper presents the case of novice programmer, Amelia, learning to program in IPRO. We observed Amelia during one 90-minute class as she attempted to create a robot that would consistently score goals for her team. Using video of her speech and physical movements as well as log data from IPRO, we examined how and why she alternated between the robot's point of view and the overall program view while building her program.

Egocentrism & Allocentrism

Two main frames of reference, egocentrism and allocentrism, are typically used to describe how objects are situated in space (Berthoz, 2000; Klatzky, 1998). In an egocentric frame of reference, objects are understood according to their relationships with the body; while in an allocentric frame of reference, all objects are oriented according to their relationships with each other, and there is no reference to the body. Egocentrism appears to come more naturally than allocentrism since the information the brain gathers has been perceived by the body and comes inherently from a body-centered point of view (Byrne & Becker, 2008). We believe that teaching students to adopt an egocentric perspective when beginning to program would be beneficial to developing a strong programming foundation.

Helping students take a first-person point of view and adopt an egocentric frame is common in education. Wright (2001) suggests that when a student is able to "get inside" a problem, that might help her to understand it more fully. Elementary teachers often reword story problems to include the child's name that they are working with. Even experts employ similar strategies. Physicists transport themselves into a problem by talking and moving their bodies when they interpret graphs (Ochs, Jacoby, & Gonzales, 1994). This same type of entering into a problem and taking on an egocentric frame of reference occurs in programming when the programmer embodies the object being programmed and enacts the each step of the code controlling it. This process of direct embodiment requires changing the frame of reference with which one views the program.

A programmer can think about a program using an allocentric or an egocentric frame of reference. For example, she could view a program from the "top down" by analyzing overall program structure or "bottom up" by evaluating it line-by-line. By adopting the two perspectives, programmers can create more advanced programs (Ackermann, 1996). A programmer may work "inside the program" by evaluating it step-by-step, only returning to an allocentric frame of reference to make major changes to the program. This interchange between "diving-in" and "stepping-out" (as described by Ackermann, (1996) is important for developing conceptual understanding and a natural part of human development (Kegan, 1982). IPRO is designed to encourage this type of interaction. Unlike traditional programming in which students cannot move away from an immobile computer, students can physically carry their programs with them on the iPod Touch as they move. Students dive into their programs as they directly embody their robots adopting an egocentric frame of reference, and collaborating with their teammates helps them to step back out. As students share their ideas with their teammates, they are able to formalize their thinking, usually through an allocentric frame of reference.

Traditional programming instruction does not teach strategies to help beginning programmers adopt multiple frames of reference, often making it difficult to learn to program. Through IPRO, we investigate direct embodiment as one method of helping students adopt an egocentric frame of reference and collaboration as a way of encouraging an allocentric frame of reference. We explore how adopting multiple frames of reference combined with other resources can help students develop more sophisticated programs.

Methods

In this study, 41 students from two classes in an urban high school in the Southwestern United States used IPRO, an application for iOS that supports multi-agent online games of virtual robot soccer. Pairs of students cooperated to create a robot that works with other robots on their team to play soccer against an opposing team. The IPRO language is a visual programming language deriving from Scheme and implementing a simple functional reactive programming paradigm (Cooper & Krishnamurthi, 2006), in that it uses the concepts of signals rather than constants.

Students in the study used IPRO during one 90-minute class. At the beginning of each class, students completed a 5-minute pre-test that included an attitudinal survey and logical reasoning questions. Then they received an introduction to IPRO followed by 15 minutes of guided instruction. After this, students entered the design-compete phase in which they were instructed to collaborate with their team members to design robots for an upcoming match. The classroom they worked in had a large open space in the center with tables and chairs along the edges. Students were encouraged to work standing up in the center of the room in order to promote movement and collaboration. The design-compete phase lasted approximately one hour. During that time,

students could test their robots in solo play, they could share sections of their code with their teammates, and every 10-15 minutes they competed against other teams in match play.

Solo Play: To try out their programs in a non-competitive environment before a match, students could enter their robots into the solo play room. In solo play, the robot and the ball appear on the playing field in random locations, and the robot runs consecutive iterations of the program until a goal is scored or until the student restarts the play.

Match Play: Every 10-15 minutes all students stop working on their programs and enter a match play room. Two pairs of students form a team and compete against another group of students. The class stands in the middle of the room observing a large screen on which the matches are projected. Each match consists of a set number of turns or iterations of the programs.

At the end of the class period, the students took a post-test, which was similar to the pre-test but also included IPRO programming questions.

For this paper, we chose to closely examine the case of how one student, Amelia, attempts to solve one of the first major challenges students face in IPRO—how to program a robot to score a goal when the robot is located between the ball and the goal. We gathered video recordings of Amelia working on this problem, interviews of her while programming, and log data from IPRO of every change she made to her program in the IPRO application. We analyzed her speech, movements, and code at a very fine grain in order to gain a better picture of the role changing frames of reference plays in the programming process.

Amelia's case

Amelia reported being moderately comfortable with programming but unlikely to pursue a career in a STEM field. The only female in her class of 20 students, she assumed a leading role on her team.

Amelia initially developed a robot that could score whenever it was facing both the ball and the goal (See Figure 1). However, after repeatedly observing her robot in "Solo Play," she determined that it could not score when it was between the ball and the goal. From the transcript, we know that Amelia understood that if the robot found itself between the ball and the goal, then it would have to move around the ball; however, she did not know how to accomplish this.

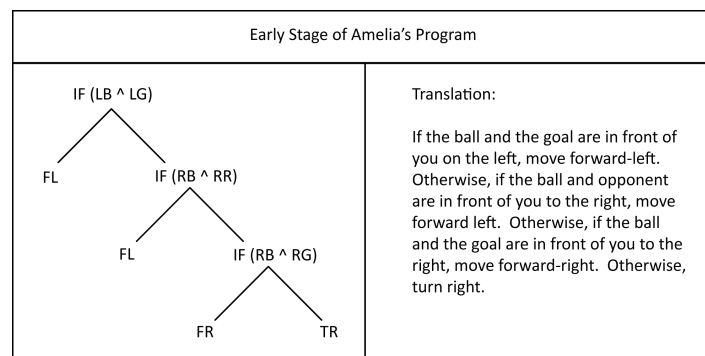


Figure 1. Early stage of Amelia's program.

She brainstormed with her partner, Roger, for a while discussing ways to create a robot that could find the ball and then reorient itself on the other side of the ball facing back towards the goal. After they arrived at a basic plan, Amelia stepped away from Roger and began to directly embody her robot (See Figure 2). She raised her eyes up to the ceiling. Then, she looked down at her iPod. Her shoulders made a slight turn to the left and then to the right. Then she turned her entire body 90 degrees to the right and looked down at her iPod. She then turned back to face the front of the class. At this point, she added the last piece of code (trans. into pseudocode as):

```

if (ball-is-forward-right? and opponents-target-goal-is-forward-right?)
  then move-forward-left()
  else turn-right()
  
```

When asked about her movements, Amelia reported that, "I was trying to figure out which way things were turning." Making her body rotate helped her decide the direction the robot should turn. Amelia then shared her changes with Roger and they observed their robot in solo play again to determine how successful the new program was. This small addition created a successful program that would score in 82% of its attempts in solo play. After scoring a goal during the next match, Amelia and Roger resisted making changes to their robot even leading up to the final match. They made slight modifications to the program a few times, but always reverted back to this program. Even at the end of the design-compete phase, when half of the class worked

together to find the best strategies for their robots, Amelia pushed to keep hers the same. "Guys, can I please leave mine as is?" she asked. "It kind of works." It did work quite well, making a critical save in one match and a steal in another.



Figure 2. 1) Amelia looks at her iPod. 2) She makes a quick, half turn to the left. 3) She turns back to her original position. 4) She makes a quick, half turn to the right. 5) She returns to her starting position and pauses for a moment. 6) She turns all the way to the right, stays there and adds new code to her program.

Analysis of Amelia

Amelia made use of multiple resources as she programmed her robot. She edited her program code, observed her robot in solo play, directly embodied her robot, consulted with her partner, and evaluated her robot during match play. Amelia used each resource for different purposes. After working with her partner Roger to develop her initial program, Amelia used solo play to determine whether or not her robot was performing the way in which she had intended. She alternated between observing her robot's performance in solo play and the program code view on her iPod. This process allowed her to make connections between the code and the robot's behavior and discover that her robot could not score when it was between the ball and the goal. From her interview, we can conclude that Amelia was thinking about her program from an allocentric frame of reference. She referred to connections between the levels, and she pinpointed where the problem in the code occurred.

After identifying the problem with her code, Amelia and Roger brainstormed possible solutions. Then Amelia began to directly embody her robot to help her make decisions on how to edit her program. She switched from an allocentric to an egocentric frame of reference, as she was interpreting her code from the robot's point of view. While directly embodying her robot, Amelia made several small movements trying out different ideas. Physically acting out the possible directions her robot could turn helped her decide on the correct code to enter into her program.

Amelia then switched back to an allocentric frame of reference as she shared the changes with Roger. Then they watched their robot compete in the next match. Throughout the design-compete phase, they used match play to evaluate their robot, and when their robot scored a goal with the new program code, Amelia resisted modifying the program further.

All of the resources Amelia employed provided different information necessary to improve her program. Amelia switched freely between resources and frames of reference as needed to help her make decisions about her code.

Discussion and Conclusion

Amelia's case suggests that novice programmers could benefit from being taught how to "step in" or directly embody their programs. Directly embodying part of her program gave Amelia a frame of reference "inside" the program, and she used this to make detailed decisions about how to edit her code. She utilized multiple resources and alternated between allocentric and egocentric frames of reference, leading to new understandings of the relationships within the program. Directly embodying a part of a program also provided a way for her to quickly try out different ideas or movements and observe the effects.

Students using IPRO regularly act out their programs in different ways to see which outcome is optimal. Directly embodied actions provide means for students with little programming experience to take on egocentric frames of reference and learn advanced concepts. While one can look back as far as Papert (1980) to find children embodying computer programming, our study provides evidence that students can quickly use embodied cognition in computational thinking with very little guidance or direction. In IPRO, students begin embodying their robots almost immediately to help them plan their programs and communicate with their teammates. Since the movements of the robot in IPRO all directly correspond to physical actions with which the

students are familiar (e.g. move to the forward-right, turn left, see or detect ball, etc.), students can easily act out how their robot should move even before they completely understand how the code works. The connection between the physical and virtual world makes the programming environment more relatable for students.

This could have important implications for computer programming curricula. By altering the programming environment, teachers can take advantage of the affordances offered by direct embodiment. Finding ways to incorporate mobile, collaborative activities in computer science classrooms may make computer science more accessible and more enticing, reaching previously uninterested student populations.

References

- Ackermann, E. K. (1996). Perspective-Taking and Object Construction. In Y. Kafai & M. Resnick (Eds.), *Constructionism in Practice: Designing, Thinking, and Learning in a Digital World* (pp. 25-37). Mahwah, New Jersey: Lawrence Erlbaum Associates.
- Anderson, M. L. (2003). Embodied Cognition: A field guide. *Artificial Intelligence*, 149, 91-130.
- Anderson, M. L. (2007). How to study the mind: An introduction to embodied cognition. In F. Santoianni & C. Sabatana (Eds.), *Brain development in learning environments: Embodied and perceptual advancements*. Cambridge: Cambridge Scholars Press.
- Berthoz, A. (2000). *The Brain's Sense of Movement* (G. Weiss, Trans.): Harvard University Press.
- Byrne, P., & Becker, S. (2008). A Principle for Learning Egocentric-Allocentric Transformation. *Neural Computation*, 20, 709-737.
- Cooper, G. H., & Krishnamurthi, S. (2006). *Embedding Dynamic Dataflow in a Call-by-Value Language*. Paper presented at the 15th European Symposium on Programming, Vienna, Austria.
- Fadjo, C. L., Lu, m., & Black, J. B. (2009). *Instructional Embodiment and Video Game Programming in an After School Program*. Paper presented at the World Conference on Educational Multimedia, Hypermedia and Telecommunications, Chesapeake, VA.
- Kegan, R. (1982). *The evolving self: problem and process in human development*. Cambridge: Harvard University Press.
- Klatzky, R. (1998). Allocentric and Egocentric Spatial Representations: Definitions, Distinctions, and Interconnections. *Lecture Notes in Computer Science*, 1404, 1-17.
- Maloney, J., Burd, L., Kafai, Y., Rusk, N., Silverman, B., & Resnick, M. (2004). *Scratch: A Sneak Preview*. Paper presented at the Second International Conference on Creating, Connecting, and Collaborating through Computing, Kyoto, Japan.
- Ochs, E., Jacoby, S., & Gonzales, P. (1994). Interpretive journey: how physicists talk and travel through graphical space. *Configurations*, 2(1), 151-171.
- Papert, S. (1980). *Mindstorms: children, computers, and powerful ideas*. New York: Basic Books, Inc.
- Pea, R. D., & Kurland, D. M. (1984). On the cognitive effects of learning computer programming. *New Ideas in Psychology*, 2(2), 137-168.
- Robins, A., Rountree, J., & Rountree, N. (2003). Learning and Teaching Programming: A Review and Discussion. *Computer Science Education*, 13(2), 137-172.
- Seitz, J. (2000). The Bodily Basis of Thought. *New Ideas in Psychology*, 18, 23-40.
- Varela, F. J., Thompson, E., & Rosch, E. (1993). *The embodied mind: cognitive science and human experience*: Massachusetts Institute of Technology.
- Wilensky, U. (1999). NetLogo. Evanston, IL: Center for Connected Learning and Computer-Based Modeling, Northwestern University.
- Wilson, M. (2002). Six views of embodied cognition. *Psychonomic Bulletin & Review*, 9(4), 625-636.
- Wright, T. (2001). Karen in motion: The role of physical enactment in developing an understanding of distance, time, and speed. *Journal of Mathematical Behavior*, 20, 145-162.

Acknowledgments

We thank Gilbert Slade & Chadwick Wood for their hard work on this project. This work was supported by NSF Grant EEC-1025223.