NORTHWESTERN UNIVERSITY

VBOT: MOTIVATING COMPUTATIONAL AND COMPLEX SYSTEMS FLUENCIES WITH CONSTRUCTIONIST VIRTUAL/PHYSICAL ROBOTICS

A DISSERTATION

SUBMITTED TO
THE GRADUATE SCHOOL OF
NORTHWESTERN UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

for the degree

DOCTOR OF PHILOSOPHY
Field of Learning Sciences

By
Matthew W. Berland

EVANSTON, ILLINOIS
June 2008

ABSTRACT

VBOT: MOTIVATING COMPUTATIONAL AND COMPLEX SYSTEMS FLUENCIES WITH CONSTRUCTIONIST VIRTUAL/PHYSICAL ROBOTICS

by Matthew W. Berland

As scientists use the tools of computational and complex systems theory to broaden science perspectives (e.g., Bar-Yam, 1997; Holland, 1995; Wolfram, 2002), so can middle-school students broaden their perspectives using appropriate tools. The goals of this dissertation project are to build, study, evaluate, and compare activities designed to foster both computational and complex systems fluencies through collaborative constructionist virtual and physical robotics. In these activities, each student builds an agent (e.g., a robot-bird) that must interact with fellow students' agents to generate a complex aggregate (e.g., a flock of robot-birds) in a participatory simulation environment (Wilensky & Stroup, 1999a). In a participatory simulation, students collaborate by acting in a common space, teaching each other, and discussing content with one another. As a result, the students improve both their computational fluency and their complex systems fluency, where fluency is defined as the ability to both consume and produce relevant content (DiSessa, 2000). To date, several systems have been designed to foster computational and complex systems fluencies through computer programming and collaborative play (e.g., Hancock, 2003; Wilensky & Stroup, 1999b); this study suggests that, by supporting the relevant

fluencies through collaborative play, they become mutually reinforcing. In this work, I will present both the design of the VBOT virtual/physical constructionist robotics learning environment and a comparative study of student interaction with the virtual and physical environments across four middle-school classrooms, focusing on the contrast in systems perspectives differently afforded by the two environments. In particular, I found that while performance gains were similar overall, the physical environment supported agent perspectives on aggregate behavior, and the virtual environment supported aggregate perspectives on agent behavior.

The primary research questions are:

- What are the relative affordances of virtual and physical constructionist robotics systems towards computational and complex systems fluencies?
- What can middle school students learn using computational/complex systems learning environments in a collaborative setting?
- In what ways are these environments and activities effective in teaching students computational and complex systems fluencies?

ACKNOWLEDGMENTS

My advisor, Uri Wilensky, provided the means, the impetus, the instruction, and the support for this whole project. He encouraged me when I needed encouraging, he kept me on track when I needed it, and his keen insights forced me to think more deeply about every tacit assumption behind the project and its goals. This project stems, fundamentally, from my admiration of and firm belief in his work and the work of his advisor, Seymour Papert.

The two other members of my committee, Bruce Sherin and Ian Horswill, provided invaluable support and teaching throughout my graduate career. Both of them gracefully indulged my tendency to discourse at length, always steering me back to intellectually stimulating conversation.

Many thanks to the kind and brilliant people of the Center for Connected Learning and Computer-Based Modeling at Northwestern. Just a few of the people there that contributed immensely to my work and my sanity: Seth Tisue, Eric Betzold, Esther Verreau, Pratim Sengupta, Dor Abrahamson, Sharona Levy, Bill Rand, Michelle Wilkerson, Josh Unterman, Spiro Maroulis, Paulo Blikstein, Forrest Stonedahl, Matt Hellige, and Craig Brozefsky. The CCL was an incredible place for bouncing ideas off of some of the smartest, nicest people in the field. The CCL provided me with a meaningful critical research community without which the work would be greatly impoverished. Particular thanks are due to Michelle and Josh, who acted as primary co-facilitators in my deployments, and to Seth, whose advice was invaluable in building robust versions of VBOT.

Thanks to the teachers, the middle-school students, the other subjects, the undergrad aides, the Northwestern professors, and the many other people instrumental in getting this project working. Thanks to professors at Brown and ISS for inspiring me and giving me direction, especially Bob Cooper, Raoul Meyer, Leslie Kaelbling, Eugene Charniak, Len Tennenhouse, and Bob Scholes.

None of this would have been possible without the aid, support, and very hard work of my sister, Nicole, my parents, Lincoln and Nancy, and my friends Ericka and Michael; every one of them taught me much and helped me more. This text was read and reread ceaselessly by all of those involved. In particular, I believe that I am already committed to edit Nicole's first book. I owe all of them more hours than I could possibly tally. Any errors are surely my own, and any particularly clear passages are probably a result of my zealous editorial staff.

Finally, thanks to my wife, Leema, for everything always.

Chapter 1: Introduction

1.1 Prologue: Story of a student

Alicia is a terrible student: if she continues at her current level of performance, she will fail 8th grade. School is a necessary but unwelcome intrusion into her preferred lives in MySpace or The Sims. Within her social network, Lisa is the least successful academically. Although unhappy with her grades, her dislike of science precludes the enthusiasm necessary to improve her work. Part of the problem is that she recently transferred into her current school that proved more academically rigorous than her previous school. As her grades suffered, her self-confidence has eroded.

When we entered the classroom with our robots on the first day of the experiment, she loudly and uncomfortably giggled at the notion of piloting her own robots and was sheepish around the computers. By day 3, she was loudly voicing her opinions on how to design the control program, and, by day 5, her team, led mostly by her, won the bot soccer competition with a superior program. Furthermore, she described an understanding in networks and systems that she could not articulate previously. In Chapter 5, we will look at Alicia's performance, and how she used our environment to learn, what went right, and what went wrong.

This project is about stepping into the lives of students like Alicia and finding ways to use her motivation, interest, and intelligence to enjoy learning by equipping her with the lenses of complex systems and computer science. This project is designed to find the best ways to use those lenses with a complex game through physical robotics or virtual robotics.

1.2 Abstract

As scientists use the tools of computational and complex systems theory to broaden science perspectives (e.g., Bar-Yam, 1997; Holland, 1995; Wolfram, 2002), so can middle-school students broaden their perspectives using appropriate tools. The goals of this dissertation project are to build, study, evaluate, and compare activities designed to foster both computational and complex systems fluencies through collaborative constructionist virtual and physical robotics. In these activities, each student builds an agent (e.g., a robot-bird) that must interact with fellow students' agents to generate a complex aggregate (e.g., a flock of robot-birds) in a participatory simulation environment (Wilensky & Stroup, 1999a). In a participatory simulation, students collaborate by acting in a common space, teaching each other, and discussing content with one another. As a result, the students improve both their computational fluency and their complex systems fluency, where fluency is defined as the ability to both consume and produce relevant content (DiSessa, 2000). To date, several systems have been designed to foster computational and complex systems fluencies through computer programming and collaborative play (e.g., Hancock, 2003; Wilensky & Stroup, 1999b); this study suggests that, by supporting the relevant fluencies through collaborative play, they become mutually reinforcing. In this work, I will

present both the design of the VBOT virtual/physical constructionist robotics learning environment and a comparative study of student interaction with the virtual and physical environments across four middle-school classrooms, focusing on the contrast in systems perspectives differently afforded by the two environments. In particular, I found that while performance gains were similar overall, the physical environment supported agent perspectives on aggregate behavior, and the virtual environment supported aggregate perspectives on agent behavior.

The primary research questions are:

- What are the relative affordances of virtual and physical constructionist robotics systems towards computational and complex systems fluencies?
- What can middle school students learn using computational/complex systems learning environments in a collaborative setting?
- In what ways are these environments and activities effective in teaching students computational and complex systems fluencies?

1.3 Methods & Data

All of the studies that I will present in this dissertation took place in classes using the VBOT constructionist virtual/physical robotics learning environment that I designed. In these classes, students program within the VBOT environment to compete or collaborate with one another in order to achieve larger gaming goals. Students in the classes used the VBOT system for one class period a day for five days. In these classes, students would use the system for the majority of the time, discuss the day, and conclude by recording their interpretations of the activity and lessons on a questionnaire. These classes were videotaped, and log data was collected, including all mouse/keyboard input to the system. Furthermore, interviews and written evaluations were conducted before, during, and after the activities. The research presented in this dissertation was conducted in classrooms using VBOT activities, described in Chapter 4, in two Chicago Public Schools. The students participating in these studies were in the 8th grade. These studies were conducted in the design research tradition (Edelson, 2002), in that the researchers were active participants in a classroom, iterating active design, data collection, and analysis.

1.4 Results & Findings

The core results of this study suggest that reasonably equivalent virtual and physical constructionist robotics environments support computational and complex systems fluencies differently. However, students across contexts exhibited similar learning gains overall. Moreover, gender, school, teacher, and previous performance did not have any appreciable effect on performance; all students showed significant improvement in at least one measured aspect of computational or complex systems fluencies. This finding builds on previous constructionist findings in the individual fluencies, such as those in Papert (1980), Wilensky & Stroup (1999a), DiSessa (2000), and Harel & Papert (1990), that suggest that students are better able to build computational or complex systems fluency when supported by an active, social, constructionist activity and environment. This work also provides a counterpoint to some previous negative research. Negative research in complex systems fluency, such as that of Hmelo-Silver & Pfeffer (2004) and Chi (2005), argue that middle-school students have trouble with the model-building aspects of systems understanding. Negative research in computational fluency, such as that of AAUW (2000) and Cuban (1985), suggest that thinking about problems computationally or writing computer programs is inherently difficult for most middle-school students. The work presented in this thesis presents data that are contrary to those negative claims.

As all groups of students exhibited learning gains, this research investigates the differences in those gains that emerged when comparing the virtual and physical learning environments. Many of these differences are contingent on differences of perspective. That is, students in virtual robotics classes are more likely to understand agents in terms of the emergent properties of the systems in which they participate, and students in physical robotics classes are more likely to understand the emergent behavior of a system in terms of the agents of which it consists. As a result, although students in the two sets of classes performed similarly overall, their different perspectives created a different distributions of scores, circuits, and different system behavior. For example, students in the virtual robotics contexts excelled at predicting and organizing the behavior of the aggregate robot "swarm" while students in the physical robotics contexts excelled at building individual behaviors which performed well in complex environments. Chapters 5 and 6 explore why these cross-contextual differences emerged, and Chapters 7 and 8 discuss the ramifications of the differences for further learning environment design. Based on this data, this work suggests that virtual and physical robotics provide effective learning environments for targeting different aspects of computational and complex systems fluencies.

1.5 Contributions

The primary goal of this work is to determine how best to motivate complex systems and computer science fluency using constructionist tools. In this section, I describe what readers versed in various academic traditions might expect to gain by reading this dissertation.

Learning sciences researcher fluent with constructionism and complex systems research: Through our findings, hopefully you can better understand how to use constructionism to teach complexity science, both strengths and weaknesses. This work should be able to give concrete recommendations towards how to optimize methods so as best design tools for teaching and learning complex systems and computational fluency. This work addresses the question of relative benefits and deficits of virtual and physical learning environments for both constructionist and complex systems learning.

Learning sciences researcher fluent with constructionism but not complex systems: This work fits into a history of constructionist work with complex systems done by Wilensky (2001), Resnick (1994a), etc. It should provide you with a new horizon on which to understand the ways in which constructionism can be used today to motivate learning. In this work, we describe a novel constructionist learning environment, and our results can be reapplied towards the design of a variety of learning environments for science and computational fluency. This work has direct applications to constructionist debates about the relative value of virtual and physical learning environments (see Resnick & Ocko, 1991, for one example).

Learning sciences researcher fluent with complex systems but not constructionism:

Complex systems methods and theory have been gaining prominence in scientific research, but many have claimed that it can be hard to teach to younger students or even that one should not try to teach it to non-researchers (Chi, 2005). This work presents significant evidence that it is not only fruitful but also relatively easy to teach basic complex systems fluency. Presented are both methods and evidence for the benefits and deficits of teaching complex systems concepts with a variety of tools.

Learning sciences researcher fluent with design research but neither complex systems nor constructionism research:

This work uses Wolfram (2002) and Bar-Yam (1997), among others, to frame the various ways in which complex systems have changed that ways that people are doing science today. We are hoping to leverage design research methods in this work to determine how best to teach students to think and learning in novel environments. Papert (1980) motivates the need for constructionist methods towards teaching students to handle the kinds of issues in the world today. In this thesis, I argue that we can leverage constructionist and complex systems methods to best teach both, rather than trying to teach the respective scientific content in a more traditional method.

Secondary school science or math teacher:

Both computational fluency and complex systems science fluency methods and tools are described and explained in this work. Chapter 2 discusses the value of several free or inexpensive tools to teach complex systems or computation fluency, and Chapter 3 explains what you can expect to see students learning by using constructionist tools such as VBOT. Furthermore, presented is a novel learning environment (VBOT) for teaching complex systems and computational fluencies which can be used free of charge in any wired classroom. Included in this work are detailed descriptions and instructions on how we used the tools and activities and what students learned through the activities.

Computer science researcher fluent with artificial intelligence and robotics:

VBOT presents a new programming language and computational fluency learning environment using behavior-based robotics. In this work we discuss the ways that beginning students can best understand various potentially difficult computer science concepts through their use. Furthermore, we address the relative value of virtual and physical robotics, although this perspective does not necessarily evaluate any usefulness for researchers who do formal robotics.

1.6 Thesis Index

“Chapter 2: The Structure, Syntax, and Design of VBOT” presents an overview of the tools, activities, and programming language that I designed for this project. It describes the VBOT system, in both overhead design and method. This is the most technical chapter, but it is necessary for understanding the rest of the project. In “Chapter 3: Related Literature”, there is an overview of relevant literature as well as a discussion of how this project fits into existing literature. Constructionism, complex systems research, design research, and learning sciences research are discussed. In “Chapter 4: Context, Methodology, and Data Collection”, I describe the environment in which we performed the studies, and the methods we used to gather data. I describe the several pilots and studies performed, with descriptions of school settings and teachers. “Chapter 5: Student Tinkering and Student Sharing” and “Chapter 6: The Virtual and the Physical” describe our various quantitative and qualitative results. We present the data and contextual data for understanding the data. Furthermore, I evaluate and analyze the results and data. In “Chapter 7: Design Principles,” I describe the iterative design process and the various incarnations of the VBOT system, with in-depth analysis of what we learned from each iteration, how the design changed, with particular attention paid to how we changed the design. In “Chapter 8: Conclusions and Future Directions,” I provide prescriptions for use of the findings and suggestions for further work in the field.

Chapter 2: The Structure, Syntax, and Design of VBOT

2.1 Introduction

Throughout the thesis, I will be making consistent reference to the VBOT programming environment designed specifically for this project. It is a full featured networked participatory simulation, and it connects to a NetLogo-based participatory simulation system called HubNet (Wilensky & Stroup, 1999b). As described in Chapter 1, VBOT lets users program a simple real-time interface to control a virtual robot ("vbot") in a joint virtual space with other vbots (see Chapter 2). Users follow the vbots on the associated NetLogo screen, while watching their vbot's behavior and history on their action panels. The action panel has three features: it shows the position of the vbot on a radar-like display; it shows the positions of the other students' vbots; and it shows a tapering history trail for the associated vbot. At the most basic level, the vbot can perform only one task: movement. The vbot "senses" information from the virtual world around it and moves in programmatic reaction to that information. This chapter describes how to program in VBOT to control that reaction, how to choose and mediate movement, and the various ways in which one can manipulate the software to change the behavior of the vbot. Furthermore, it describes the relationship between the VBOT software and the physical robots used in the simulations and interventions discussed elsewhere in the thesis. This chapter is designed to inform the reader of the structure, syntax, and usage of the VBOT system. There is also a short tutorial on how to begin building VBOT programs. As with any programming language, learning to program with VBOT can be a difficult process, so throughout this chapter, basic understanding of computer programming is assumed.

Basics of VBOT Software

VBOT is built on top of the NetLogo multi-agent modeling environment (Wilensky, 1999) and the HubNet networked learning architecture (Wilensky & Stroup, 1999b) used for participatory simulations. In these simulations, participant-users interact in a social space (such as a classroom) while virtual agents, that the participants control, interact in a virtual space, which is typically in a shared virtual environment visible to all participants. A primary design goal NetLogo/HubNet is to make multi-agent modeling and programming accessible to a wide audience without sacrificing the ability to make detailed scientific models of complex systems. VBOT uses this base to create an immersive collaborative robotics modeling and programming environment designed primarily for use in middle school. VBOT consists of 1) an interface through which users build the control system of their personal virtual robot (**vbot**) in real-time (i.e. changes in the program are reflected immediately in the behavior of the vbot), 2) a shared space in which virtual robots exist, and 3) activities for users using the social space.

As just described, the VBOT system is built on top of the NetLogo multi-agent modeling environment (Wilensky, 1999) and the HubNet participatory simulation architecture (Wilensky

& Stroup, 1999b). HubNet is a system built on top of NetLogo called a “participatory simulation environment” in which individual participants can each control an agent ("turtle") in a NetLogo model. For instance, in the HubNet model Gridlock (Wilensky, 1997b; Wilensky & Stroup, 1999a), many users each control a stoplight a virtual space consisting of a set of streets, streetlights, and cars. Users change their lights to red or green in order to maximize the flow of cars through the traffic lights. Using the language of agent modeling, each stoplight is an agent in a complex system from which the flow of traffic in the city emerges. The complex system here is one in which cars, lights, and roads interact to create a dynamic equilibrium.

Building upon the HubNet and NetLogo framework, the VBOT system enables each user to control one agent in a model. Instead of controlling the agent directly, as one might do in a HubNet simulation, or controlling the agents through general rules for classes of agents, as one might do in a NetLogo model, each participant builds a “behavior circuit” which controls the actions of his/her agent virtual robot. A behavior circuit consists of a set of navigation and behavior rules designed in a format of a simple electronic circuit.

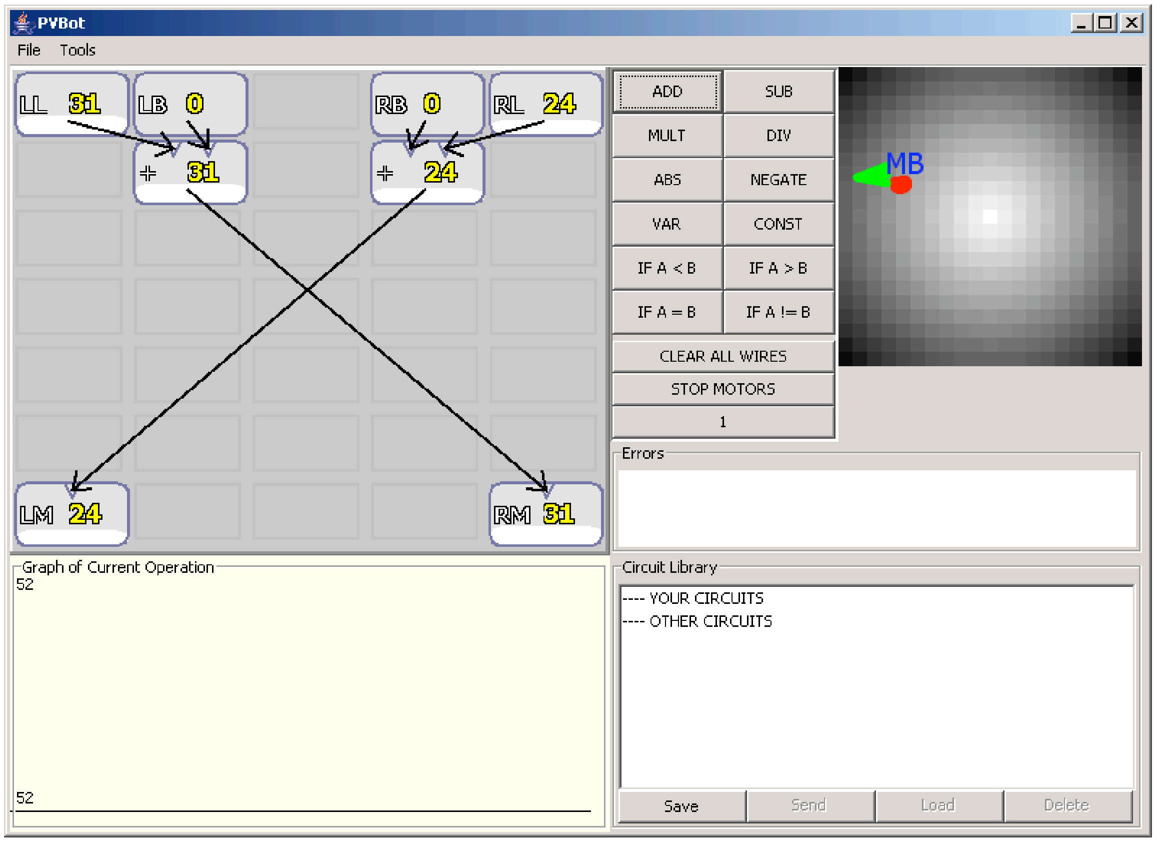


Figure 2-1: The VBOT client interface

In a VBOT activity, each student in a classroom uses the VBOT client interface (Figure 2-1) to design circuits that control the behaviors of their virtual robots (vbots). These circuits are called behavior circuits and consist of a set of navigation and behavior rules designed to follow the metaphor of a simple electronic circuit. While building circuit-based programs has traditionally been an undergraduate-level activity, recent studies have shown that high-school and middle-school students can productively use such systems for learning systems-based thinking (see Hancock, 2003). In the VBOT system, each student's vbot corresponds to a virtual robot "agent" in a shared virtual space (as seen in Chapter 2). During activities, students can build behavior circuits, save behavior circuits, load ("run") or share behavior circuits that they have built, or load behavior circuits that have been shared by others.

Beyond creating and manipulating the behavior circuits the client interface (or Action Panel) supports users in monitoring their vbots' progress through three features: it shows the position of the user's individual vbot on a radar-like display; it shows the positions of the other users' vbots; and it shows a tapering history trail for the associated vbot.

Computational Paradigm

This chapter discusses VBOT as a programming language and the syntax and structure of programs ("circuits") written in the language. The fundamental metaphor of the language is functional with some procedural elements. This section describes the basic structure of these distinctions and how VBOT program flow follows and deviates from these structures.

Procedural languages, like C or Java, are sequences of statements that execute in a predefined order, based on a given logic. For example, programs written in a procedural language often follow logic similar to:

1. Execute instruction A
2. If user has selected interface element X, go to step 1

3. If user has selected interface element Z, go to step 5
4. Execute instruction C
5. Execute instruction D
6. Go to step 1.

The vast majority of programs written in the world today are written in a procedural language. The most common structure of such programs involves a simple loop in which the set of instructions is repeated as long as the program remains active. This loop can be seen in Step 6 of the sample program, above.

Functional languages, such as LISP, present the most common competing program metaphor. In a functional language, every statement is a function, much like a mathematical function (e.g., "$y = x^2$"). Each function returns a value, and these values provide the input to other functions. Programs written in a function language might have a logic similar to this:

- f(x,y) = x * (y + 5)
- If user has selected interface element X, g() = 2; otherwise, g() = 3.
- Evaluate f(g(),g()))

In this simple example, f(g(),g()) would always evaluate to 14 or 24. The salient difference here between a procedural and functional language is that program logic is not designed to be a sequenced order of events. Rather, the functional metaphor fosters a need to consider the evaluation and prerequisites of relevant functions.

VBOT presents users with a combination of these programming approaches. As with functional languages, in VBOT, the run order is determined by data flow; the simulation determines the exact order of execution. Each independent path from a sensor to a motor can be considered to be a parallel circuit. Each operation, however, is run functionally. There are always at least two functions in a VBOT circuit, the left motor (as in line 1, below) and the right motor. Each motor is then evaluated as a function where "leaf operations" (usually a sensor or a constant, below line 1c) evaluates to a number. Leaf operations consist of any operation in which the static value is determined external to the program logic, either by the hardware or the user. For instance, in Figure 2-2, the LM (left motor) function is evaluated by multiplying the constant 3 and the RL (right light) sensor (here evaluated in line 1a) :

1. $f_{LM}()$ = Multiply(f_{RL}, 3)
2. $f_{RL}()$ = 90, determined externally
3. $f_{LM}()$ = Multiply(90 , 3)
4. $f_{LM}()$ = 270

The function $f_{LM}()$ is evaluated every tick, and evaluating the function requires recursive traversal down the "function tree." That is, $f_{LM}()$ evaluates to the function Multiply($f_{RL}()$,3), which in turn requires the evaluation of $f_{RL}()$. This evaluation tree is similar to a function program.

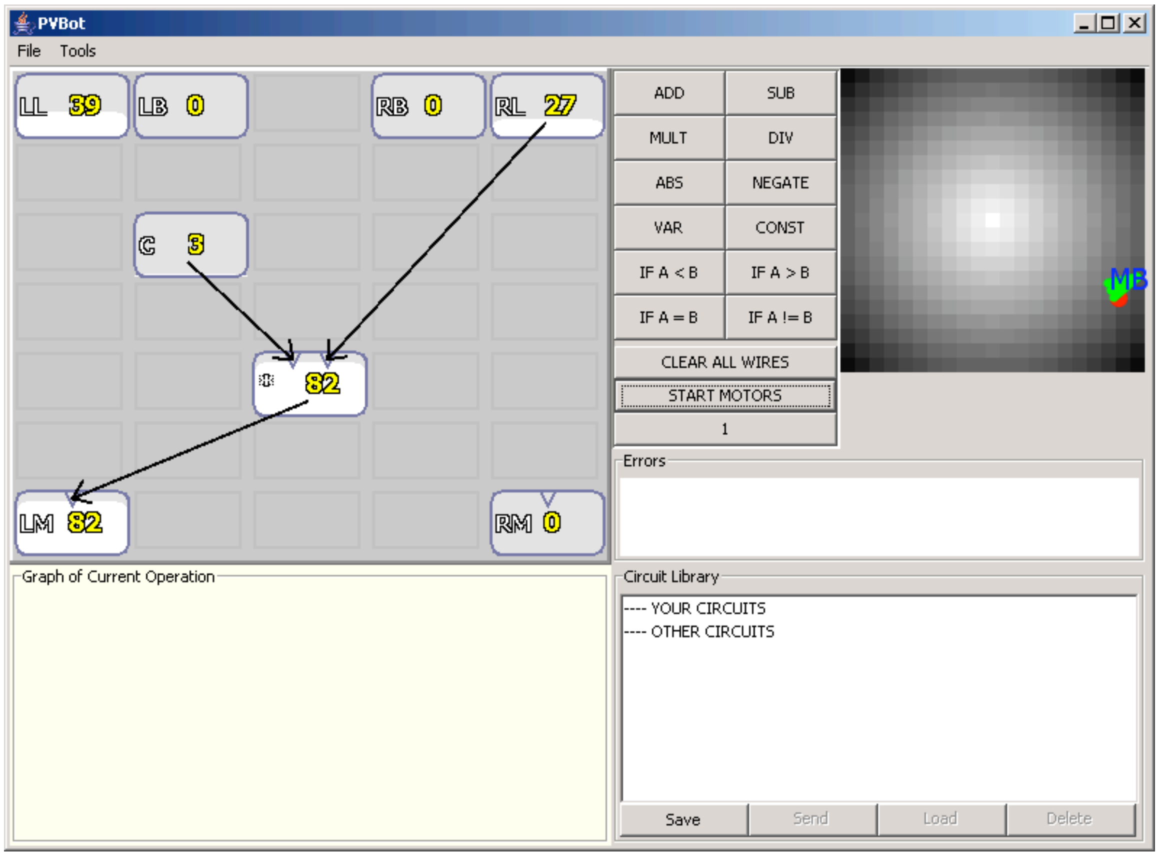


Figure 2-2: Addition in VBOT

2.2 Syntax

While VBOT's programming paradigm might seem complex, it is important to note that this paradigm is transformed into a visual programming environment when users work with it and that they are largely unaware of the functional characteristics of the language. This transformation to a visual language occurs through the building blocks (objects) of the VBOT syntax. These building blocks include these fundamental building blocks (objects): operations, wires, motors, sensors, and one breadboard. A circuit is defined as a mapping from sensors to motors using wires and operations on a breadboard. Users add operations to the breadboard; they connect those operations to motors with wires. In this way, users create mathematical functions that take the numeric input from the sensors and produce a numerical output for the motors. However, programs can get increasingly complicated with the addition of more variables, sensors, branching logic, and rudimentary looping. The following sections describe the syntax objects more carefully.

Breadboard

The breadboard is the space in which all of the motors, sensors, operations, and wires are connected. Its metaphor is a real circuit breadboard. Solid-state electronics are, in general, combinations of transistors, inputs, and outputs on a breadboard. For example, a digital watch is a set of output LEDS connected to a simple time-keeping chip (made of solid state transistors). In the same way, the VBOT motors (show below as LM and RM, at bottom) are the outputs, which produce all movement and stopping of movement in the vbot. The vbot will simply drift without impetus from the motors as an object in a minimally frictional space.

All connections between objects happen on the breadboard, and the breadboard itself works through flow cycles (or "ticks"). During the course of each tick, the breadboard evaluates all terminal operations and motors using recursive evaluation. Further explanation of recursive evaluation and an example is given below in the section addressing flow in VBOT.

Motors

The motors impel movement for a given vbot. As described earlier, the metaphor of a vbot is a small robot car with two rear wheels. Each wheel is powered by its own motor. The left rear motor, if powered without the right rear motor, will turn the car right. The right rear motor moves the car left. When both motors are powered equally, the car moves forward in a straight line (see Section 2.3, Example Programs, for more examples).

In middle-school implementations, the breadboard provided only two motors, left rear and right rear. We did this to align the virtual programming experiences with the physical in which the number of motors was limited by the physical structure of the robots. However, it is possible to make any number of motors control any number of actions in VBOT. For instance, one could create a VBOT extension in which a motor controls a grappling claw on the virtual robot.

Each motor has a numerical range from 0 to 100% (although there is no hard cap at 100). In our experiments, 0% motors were brakes and the power scaled linearly to the movement of the motor up to 100%; after 100%, there was a logarithmic power increase, as the vbots could become too fast to see clearly if an incorrect circuit was built. The space past 100% could be thought of as "overdrive," and users were taught that overdrive produces somewhat erratic behavior.

Wires & Flow

Wires create flow within a circuit in a manner similar to water pipes. For example, using the pipe metaphor, supposing any segment of pipe can pass a maximum of 100 L per second of water through at any given moment. Plumbers can connect the pipes however they wish, but all flow through the pipes is directed. The water stems from a reservoir source and flows to an outlet (or "sink," using flow terminology). VBOT wires are much like these pipes, where sensors provide the source of the signal, and the motors provide a sink. The operations are mid-points in the stream and provide both checkpoints for and modifications to the data. Using the water pipe metaphor, the operations might represent a midway treatment plant, a water filter, a cooling system, a pipe interchange, or anything that might reasonably affect the water flow midstream. VBOT treats the signal from the sensors as a source of energy (like water).

In Figure 2-2, the left light sensor (LL) and a constant (here, 3) connect to a multiplication operator ("*"). They provide the reservoir of "energy," which is then communicated to the multiplication operator. There is, as reflected in the number, multiplication in that operator. This multiplication is directed, meaning that the energy flow is always unidirectional; the output of the multiplication operator is not used by the light sensor in any way. That is, the light sensor provides the signal energy for the circuit, so it does not use energy from other operations. The arrows on the wires show the direction of the flow.

The flow metaphor of programming has been used in many scenarios, as described in the Chapter 3. The process of flow in VBOT presupposes a unidirectional current; it ends somewhere other than its origin and is repeated many times on a fixed schedule. This is the implicit logic of the vast majority of end-user computer programs.

Sensors (& Noise)

This section describes the sensors and the “noise” in the sensors’ readings. This dissertation compares the experience of students programming physical and virtual robots. Thus far in this chapter, we have focused on the client interface that is consistent across environments. However, there exist distinct differences in how the virtual and physical sensors work, thus, we describe these differences in this section.

A sensor is any operation that outputs a number corresponding to some input from the play space (such as light or heat). In practice, there were 4 categories of sensors: light, “bot”, “bump”, and “VAR.” Light sensors, for example, return a light reading between 1 and 100. The virtual light sensor is defined as the measure of the “virtual light” at any given point in the “virtual space.” The virtual “bot” sensor (LB and RB) outputs the percentage of vbots in proximity to the sensor. The physical “bump” sensor registers any activation of the physical touch sensor on the physical vbot. The “VAR” sensor outputs a given variable that is updated from the server, and it can represent any value that the facilitator chooses.

It is not advantageous for the light sensor readings to be "perfect" (unerring) as that will often create oscillatory behavior in simple circuits. Furthermore, since we are working with the paradigm of physical robots as well, virtual circuits would not map appropriately to physical behavior, as virtual circuits “over-trained” to perfect information do not translate to the physical world. To that end, the virtual vbot sensors contain some randomness. A variable and normally randomized amount of "nonsense" is added to any given sensor reading. The analysis in Chapter 6 deals in part with the different reactions of different groups to perceived randomness in physical and virtual robots.

Network Dynamic

The network layout of the VBOT system is client/server. A server, running both NetLogo/HubNet and a VBOT NetLogo server extension, takes all input from each user's client, and then sends all updated information to all clients. In this way, all clients have synchronous information with which to update the Action Panel. Although this creates unnecessary overhead, the small amount of data transferred per client and the relatively small number of clients (less than 50) makes this a viable network architecture. In our real world tests with up to 40 clients, there was negligible network lag.

Operations

There are two categories of operations: numerical and logical. All operations are visible in the buttons on Figure 2-1 such that most buttons correspond to an operation.

Numerical operations compute arithmetical outputs of several numerical inputs. The most common operations are adding (heretofore "ADD" or "+"), subtracting ("SUB" or "-"), dividing

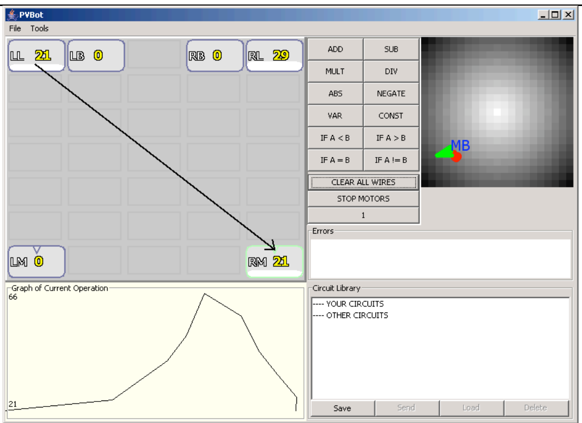
("DIV" or "/"), and multiplying ("MULT" or "*"). A constant, which is any user-defined whole number, is a numerical operation in name only, as it represents a static whole number. When using the VBOT system, numerical operations are taught first, and a typical use case for a beginner would involve increasing the speed of a vbot by adding 10 to a given input and connecting that ADD operation to the motors.

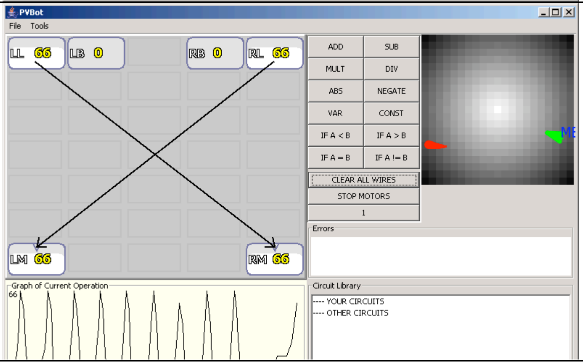
Logical operations compare two values to produce an output. An example of a logical operation is the "IF A<B" operation. Similar to the last example, a typical beginner will build a circuit that activates LM when sensor LL reads less than 50%. In this case, when sensor LL is less than 50%, the left motor (LM) will be activated at half power. Otherwise, it will remain inactive.

Several examples of numerical and logical operations provided in the sample programs that follow.

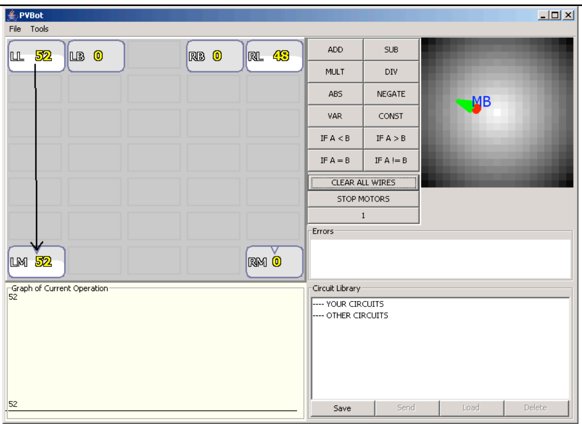
2.3 Example Programs

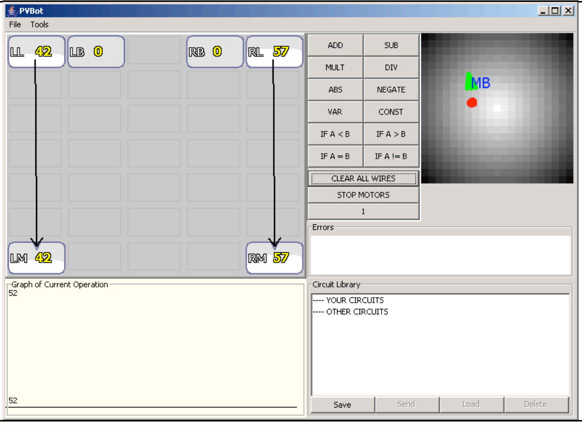
Love (adapted from Braitenberg, 1984)

Description	Code	Interface
ACTION: Connect the left light sensor to the right motor. EFFECT: The vbot will spin in a circle, since only its right motor is activated. The more light that the left light sensor receives, the faster the vbot will turn.	RM=LL;	 PVBot File Tools LL 21 LB 0 RB 0 RL 29 LM 0 RM 21 ADD SUB MULT DIV ABS NEGATE VAR CONST IF A < B IF A > B IF A = B IF A != B CLEAR ALL WIRES STOP MOTORS 1 MB Errors Graph of Current Operation 66 21 Circuit Library ---- YOUR CIRCUITS ---- OTHER CIRCUITS Save Send Load Delete

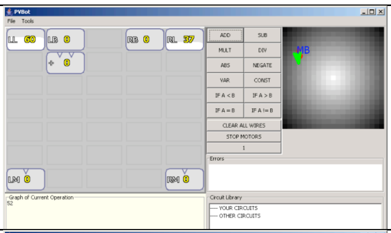
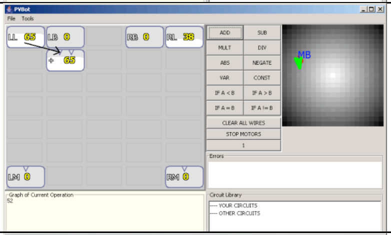
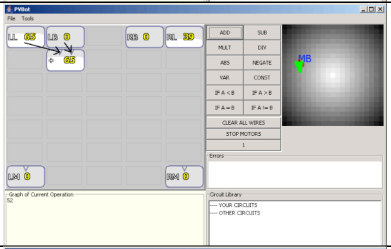
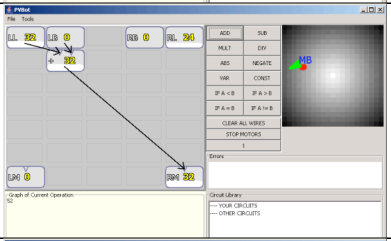
ACTION: Connect the right light sensor to the left motor.
 	
EFFECT: The vbot will seek the light, turning towards it whenever possible.
	RM=LL;
	LM=RL;
		 PVBot
		File Tools
		LL 66
		LB 0
		RB 0
		RL 66
		LM 66
		RM 66
		ADD
		SUB
		MULT
		DIV
		ABS
		NEGATE
		VAR
		CONST
		IF A < B
		IF A > B
		IF A = B
		IF A != B
		CLEAR ALL WIRES
		STOP MOTORS
		1
		Errors
		Graph of Current Operation
		66
		Circuit Library
		---- YOUR CIRCUITS
		---- OTHER CIRCUITS

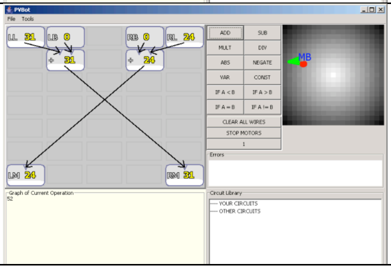
Fear / Explorer (from Braitenberg, 1984)

Description	Code	Interface
ACTION: Connect the left light sensor to the left motor. EFFECT: The vbot will spin in a circle, as only the left motor is activated.	LM=LL;	 PVBot File Tools LL 52 LB 0 RB 0 RL 48 LM 52 RM 0 ADD SUB MULT DIV ABS NEGATE VAR CONST IF A < B IF A > B IF A = B IF A != B CLEAR ALL WIRES STOP MOTORS 1 MB Errors Graph of Current Operation 52 52 Circuit Library YOUR CIRCUITS OTHER CIRCUITS Save Send Load Delete

ACTION: Connect the right light sensor to the right motor.
 	
EFFECT: The vbot will turn away from the light whenever possible.
	LM=LL;
	RM=RL;
		 PVBot
		File Tools
		LL 42
		LB 0
		RB 0
		RL 57
		LM 42
		RM 57
		ADD
		SUB
		MULT
		DIV
		ABS
		NEGATE
		VAR
		CONST
		IF A < B
		IF A > B
		IF A = B
		IF A != B
		CLEAR ALL WIRES
		STOP MOTORS
		1
		MB
		Errors
		Graph of Current Operation
		52
		52
		Circuit Library
		YOUR CIRCUITS
		OTHER CIRCUITS
		Save
		Send
		Load
		Delete

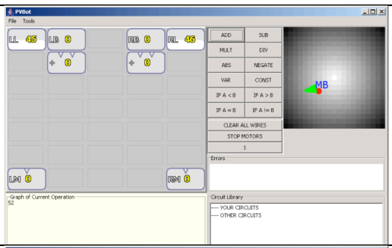
Values / Knowledge (from Braitenberg, 1984)

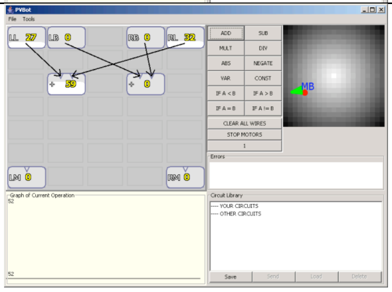
Description	Code	Interface
ACTION: Create an ADD ("+") operation box. EFFECT: The vbot motors are inactive, so the vbot will drift.	ADD = 0 ;	
ACTION: Connect the left light sensor to the ADD box. EFFECT: The vbot motors are inactive, so the vbot will drift.	ADD = LL + 0 ;	
ACTION: Connect the left bot sensor to the same ADD box. EFFECT: The vbot motors are inactive, so the vbot will drift.	ADD = LL + LB ;	
ACTION: Connect the ADD box to the right motor. EFFECT: The vbot will spin in a circle in this case, as only the right motor is activated.	RM = LL + LB ;	

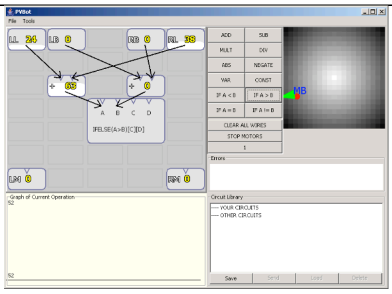
ACTION: Repeat these steps for the associated right sensors and left motor.
EFFECT: The vbot turns towards either light or other vbots.
	RM = LL + LB ;
	LM = RL + RB ;
		

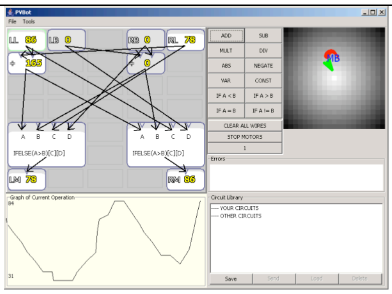
Instincts (from Braitenberg, 1984)

Description	Code	Interface

ACTION: Create two ADD operation boxes.
 	
EFFECT: The vbot motors are inactive, so the vbot will drift.
	ADD1 = 0;
	ADD2 = 0;
		

ACTION: Connect both light sensors to the first ADD box (ADD1), connect both bot sensors to the second ADD box (ADD2).
EFFECT: The vbot motors are inactive, so the vbot will drift.
	ADD1 = LL+RL;
	ADD2 = LB+RB;
		

ACTION: Create an "IF A>B" operation box. Connect ADD1 to the A slot in the IF box, and connect ADD2 to the B slot in the IF box. EFFECT: The vbot motors are inactive, so the vbot will drift.	IF(ADD1>ADD2) = 0	

ACTION: This is wired such that each motor has its own IF box to determine its numerical input.
EFFECT: If the light sensors are more active than the vbot sensors, the vbot will turns towards light. If the vbot sensors are more active than the light sensors, it will seek out other vbots.
	IF(ADD1>ADD2)
	THEN
	[LM = RL ;
	RM = LL ;]
	ELSE
	[LM = RB ;
	RM = LB ;]
		

2.4 VBOT User Interface

Graphs

The graph window records the value of a "box" on the breadboard (operation, motor, or sensor) at every time-tick, and plots the data point in the rectangular window. Users can select which “box” to graph and change the graph throughout the runtime. When the graph trails off the right of the graph, it rescales the graph by roughly 20% to fit all data. It continues to rescale the graph as such until the graphed "box" is changed or the graph is reset.

Using the PVBOT graphing system, users can watch the output of any given "box" on the breadboard. For instance, they could watch a graph of the left light sensor's output to determine the maximum and minimum values of light on a certain path in the world. Another use of the graphs might be to determine when and how often logical operations (e.g., “IF A<B”) are triggered to check program logic. If users have created a logical operation that is triggered inappropriately, graphing the function can reveal the error quickly. The graphs were underutilized by most users, as they are mostly useful for debugging or advanced use. Although classroom interventions did not introduce the topic of formal debugging, a handful of students used the graphing functions towards formal debugging.

Saving and loading

Users can both save and load ("run") their circuits as necessary. VBOT contains a "save/load" panel in which users can see all of the circuits that they have saved on their own computer. To save a file, users can either use the File menu (with key command, as normal on a given Windows/Mac/Linux system) or the "save" button right in the save/load panel. Most students in our implementations opted to use the quick buttons. To load a circuit, one can simply double-click or click the file and the click the "load" button.

During a given activity, users have the opportunity to create and employ multiple circuits. They can do this by saving the circuits with mnemonic names (such as "chaser" and "escape") and double-clicking the circuits to switch when necessary.

Button Panel

The client interface also (shown on Figure 2-1) contains a Button Panel. The majority of these buttons are used to place operations on the breadboard. Beyond these operations users have access to “CLEAR ALL,” “START/STOP” and “role” buttons. The “CLEAR ALL” button resets the breadboard. Since circuits are often small, users often used the clear all button in lieu of destroying one wire or operation at a time.

The “START/STOP” button is the most utilized debugging feature for vbots. If a user wants to immediately cease all operation of a virtual vbot in order to consider a circuit, this button stops all client-side computation. This means that the vbot is frozen on the virtual shared space – until the user selects the ‘START/STOP’ button again. Once stopped, the user can build and test the behavior from a predetermined position by adjusting their breadboard and pressing “START/STOP” again to resume action. Adults used this much more frequently than students.

The “role” button is a "catch all" button for the interface. In many vbot simulations, a user must input a choice unrelated to circuit building, and this button facilitated user choice. For instance, in one implementation of Moon-Tag (see below), students changed between the roles of doctor, monster, or worker simply by clicking the role button. At the beginning of each run, a facilitator identified the button’s role for the round (for instance, "1 = worker, 2 = monster, 3 = doctor") and inputted that into the NetLogo simulation on the server. Then, throughout the round, students would change roles as desired by selecting the “role” button. Another common use of the “role” button in VBOT is simple polling, in which one VBOT-HubNet simulation monitors the value of each user’s role button.

Error panel

It is impossible in this paradigm to create an illegal connection on the breadboard; any attempts will result in the wire’s disappearance. The error is then added to the error panel. For instance, if a user was to try to create a wire from a motor to a sensor (with the semantic being that the output of the motor is fed into a sensor, which does not work in this paradigm), the wire would disappear and the associated error message would read "sensor LL takes no inputs."

2.5 VBOT Activities

Orbit & Flocking Activity

In Orbit activity, the first instruction given to the group was "everybody move to the middle." Students then build circuits designed to move the bot to the middle of the screen. For virtual VBOT (VVBOT), the middle of the screen is marked by a "light source" which radiates “light” outward from white to black (see Figure 2-3 below). In physical VBOT (PVBOT), the light sources were simple lamps. In this case, a correct circuit is two crossed wires connecting each light sensor to its opposite-side motor (the “Love” circuit described above). In the Love circuit, the right light sensor causes the left motor to be activated (and vice-versa). The activity progresses once all students complete this circuit. The Love circuit creates a crude “orbit” around the light in both physical and virtual VBOT.

Once all students’ bots are near the light, the instruction is given for the students to “flock together outside the light." This commences the Flocking activity. The goal of the Flocking activity is to create a stable group (“flock”) of bots that travels together.

In VVBOT, creating a stable flock requires “bot sensors” (described above) to find the other bots. Once the bots are flocked, they need to stay out of the light, of which there are several viable strategies. One such strategy is to nominate certain students to avoid light and summarily flock around those students. Another strategy is for each student-group to create a more complex circuit that enables both light-avoidance and flocking. Over the course of our enactment, the students designed many such strategies.

In PVBOT, the task is more difficult. It requires the students to self-organize navigation patterns, as the sensors are unreliable. Over the course of the enactment, students designed several strategies to deal with the situation. One particularly good strategy was to slow the robots upon any outside bump, while simultaneously avoiding light. Another strategy was simpler, but perhaps more novel: all students design a circuit that avoids light, but they also manipulate the starting positions of the robots.

Simple-Tag Activity

This activity was only enacted in the virtual class. In Simple-Tag, the goal for the class was to maximize total “tags”. A “tag” occurs when a “tagged” vbot was co-located with an “untagged” vbot. The untagged vbot is then tagged. Tags are marked by a red dot attached to tagged vbot that shows up on the shared screen. A good strategy is to create a vbot circuit that causes one’s vbot to find other vbots more quickly than the rest of the classroom. The students designed several circuits that accomplished this task.

Bot Soccer Activity

This activity was enacted only in the physical class. For Bot Soccer, the students are divided into two teams. Each team, in this case, consisted of three robots, and the teams were allowed to talk to each other to decide strategies. The two teams were placed facing each other, in a line, with their “goal” being a light located behind the starting line. The goal is to drive the ball into the other team’s light-goal. This engenders several interesting strategies. Some teams designed complex strategies involving goalies and forwards. Other teams designed strategies that mostly consisted of a forward-attack rush. The success of a team was primarily contingent on their coordination. One team’s strategy is discussed further in Chapter 6.

Moon-Tag Activity

In Moon-Tag, the goal for the class was to collect “moon-rocks” from the edges of the screen and deposit them in the center of the screen (see Figure 2-3). The students switched between three roles (doctor, monster, and worker) using a button on their VBOT client. A doctor made any other vbots in the vicinity “unsticky,” a monster made other bots sticky, and workers became sticky or unsticky as they encountered doctors or monsters. As the agents traveled across the shared virtual screen, they could pick up the moon-rocks only if they were “sticky,” and dropped the moon rocks only when they were no longer sticky. As the goal of the activity was to drop moon rocks in the middle of the screen, the goal for the class was to ensure a steady flow of workers passing through the edges and the middle, becoming sticky on the edges to pick up errant moon rocks and becoming unsticky in the middle of the screen in order to drop any rocks that might have been picked up. Students could change roles freely. Students whose vbots were in monster role gained points for making workers sticky near the edges of the screen, making doctors unsticky near the middle, and causing workers to drop moon rocks near the middle. The class, however, only got points for the moon-rocks collected in the middle of the screen (or the sum of all worker-points).

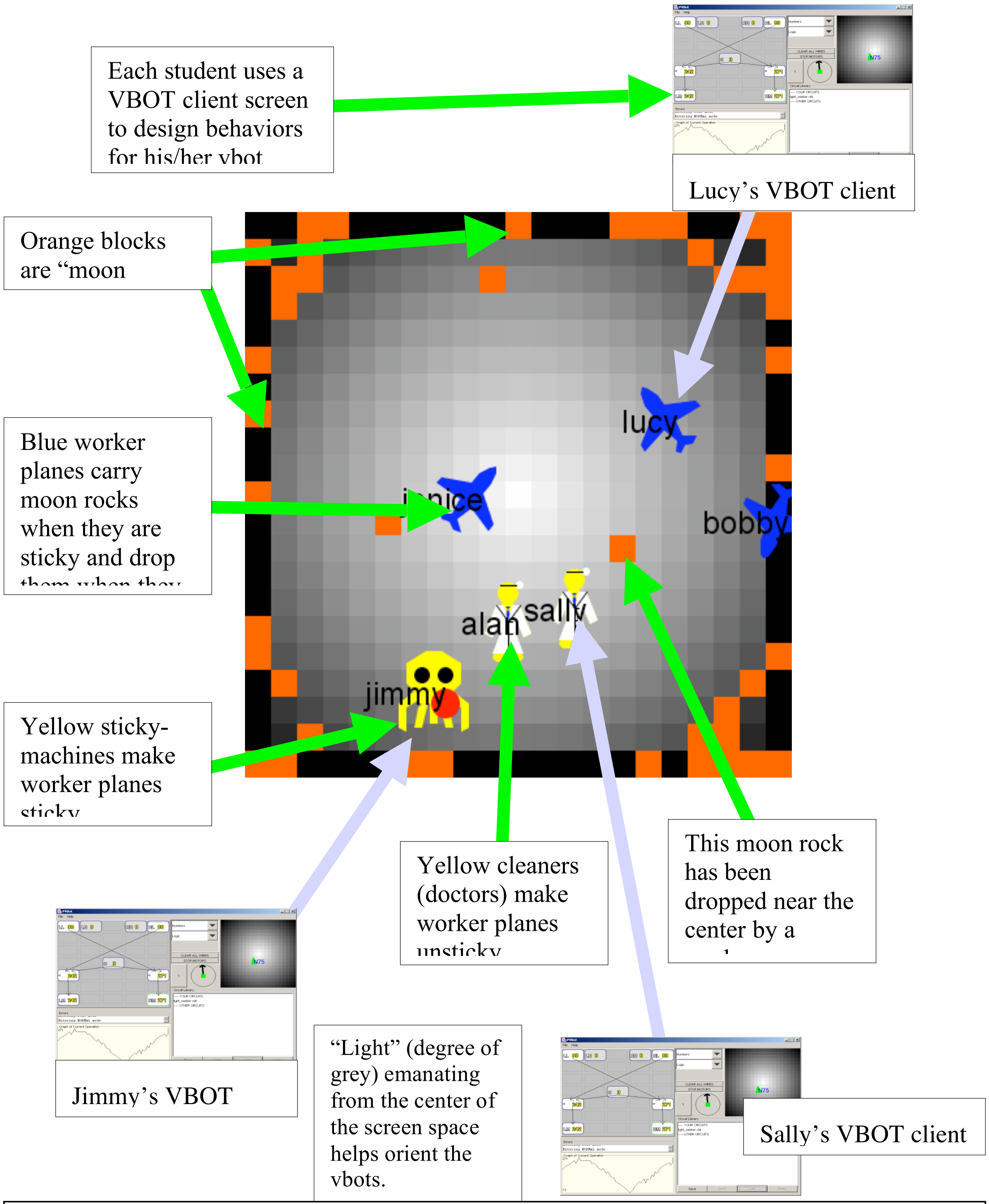

Each student uses a VBOT client screen to design behaviors for his/her vbot
Lucy’s VBOT client
Orange blocks are “moon
Blue worker planes carry moon rocks when they are sticky and drop
Yellow sticky-machines make worker planes sticky
lucy
janice
bobby
alan
sally
jimmy
Yellow cleaners (doctors) make worker planes unsticky
This moon rock has been dropped near the center by a
Jimmy’s VBOT
“Light” (degree of grey) emanating from the center of the screen space helps orient the vbots.
Sally’s VBOT client

Figure 2-3: Moon-Tag

2.6 Physical VBOT Interface

This dissertation compares the benefits and constraints of programming in physical and virtual environments. This section provides more detailed information about the VBOT interface with physical robots and the ways with which users interacted with these physical robots.

Physical Robots and the VBOT Software Interface

VBOT software works similarly with both virtual and physical robots. In general, physical robots work by running uploaded program circuits that are written in VBOT, whereas virtual robots run the current program circuit without explicit uploading. Thus, the major difference is physical; you must attach a communications device ("USB tower") to the computer in use, and you must select "download to robot" from the tools menu.

This physical difference results in a difference in use. That is, unlike the virtual-robots that automatically respond to programmatic changes, the physical robots must be told to download the new program stopping the action. Moreover, since every program change takes at least 30 seconds to download, activate, and observe the robot, it is time-consuming to implement incremental changes (as you might with virtual robots).

After users have built a circuit in VBOT, they can select the "Download to Robot" menu item (or associated key command). This compiles the circuit and downloads it to the robot nearest to the USB tower. The compilation stage translates the VBOT program into NQC code, and the NQC program is called externally. NQC is an executable (written for Mac, Linux, and Windows) that communicates with the physical vbots using a c-like programming language that includes most of the functionality of the particular brand of physical robots we used (NQC, 2005). Thus, the NQC programs are created as a translation of the students' work in VBOT in order to communicate with the physical robots; this was done in the background and students never worked with NQC directly.

An additional difference exists in the sensors. The sensors in physical robots are both less accurate and less precise than sensors in virtual robots. Sensors in a simulation can be perfect (i.e. a light sensor can give the exact light level readings), but, in the physical world, perfect information is impossible. This problem is compounded by the extra noise inherent in inexpensive sensors. VBOT compensates for some of the noise by normalizing the data upon program initialization, but this does not solve the problem because the sensors feed erratic data to the program. As we discuss in the literature review section, however, noise can be useful to teaching and learning about cybernetics, as many reactive systems can enter terminal loops without any randomness in the system. That said, too much noise can undermine even the best-written programs.

Physical VBOT Structure

A physical vbot is a Mindstorms RCX 1.0 robot (LEGO, 2002) built with a specific sensor and motor array. We provided students with these robots for certain implementations of VBOT. The physical vbot's sensor array consists of two motors, two light sensors, and one "bump" sensor. The bump sensor turn out to be significantly more valuable than a directional "bot" sensor for playing soccer, as the students would want to test for a bump of the ball as well as other robots. The Braitenberg (1984) metaphor of the simple wire-circuit car was used to construct the physical robots (see Chapter 3 for more information on Braitenberg). In Appendix 1, we illustrate the construction of a physical vbot and the final robot structure.

2.7 Related Environments

There exist no comprehensive studies of the relative affordances of comparative physical and virtual environments in constructionist learning. This is one need that this work is intended to address. However, several environments address complex systems and computational fluency goals. This section examines similar, constructionist virtual and physical learning environments and assesses design and design goal similarities so as to highlight the novel features of VBOT.

Virtual Robots as Learning Tools

We define “virtual robots” more broadly than a virtual instantiation of a physical robot. It is a virtual autonomous agent or any object that self-determines behavior. Humans are autonomous agents, as are paramecia, bacteria, and many robots. Autonomous action and autonomous decision-making are often called “intelligence.” The simplest example of "autonomous intelligence" is a thermostat; once it is set at a temperature, it determines when to turn the heater on and off. To this simple model we can add memory, logic, and other functionality. Further complicating the environment in which our thermostat with memory and logic might lead to a mimic of the simplest forms of life. A simple organism machine might “know” how to find food or reproduce when placed in a novel environment. Learning environments that use virtual autonomous agents, or virtual robots, are often designed to teach younger students how to program. The following section is both a review of these tools and a review of literature on virtual robot learning environments.

Logo Environments

A Logo environment is one that uses the Logo paradigm of programming, characterized as “low threshold, high ceiling” by Tisue & Wilensky (2008) and described in depth by Papert (1980). The Logo language was originally designed as a learning programming language, and it has engendered many variants, mostly designed as programming languages and modeling environments in which programming is easier for novice users and creates immediate, graphical effects. We describe NetLogo (Wilensky, 1999) and Flogo (Hancock, 2003) as two modern Logo variants relevant to the VBOT project. VBOT is based on HubNet and NetLogo, and, as such, it is a Logo derivative at some level. Logo derivatives have been mostly single-user learning environments, with the notable exception of the HubNet features in NetLogo.

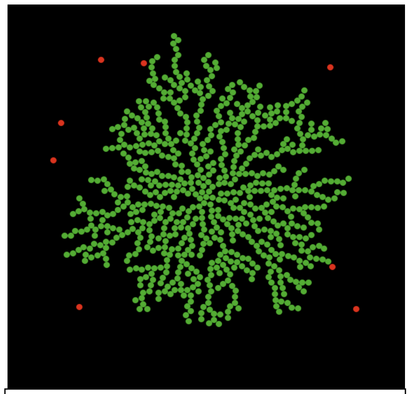


Figure 2-4: NetLogo

NetLogo (Wilensky, 1999) is a multi-agent Logo language created to enable modeling of complex phenomena with little formal training. NetLogo is a Logo language in which a user can program (and directly command) both a set of agents (“turtles”) and the environment in which the agents reside (a grid of “patches”). NetLogo is designed such that a user can add complexity to the environment using both the patches and the interrelationship between turtles and turtles and patches. In this environment (shown in Figure 2-4), thousands of turtles simultaneously interact in a virtual space. In contrast to the programming-by-example environments, NetLogo is a text-based programming language more similar to traditional programming languages. Included in the environment is a variety of example programs and working simulations (“sample models”) that users can modify or extend. This environment is designed to be a general-purpose useable programming environment rather than a targeted support for computer science learning, although it has been used as such with success (Abrahamson & Wilensky, 2004a, 2004b; Blikstein & Wilensky, 2004; Wilensky, 2001).

Flogo (Hancock, 2001, 2003) is a virtual/physical robot programming language in which a user designs a behavior for a virtual or physical robot using a circuit metaphor. Hancock (2001) describes "the characteristic of Flogo on which our hopes are most pinned is 'liveness' — meaning that instead of existing as an inert object that is first edited and later run, a Flogo program can be active, and its activity visible, even as you are building or modifying it." That is, users can modify their programs while they are running. In addition to this real time programming, Flogo uses a live dataflow "pulse" metaphor, meaning that a user can visually follow data as it is processed (see Figure 2-5). Hancock (2001, 2003) tested a Flogo physical robotics system with students ranging in level from secondary school to graduate school. His data suggest that this "temporal structure, Gibsonian learning, and tinkering friendly system[s]" helped students build constructively and learn computer science content.

In both structure and use, Flogo shares features with VBOT. Indeed, Hancock's work was invaluable in the design of VBOT, and VBOT was redesigned to emphasize both "liveness" and "tinkerability." VBOT extends the Flogo work by basing the activity design on theories of collaboration in learning, focusing on complex systems fluency and enabling using multiple agents (similar to NetLogo). Chapter 7 covers the relationship of VBOT to Flogo in more detail.

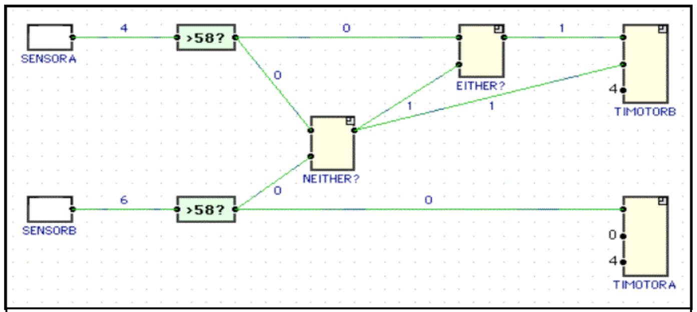


Figure 2-5: Flogo

Programming by Example

"Programming by example" refers to a type of computer programming in which a user writes agent code by providing examples of desired behavior, rather than the more traditional method of explicitly typing out instructions using a provided word-based language.

"Programming by example" refers to the way in which the users can command their agents on the screen. To "teach" a character to move up on the screen when it is next to a rock, the user places a rock one space to the right of the character, presses the "teach" button, and then moves the character up using the mouse. This "example rule" is automatically added to the character. The character repeats this action ("moving up") when, in the course of other actions, a rock is one space to the right of the character. Figure 2-6 shows one such simulation: the program code written by the user is at the bottom; the action window "worksheet" is in the top right; and a graph of the characters' positions is in the top left.

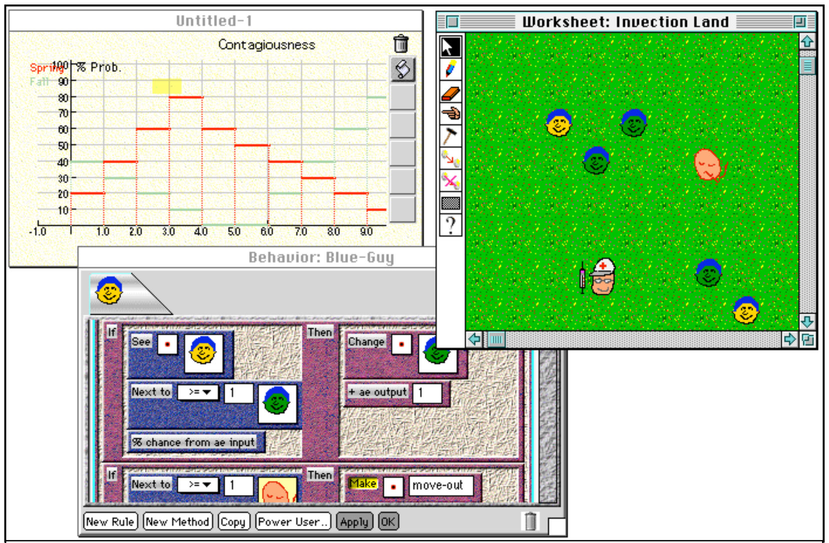


Figure 2-6: AgentSheets

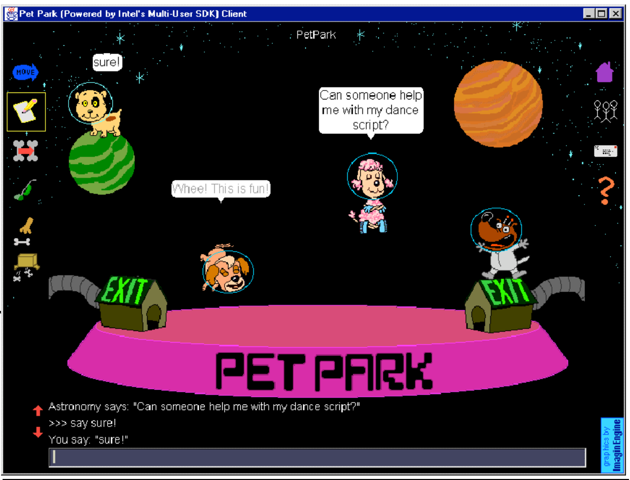
The given example was generated in AgentSheets. AgentSheets is a programming environment in a user builds simulations or games using the “programming by example” paradigm (Repenning, Ioannidou, & Zola, 2000). It is intended primarily for primary school students. The simulations and games are programmed in a virtual 2D environment in which each element is active as an independent agent. For instance, in a simulated outdoor environment, both a rock and a player character would be independent agents, that the students could program. All graphics are either drawn in the program by the user or provided with the program. The user is responsible for programming all of the behaviors for each interactive element, or agent, of the simulation or game.

AgentSheets is a virtual robot simulator because users can “teach” robot-agents how to move independently by building a set of example rules for robot-agents. As such, it is relatively easy to create simulations in which, say, a character avoids other robot-agents that have been programmed by the user. More complex scenarios can require hundreds of such rules. The programming environment supports these complex scenarios by providing several scenarios that students can modify or use for creating their own games or simulations.

A similar paradigm is used in ToonTalk. ToonTalk is an environment developed by Kahn (1996) as a visual programming language also built on the constructionist “programming by example” paradigm. However, unlike AgentSheets, it takes place in a 3D world in which every virtual object is a program abstraction. In ToonTalk, users program “robots”. Each “robot” can hold a variety of “memories” which are added to the robot abstraction by example. A memory is equivalent to a line of program code. For example, users might teach the robot to “double numbers" by showing the robot a "magic doubling wand" and having it "use the wand" on an abstract box until the wand is used 10 times.

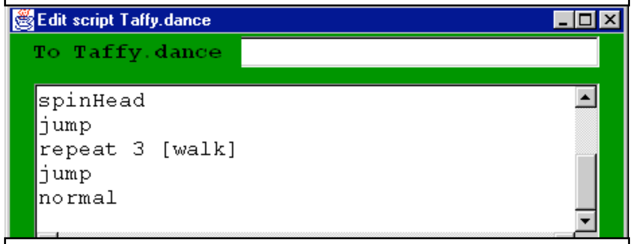
Social Agent Environments

Social agent environments are programming and learning environments in which the primary medium is inherently collaborative or competitive. Programs written in social agent environments are designed primarily to interact with multiple users. Second Life (2006) is a commercial example of a social agent environment; Second Life is a multiplayer online game in which hundreds of thousands of users interact with object-programs in a persistent online space. MOOSE Crossing (Bruckman, 1997) is presented as an example of a social agent environment particularly relevant to the VBOT project. Designed as an explicitly constructionist environment, MOOSE Crossing emphasizes social collaboration over the more individual programming paradigms of the Logo or Programming by Example environments described above.


Pet Park (Powered by Intel's Multi-User SDK) Client
PetPark
sure!
Can someone help me with my dance script?
Whee! This is fun!
EXIT
EXIT
PET PARK

```
Astronomy says: "Can someone help me with my dance script?"
>>> say sure!
You say: "sure!"
```

Figure 2-7: Pet Park



```
To Taffy.dance

spinHead
jump
repeat 3 [walk]
jump
normal
```

Figure 2-8: Programming Pet Park

MOOSE Crossing (Bruckman, 1997) is a system with a MUD ("Multi-User Dungeon") architecture in which each user-designed character "lives" in a textual programmable discussion space. Users engage in a textual "chat room" in which actions that change the environment of the chat room can be made (see Figure 2-7). For example, a user could set an alarm for 10:30 PM; any characters in the chat room at 10:30 PM will get the textual alarm "BRRNG! BRRNG!" By programming the environment in which people can textually interact, the programming is inherently social. Bruckman (1997) states that "the central claim [here] is that community and construction activities are mutually reinforcing." Users can program characters to automatically greet users upon entrance or to discuss short topics with other characters. Examples in Bruckman's (1997) work show that users can successfully create a vibrant community organized around the programming features of the space.

The Pet Park system detailed by DeBonte (1998) is a graphical form of MOOSE Crossing, in which the agents are persistent in the world (see Figure 2-7: 2-7 and Figure 2-8). DeBonte's work suggests that the persistence and graphics create a functionally different space. The design of VBOT stems in part from the constructionist social collaboration work described above. VBOT is real-time, social, and graphical, much like PetPark, although the design and learning goals are different.

Physical Robots as Learning Tools

Physical robots have been used to teach computer science for over 20 years (Papert, 1980). More recently, the LEGO Mindstorms robotics system has been released to wide acceptance and a new, redesigned version ("NXT") was released in late 2006 by LEGO. It is used in hundreds of classrooms to teach computer science, math, and core science. In this section, we describe the prototype system for LEGO Mindstorms, the Programmable Brick, and the most common ways in which physical robotics have been used for teaching and learning. As described above, Physical VBOT uses LEGO Mindstorms robots. This section describes other uses of this system and its relatives.

The Programmable Brick

The Programmable Brick (Resnick, Martin, Sargent, & Silverman, 1996) is a physical robotics system designed as a constructionist learning environment. Users attach simple sensors and motors to a pre-built processor ("the brick") and board (see Figur). The motors then move according to a user-created program that uses the input from the sensors. The LEGO Mindstorms system was designed using the programmable brick as a prototype, though there was ongoing design of the brick into the late 1990's, after the LEGO Mindstorms kit had already been released commercially. The LEGO Mindstorms system not only provides a research base on which we build but is also the physical system with which our students work. Several groups have evaluated Mindstorms (or a similar robotics system) for out-of-the-box public classroom viability with positive results, although none of them are immediately relevant to this study as they are more traditionally teacher-centered (see, for example, Lawhead, Duncan, Bland, Goldweber, Schep, Barnes, & Hollingsworth, 2003; Klassner, 2002; Tribelhorn & Dodds, 2006).

Martin (1996a, 1996b) provides a constructionist study of students learning to control LEGO Mindstorms robots. The students in his study range in level from primary school students to MIT undergraduates. The study suggests that the embodied nature of the programmable brick encourages elementary students to

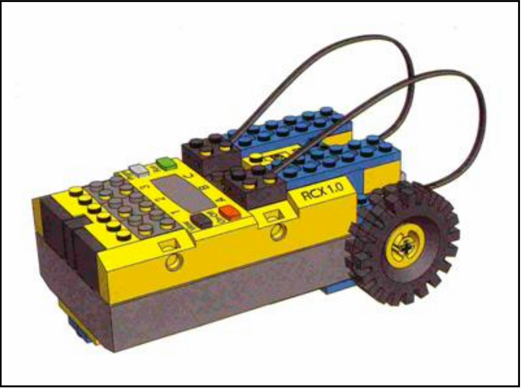


Figure 2-9: Lego Mindstorms Robot

make a wide variety of aesthetic and functional decisions. In one scenario, a group of young students build a robot with a brush attached and program the brush-robot to “dance” in concentric circles. The creativity was not limited to primary school art class. The aesthetic, whimsical, and creative aspect of the robots appeared to motivate the undergraduate students working with the system. These students reported that the creative aspects of the tasks retained their interest long enough to become fluent with the technology. Although Martin felt that the students that acquired fluency with the technology primarily learned only the technology itself (as opposed to content that may be more obviously aligned with traditional curricular goals), he suggests that this was a limitation of the crudeness of the early technology; the mechanical engineering skills necessary to become fluent with the programmable brick were a significant barrier to learning outside content using the bricks as a tool.

Martin (1996a) suggests that only by using unreliable physical sensor machines can students independently distinguish "policies" and "plans". A policy is a set of behaviors that run as triggered by certain criteria, rather than a list of ordered instructions. For instance, a policy behavior might be written: “when the temperature is under 50 degrees, move toward the closest heat source.” A policy would consist of many such behaviors, potentially triggering in parallel. A plan, on the other hand, is a set of deterministic instructions. For example, a plan called GO_TO_STORE might be written as such: "forward 5 m, turn left 90 degrees, forward 10 m, right 90 degrees, forward 10 m, left 30 degrees, forward 2 m, right 10 degrees, forward 40 m." Plans rarely achieve optimality in physical robotics due to the unreliability of sensors and environmental noise; policies are often easier to understand and work more reliably in a variety of environments (Arkin, 1998; Brooks, 1999; Jones, 2004). Martin (1996a, 1996b) found that the programmable brick learning environment can motivate students to develop a more flexible programming approach, moving between policies and plans depending on the environment in which the robots operate.

Physical Robotics in the Classroom

Undergraduate engineering education

Undergraduate engineering and computer science education have been the most common uses of robotics. I was first introduced to Papert and LEGO/Logo robotics during a robotics course with Leslie Kaelbling at Brown University as an undergraduate. My experience was not uncommon. Martin (1996a) describes some of the ways that he has analyzed the use of robotics for teaching undergraduates, and a cursory search of major universities shows that many of the larger computer science departments have a class in which LEGO Mindstorms robots are used.

High school and college robotics competitions

Robotics competitions are often used to motivate students in engineering or computer sciences disciplines. These happen at several levels, ranging from middle school to undergraduate. For example, Northwestern has such a competition, as does Brown and MIT. There are several such competitions in the Chicago-land area. Two of the more well-known competitions are RoboCup (1997) and FIRST (2006). Some of the robots that are built for these competitions are impressive, but most are simply sequences of commands designed to traverse a pre-set course. In other words, the robots might have a set of commands such as: turn left, forward 80 tire rotations, turn right, forward 20 tire rotations, and then stop. Our work on VBOT works to extend this work by enabling students to provide rules for their robots to follow rather than a sequence of steps; VBOT uses "policies" where these robots use "plans."

Primary school robotics

Granott has designed robots with an explicit focus on teaching students about logical thinking and basic programming. Granott's "weird creatures" (1993) follow behavioral patterns hidden from the user. A user then tries to determine the flow of programming for a given robot. Building on this, Mioduser (1996) and Levy, Mioduser, & Talis (2001) have done work with primary school children in which the students build and analyze "weird creatures" or "weird structures." The robotics work in primary schools is relatively new, but groups like RoboCup Jr. (2006) are using some of these techniques to appeal to younger students.

Constructionist projects in physical robotics

Resnick (1998) describes various physical robotics technologies that the media lab has used in primary, middle, and secondary schools. One interesting project implements Hancock's (2001) architecture for "communicating Crickets" – robots that work together to accomplish small tasks. Hancock's work is further described below in the "Logo Environments" section. Resnick, Berg, and Eisenberg (2000) found that " children, by teaching their creatures to communicate with one another, can learn some general principles about communication." Resnick et al. (2000) further describe many types of physical robotics projects possible with the Programmable Brick and Crickets kits. The students designed robotic projects as complex as an automatic bird feeder, a mini-golf machine, and a robotic flower that opens in the presence of light.

Chapter 3: Related Literature

3.1 Defining Computational and Complex Systems Fluencies

Print literacy and print fluency have been mostly “invisible” throughout recorded history, in that scholars did not study the differences between speaking a language and understanding its rhetoric (Manguel, 1997). For Plato and the classic philosophers, reading was aloud, writing existed only to record history as fact, and languages other than Greek or Latin were the exotic forms of barbarians and usually not committed to paper. Although reading and writing were understood to be different than speaking, the difference was generally understood to be at the expense of the written word. For instance, Aristotle’s Poetics (as translated by Heath, 1996) was posited as a guide to the forms of reason and argument as spoken, not as read. Fluency itself was not a scholarly subject until the rise of philology during the Enlightenment (Cassirer, 1979), although writing about the act of writing flourished during the Renaissance. Petrarch (trans. by Cassirer, 1948) proposed that writing existed as a way to ties together existing memories and understandings – a concept notably similar to Piaget’s (1972) description of constructivist learning in children.

As a philosophical descendant of Piaget, DiSessa (2000) defines literacy as "a socially widespread patterned deployment of skills and capabilities in a context of material support (that is, an exercise of material intelligence) to achieve valued intellectual ends" (p. 19). That is, literacy is the ability to use a medium as a material, cognitive, and social tool. Defining literacy in terms of these three facets highlights the complexity of literacy and the degree to which true literacy can change the way individuals think and learn.

Print literacy, for example, requires material understanding of the mechanics of letters and words, cognitive ability of how to consume and produce effective prose, and social understanding of what written words are appropriate in a given situation and how best to communicate using written language. Only by being able to make sense of each of these aspects can one be said to be print literate. Moreover, being print literate makes available new ways of making sense of and accessing the world of information. Just as print literacy opens a lens through which we can look at the social world, so can other literacies change our perspectives. Specifically, computational literacy opens a lens to the entire world of technology and engineering (DiSessa, 2000), and complex systems literacy gives us new perspectives on a variety of scientific phenomena (Jacobson & Wilensky, 2006).

In the following sections I will clarify each of the constructs in DiSessa's framework (material, cognitive and social literacies) by applying them to the VBOT architecture and corresponding activities. I will then use these constructs to define complex systems literacy and the benefits of developing this perspective. Henceforth, we will use the term "fluency" to mean a type of literacy not rooted in print literacy.

When used informally, fluency, as opposed to literacy, is rooted more in practical concerns than in socio-political ones. Colloquially, one might say, “Alice is a very literate person.” This would imply that Alice understands and uses context from history, literature, and the arts; Alice can cross-apply this understanding. As a contrast, if one were to say, “Sam is a very fluent person,” it implies only that Sam can speak and read the language well enough to have a conversation. It is for the reason that the word “fluency” is used, as the thesis addresses issues more related to practical understanding than deep contextual understanding.

Computational Fluency

The term “computational literacy” has often been used to describe the mastery of a few standard computer applications. For example, as a secondary school teacher, I taught a class called "computer literacy," which was designed to teach mastery of Microsoft Word, Excel, and PowerPoint. Papert (1980) and DiSessa (2000) change this definition to argue that “computational literacy” implies both the ability to use computer software and the ability to create and manipulate computer software (or hardware) to communicate and disseminate ideas. This definition of literacy parallels the generally understood meaning of print literacy; a print literate individual can express oneself in writing. The expressive and authoring aspects of computational fluency are largely ignored in the pre-collegiate curriculum. Moreover, we argue that such computational fluency can greatly benefit citizens of our era, because computers shape so much of our interaction and communication.

DiSessa's (2000) definition of literacy can unpack this more rich understanding of "computational fluency." As defined by Papert and DiSessa, computational literacy introduces the possibility that computational literacy does more than enable work with end-user programmers. Instead it changes how individuals think and learn. In the following we define the social, cognitive and material aspects of computational fluency by identifying how our project addresses each aspect.

Material computational fluency is the ability to use the material tools. This is the aspect that most closely aligns with more common definitions of computer fluency. It encompasses both the ability to use computer programs and knowledge of programming languages. For instance, any use of VBOT requires some level of *material* computational fluency: students must learn the VBOT circuit language to compete or collaborate with one another through the system.

Cognitive computational fluency describes the student's ability to think with the computer-as-tool. Developing cognitive fluency requires that students adapt their thinking processes to align with their tools (in this case, VBOT). This is, functionally, being able to think like a computer programmer or a computer artist. Cognitive computational fluency allows the student to use the computer as a tool to solve even those problems that do not require computers. Moreover, being able to conceptualize what elements of given problems with which the computer can aid is important to cognitive fluency. Students build computer programs and hopefully become familiar with the computer as a protean "tool to think with" (Papert, 1980) rather than as an inert tool. For example much as student may use a pad of paper to solve a 10-digit division problem, so they can use a computer to help work through a 1000-digit division problem.

To communicate in and around a computational environment, one must share certain technical and socials means and goals; this is *social* computational fluency. As the photography-literate discuss "light levels," so the computationally literate can discuss "efficiency" or "cleanliness." Inherent to our project is communication between students and the trading of information and resources. Students that can effectively communicate and trade will have an advantage, and, at some point, every student will discuss the sharing and communication that they have with the whole group.

Complex Systems Fluency

Complex systems fluency refers to an individual's ability to negotiate the relationships between "agents", "aggregates", and "levels thinking." This terminology as used in this paper is derived from Wilensky & Resnick (1999), who use the example of a traffic jam:

> "Two high-school students were writing a computer program to simulate the flow of traffic on a highway. They began by writing some simple rules for each car: Each car would accelerate if it didn't see any other cars ahead of it, and it would slow down if it

> saw another car close ahead. They started the program running, and observed the patterns of traffic flow. On the screen, a traffic jam formed. They continued to watch and–much to their surprise–the jam started drifting backward along the highway. 'What's going on?' said one of the students. 'The cars are going forward, how can the jam be moving backward?'"

In this example, the agents are the cars in the traffic jam. A traffic jam consists of cars. The traffic jam is a result that emerges from the aggregation of cars. Understanding the relationship between the individual cars and the emergence of the aggregation of the cars is called *levels thinking*. Levels thinking is not only the ability to think at both the level of the cars and the level of the traffic jam, but also the ability to concretize the relationship between the two levels.

Complex systems research has become increasingly important for understanding scientific phenomena (Holland, 1995; Wilensky & Resnick, 1999; Wolfram, 2002). Scientists use complex systems methods to model phenomena in domains varying from physics (Bar-Yam, 1997) to social interactions (Watts, 2003). Some have argued that complex systems theory is a new kind of science, one posed to usurp the mantle of scientific explanation from traditional equation-based science (Wolfram, 2002). Others have shown that modeling with complex systems is more comprehensible to high school students than traditional equation-based science (Centola, McKenzie, & Wilensky, 2000; Ioannidou, Rader, Repenning, Lewis, & Cherry, 2003; Jacobson

& Wilensky, 2006; Wilensky, 1997a, 2001; Wilensky & Reisman, 1998, 2006).

We use DiSessa's (2000) framework for computational literacy to describe complex systems fluency. This framework breaks literacy into material, cognitive, and social fluencies. In this section, we describe how these elements are addressed.

When working with complex systems, scientists use a diverse set of strategies and overlapping tools. Therefore, *material* complex systems fluency concerns a range of tools. Within those tools, there is an emphasis on computational tools for analyzing data and modeling phenomena. Thus, material fluency in complex systems requires a level of both computational fluency and scientific fluency. Students need to be able to produce theories and hypotheses, evaluate them, and report on results. Furthermore, the students need to be able to build models, material or computational, which produce interpretable results.

Cognitive complex systems fluency is the ability to think with and from complex systems theories and models. Can students use multi-agent systems to consider a phenomenon? Can students approach a problem from multiple levels, such as the agent-based and the emergent or aggregate? As students become more familiar with the relationships between levels of complex phenomena, they begin to use emergent and complex systems thinking as a tool with which to think about everyday physical phenomena. Wilensky & Resnick (1999) call this "levels thinking."

As complex systems fluency is a form of scientific fluency, *social* fluency becomes a vital element of complex systems fluency. The scientific community revolves around the communication of results and hypotheses (Latour, 1987). Students becoming complex system literate should learn the terminology of the complex systems scientist and an ability to communicate about the various levels of a complex phenomenon. Students should be able to describe phenomena from the agent-based, aggregate, and emergent perspectives. Scientific findings are meaningless if not communicated and immersing oneself in a "new kind of science" (as per Wolfram, 2002) requires a new kind of communication.

3.2 Learning Theories for Supporting Complex Systems and Computational Fluency

Methods and theories of complex systems are largely absent from school curricula. Informal conversations with teachers and principals suggest that they believe that there are two possible reasons for this absence: 1) these concepts are too hard for students to grasp, and 2) the cost of entry (training, resources) to introduce these concepts into school curricula must be prohibitively high. Findings in VBOT pilot studies have suggested that both of those assumptions are problematic (Berland, 2006; Berland & Wilensky, 2004, 2005).

Research shows us that teaching students to use computer programming to do complex systems science can make computational fluency and complex systems fluency mutually reinforcing (Wilensky, 2001; Wilensky & Resnick, 1999). Our pilot studies contribute to this literature with preliminary findings that there are correlations between complex systems and computational fluency in students using VBOT (Berland & Wilensky, 2004, 2005). Moreover, we see robots as a natural fit in complex systems research. Complex systems events involve many similar elements doing simple tasks that, together, create some emergent phenomenon (Holland, 1995; Wilensky, 2000). Similarly, it is common for robotics research to use several simple robots that collaborate in the creation of a phenomenon (see Parker, Schneider, and Schultz, 2005, for a variety of examples). Despite these parallels, and despite that robots are used to teach computer and mathematics fluency (Resnick & Ocko, 1991), robotics, as a field, has only recently begun to be addressed in complex systems research (Pollack, Lipson, Funes, & Hornby, 2001). Furthermore, there is little research using robotics to address complex systems fluency.

Given the small research base of literature focused on fostering complex systems and computational fluencies, we begin by identifying ways of learning that influence our work: constructionism and play. We then move on to focus on the potential of programming and robots for developing these fluencies.

Constructionism

Constructionism postulates that one learns more about an object or a concept by participating in the process of building the object or concept. “Learning by making” is a major component in robotics education projects (Hancock, 2003; Portsmore, 1999; Resnick & Ocko, 1991). As Wilensky (2000) notes, tools that utilize the individual components needed to complete the aggregate can both help the student understand the final concept but also allow the investigator to understand the process of learning. Wilensky differentiates “black box” projects in which subjects begin in the middle of the process of creation with “glass box” projects in which subjects can see the process of creation from start to finish.

The Logo computer language is an example of a "glass box" constructionist educational programming environment. Users program a "turtle" (a virtual agent) with a simple, but full-featured programming language, streamlined for beginners, in which all relevant program code is visible to the user-programmer. The Logo computer language is discussed in more detail by Papert (1980). Often Logo is used as a mathematics programming environment or a drawing language. However, the Logo programming environment can have much more complex applications; constructionism and Logo have been the focus of some encouraging studies of computational fluency (such as those detailed in LCSI, 1999).

Play and Collaboration

Commensurate with constructionism is the notion of play. Traditional school curricula rarely involve significant "play" time, and efforts to raise standards for teaching and learning and schools have neglected to target sources of student motivation. A considerable body of research, however, has shown that play can be a powerful academic motivator (Dewey, 1913; Harel & Papert, 1990; Kafai, 1995; Papert, 1980; Vygotsky, 1978). Additionally, there is convincing evidence from both the social and cognitive streams of learning research indicating that learning and transfer are more easily achieved when the students are motivated to work through activities (e.g., Ames & Archer, 1988; Dweck & Elliot, 1983; Pintrich & Schunk, 1996). Schank & Cleary (1994) also show that intrinsic motivation can lead to more personally relevant, stable knowledge acquisition for many students.

There have been few studies using real-time games in constructionist learning environments. Recent research has shown games to be effective teaching tools in several domains (Gee, 2004; Squire, 2004; Steinkuehler, 2004). Gee (2003) discusses how and why games can be effective in teaching fluencies. Constructionist research has long used games to teach mathematics (Kafai, 1995), and recent games research is making significant headway into the processes and motivational aspects of how and why games are important for learning and teaching. Pilot studies have shown that students regard VBOT as a game and interact with each other as if in a game scenario (Berland & Wilensky, 2005). VBOT is designed, in part, so as to integrate findings from the field of game research; we will cover this relationship in more detail in Chapter 7.

Studies have also shown that enabling social-help interactions leads to improved cognitive and social functioning (Gutierrez, Rymes, and Larson, 1995; Vygotsky, 1978). Vygotsky (1978) describes the Zone of Proximal Development (ZPD), which is the level at which children can function in a social-help setting as opposed to an individualized one. Gutierrez et al. (1995) show that often the most productive and thoughtful interactions occur in informal spaces within the classroom, outside of the direct view and control of the teacher. This literature combines to highlight the importance of individuals learning through playful collaboration.

Since the advent of the personal computer as a learning tool, collaborative programming has been evaluated several times as learning method (e.g. Papert, 1980; Tiffin & Rajasingham, 1995). Recently, new programming methods, such as "extreme programming," have been used in collaborative programming research (Beck, 1999). Thus we view programming as a fruitful environment for fostering the collaborative learning supported in this literature.

Even while constructionism, play and collaboration offer theoretical guidelines and hope for fostering complex systems and computational fluencies, there have been few comprehensive collaborative constructionist studies of complex systems fluencies. This study attempts to address this need by creating just such a learning environment and then examining the learning that occurs therein.

VBOT was designed to use constructionism and play through the metaphor of virtual and physical robotics. Towards that end, Braitenberg (1984) provides a virtual robotics starting point for projects in art, philosophy, electrical engineering, cognitive science, and several other disciplines, as seen through virtual, physical, and theoretical robotics. The book begins with descriptions of how to build (as circuits, thought experiments, programs) a simple set of autonomous robots that either “love” or “hate” light. To love light is to tend to travel towards it. For instance, a “love” robot might be a small, two-wheeled artifact that orbits a lamp. A “love” circuit could simply connect a light sensor to a motor attached to a wheel. As the light sensor sees more light, the motor speeds up, spinning the wheel faster, and sends the robot towards the lamp. For more detail on the technical detail of these circuits, consult Chapter 2.

By progressing through simple circuits, such as “love” circuits, which involve only light sensors, motors, and wheels, Braitenberg prompts a difficult set of questions previously raised by biology, computational, and philosophy. If simple circuits can create complex behaviors, what is complexity? How does the complexity of the behavior relative to the complexity of the circuit contribute to an understanding of how the brain works? Braitenberg’s work is remarkable in the ease with which beginning students can reach understandings of his simple circuits, and of the questions raised by the behaviors. Braitenberg shows that these circuits are valuable, motivating, and interesting tools for raising and also answering a range of questions for any level of learner.

It is from this work that much of the research in educational robotics has stemmed.

3.3 The Virtual and the Physical in Robotics

To describe the history of the rift between the virtual and the physical in the literature would be presumptuous; works as monumental as Plato's *Parmenides* have mapped the boundaries of the question. More recently, post-structuralism has attempted to dissolve the boundary between dichotomies such as virtual and physical (see, for example, Haraway, 1991). This project's goals are more prosaic. To that end, we will address only work dealing specifically with virtual and physical robotics. Indeed, there has been significant debate in the computer science literature about the relationship of virtual to physical robots (see, as a reference, Jakobi, Husbands, & Harvey, 1995; Wainer, Feil-Seifer, Shell, & Mataric, 2007). The vast majority of this literature concerns the relevance of virtual robotics techniques to the use of physical robotics for real-world tasks. The VBOT project has little in common with the world of real world physical robotics. That is, the real world physical robots, such as the iRobot models used for research or the robot arms used to build cars, have a different set of functionalities than the LEGO Mindstorms robots used in the VBOT project.

Noise is the name given to perceptual and motive inaccuracy in robotics, and, in terms of physical robotics, it has played a significant role. Without a large expenditure, it is fairly difficult to get sensors for simple robots that read light, elevation, temperature, or any number of possible information sources with reliable precision and accuracy (Martin, 1996b). Furthermore, the world that humans inhabit is "noisy" in that reliable information about objects or people is often obscured or only partially available. To that end, modern robotics often uses sensors and processes designed to minimize noise and use partial information (see, for example, Brooks, 2002).

Thus, one significant difference between physical and virtual robots is that it is mathematically and conceptually difficult to simulate "real-world" noise in a virtual setting. It is not simply a matter of preparing a virtual world with partially obscured information, but instead preparing a virtual world with dynamic partial information that is constantly changing in a semi-random (but rarely fully random) contingent noise (i.e. the world is fully interrelated). This difference will be addressed again in Chapter 6 in which students' understanding of "noise" and physicality is analyzed.

3.4 Open Questions

Although there have been many studies in computational and complex systems fluency, more work is needed. In this section, we will describe the most salient open questions in the literature.

There is no comprehensive study of the relative affordances of comparative physical and virtual environments in constructionist learning. As discussed in Chapter 2, virtual and physical environments have both been used to success in constructionist research and otherwise. Although there have been several studies of the relative affordances of virtual and physical environments (see Sharlin, Watson, Kitamura, Kishino, & Itoh, 2004 for a review of research on tangible interfaces), that work has not been explicitly both constructionist and empirical. This study, in comparing the virtual and physical environments empirically, has the advantage of comparing them in the same situations and with the same subject pool.

As the field of complex systems is young, there have been few comprehensive collaborative constructionist studies of complex systems fluency. Even though there is significant work using constructionism to teach complex systems fluency (such as Wilensky & Resnick, 1999), the field's youth leaves many unaddressed topics. For instance, there has been relatively little research concerning the use of physical systems in teaching complex systems fluency. While this gap is being addressed by researchers such as Abrahamson, Blikstein, Lamberty, & Wilensky (2005), there remains much work to be done in this field.

While some research has shown a relationship between computer and complex systems fluency, there is little evidence that these fluencies are mutually beneficial. Research shows us that by teaching students to use computer programming to do complex systems science, computational fluency and complex systems fluency can be mutually reinforcing (Wilensky, 2001; Wilensky &

Resnick, 1999).

There have been no comprehensive studies on the use of physical robots in teaching complex systems fluency. Robots are a natural fit in complex systems research. Complex systems events involve many similar elements doing simple tasks that, together, create some emergent phenomenon (Holland, 1995; Wilensky, 2000). Similarly, it is common for robotics research to use several simple robots that collaborate in the creation of a phenomenon (see Parker, Schneider, and Schultz, 2005 for a variety of examples). Despite these parallels, and despite that robotics are commonly used to teach computer and mathematics fluency (Resnick & Ocko, 1991), robotics, as a field, has only recently begun to be addressed in complex systems research (Pollack, Lipson, Funes, & Hornby, 2001). Furthermore, there is little research using robotics to address complex systems fluency.

There have been few studies using real-time games in constructionist learning environments. Recent research has shown games to be effective teaching tools in several domains (Gee, 2004; Squire, 2004; Steinkuehler, 2004). Gee (2003) discusses how and why games can be effective in teaching fluencies. Constructionist research has long used games to teach mathematics (Kafai, 1995), and recent games research is making significant headway into the processes and motivational aspects of how and why games are important for learning and teaching. Though VBOT is not explicit designed as a game, it can exploit many of the same motivational tools (Berland & Wilensky, 2005). The present study is therefore able to integrate findings from the field of game research.

There has been little research in understanding the spread and flow of information in collaborative constructionist environments. Research on social networks has recently made significant progress about transmission of information around small groups (Brown & Duguid, 2002; Watts, 2003). As of yet, few studies have applied it to constructionist learning environments, even though constructionist learning often involves collaboration and the sharing and distribution of information. The present study aims to demonstrate the value of collaborative learning in small social learning networks research and constructionist research.

VBOT provides an untested model for collaborative programming environments. Since the advent of the personal PC as a learning tool, collaborative programming has been evaluated several times as learning method (Papert, 1980; Tiffin & Rajasingham, 1995). Recently, new programming methods, such as “extreme programming,” have been used in collaborative programming research (Beck, 1999). The present study provides a new framework for simple collaborative programming in complex environments.

The above literature review has identified a number of questions that the current literature does not yet address. This work is structured to fit into this literature by building on the existing designs in which users program virtual and physical robots, extend these designs to align with the theories of constructionism, and playful collaboration. We do this with the expectation of learning how collaborative constructionism can support students as they develop both computational and complex systems fluencies. In Chapter 2, we focus on the design of VBOT and how this literature influenced the technical design. Chapter 7 focuses on the design of the VBOT activities and how those related to collaborative constructionism. We then move to study the interactions and learning that occurs in this supportive environment.

Chapter 4: Context, Methodology, and Data Collection

This work is the culmination of a series of three tests of the VBOT architecture, activities and theory. This chapter discusses details of the methodology and context for each of these trials. It is organized into descriptions of the methods and analyses of the pilots followed by a description of methods and data collection for a controlled study performed in 4 middle-school classrooms.

Although each of these studies represents a change in focus, reflecting lessons learned from the previous work, they follow a similar design research methodology. Design research (as described in Collins, 1992; Edelson, 2002) can take myriad forms, from relatively uninflected social anthropology to UI studies that are conceptually closer to traditional lab studies (such as Schank, Fano, Bell, & Jona, 1993). This work tends more toward the classical UI study than a minimally invasive social anthropology study. However, this study differs from UI studies in that we privilege teacher practice and school context. In VBOT studies, we collaborate with teachers in their common environments. In our research studies, we attempted to ensure that teachers could teach and discuss with the technology, rather than teaching about the technology.

The data collected for each of the pilot studies were focused on the affordances of the tools, the effectiveness of the lesson in teaching the target concepts, and the relative affordances of the various lessons and tools. To that end, the data is both quantitative and qualitative. The qualitative data collection is designed to facilitate analysis of students' computational fluency, complex systems fluency, and understanding of the target content. The qualitative data was collected primarily through videotape data of the activities and interviews with the students. Interviews were conducted immediately before the activity began, during the activity (in situ), and immediately after the activity. The pre-activity and post-activity interviews were entirely conducted by the author. Researchers videotaped the activity on hand specifically for data collection and technical support. The quantitative survey and activity log data was collected in the second and third of the three studies. The quantitative data collection was designed to facilitate analysis of student practice, program design, and study design. In the following, we provide more details about each of the trials, addressing how they differ in context and research methodology.

4.1 P1 – Initial Pilot

The first VBOT pilot (P1) was designed to test the basic VBOT architecture and GUI. To that end, it was conducted at Northwestern University, outside the school environment and was entirely facilitated by a team of researchers, led by me.

The sample consisted of 3 girls and 6 boys (n = 9). All 6 boys and 2 of the girls were either 13 or 14 years old; one girl was 11 years old. Our sample was gathered in an informal manner; one of the facilitators knew the parents of the various children socially. The self-selected group of students was from two schools, one public middle school and one religiously-affiliated middle school. The students were not reimbursed.

The first researcher facilitated the activities with help from 3 collaborating researchers. The other researchers were instructed to help participants use the technology, remedy technical problems, and collect data. None of the facilitators helped students with conceptual difficulties or content issues. In fact, we were not often asked to help students solve problems. More often, we mediated social issues and answered technology questions.

Data collection

All activities were videotaped using two handheld cameras. One camera was focused primarily on group interaction by videotaping the classroom as a whole, while the other cameras focused on specific student strategies and interactions. We also conducted pre- and post-interviews with the students.

Activity

The classroom was arranged in a “V-shape” with students pointed at the center-front of the room such that they could see one another, the screen in the middle of the classroom, and their individual computers. We didn't collect social network data in this study, but the table was always active with student-talk, and all students could see each other over their screens. Students used 13" Macintosh iBook G3 computers. Around minute 40, the experiment was interrupted by a pizza lunch, our only participant payment. This allowed for significant social time, and the participants ate and talked for about 15-20 minutes. The pilot ran around 90 minutes (not including lunch). There were 4 activities provided by the facilitator, and the students were allowed to vote twice between two sets of activities. All activities were done with a preliminary version of virtual VBOT described in more detail in Berland & Wilensky (2004).

All activities were restricted to variations on the flocking activity described in Chapter 2. While these activities were simple in both goal and process, they generated a range of discussion about the types of circuits to build. The students modified their circuits while they were discussing them in order to generate better results for the group.

Due to our relative success with leading group interaction and discussion using VBOT, this short study suggested that a move into formal classrooms would be both practical and fruitful.

4.2 P2 – Second Pilot

P1 was our initial trial. While P1 was provisionally successful in teaching the students about complex systems and computational fluencies (see Berland & Wilensky, 2004), it was most helpful in highlighting the content and form of what we had missed. In that short session, we found major problems both in methodology and design. Our software was too limiting, the assessments weren't adequately effective for evaluating fluencies, and the activities did not optimally exploit the content. Thus, the transition from P1 to P2 involved substantial development over the following year. We redesigned the software and the assessments. Our design philosophy and process is detailed in Chapter 7. Given these changes, P2 was designed to test the new technologies and redesigned VBOT systems towards teaching complex systems and computational fluencies. The research methodology changed considerably in the following ways: 1) we decided to focus more on the teacher-facilitator, 2) administer more comprehensive questionnaires, and, most importantly, 3) log all contact with the VBOT system. It was difficult to practically evaluate the actions of the students when we were limited to analyzing their actions only on a shared space.

Description

P2 ran for three days (80 minutes / day) in a single 6th grade science classroom. The classroom in which we worked was typically teacher-centered; the teacher led the class in both lecture and discussion. This contrasted with the relatively open discussion and interaction that characterizes the classroom when using VBOT. The VBOT activity was situated in class time usually dedicated to astronomy. As such, the VBOT activities were designed around an astronomy metaphor, the “Moon-Tag” activity described in Chapter 2 was contextualized in this astronomy setting. This class only used virtual VBOT architecture because we wanted to focus on testing the redesign of the architecture and activities from P1. We planned on using the underlying structure provided by the architecture and activities to compare physical and virtual environments in the final implementation. Our main research questions were:

- How did the class participation patterns differ between a baseline lesson and the VBOT activity lessons?
- Did the target students show increased decentralized and non-deterministic thinking during and after using VBOT?
- How can any observed changes in complex systems fluency or classroom patterns of participation be accounted for?

How did new or improved aspects of the VBOT architecture support observed changes?

Participants

This study included 26 students, each of whom participated in the VBOT activity for 80 minutes a day for 3 days. Three students missed one or more lessons. According to the teacher and administrators, this class was a representative sampling of 6th grade students at the school. The school was an urban public middle school (approximately 40% White (Non-Hispanic), 40% Black, 20% Hispanic, 35% low income).

The students worked both individually and in pairs. Based on both Harel & Papert’s (1990) and Kafai’s (1995) work in which they show the positive interaction effects of pairs work in programming / learning activities, the VBOT activity was designed with students working in pairs to maximize the interaction in the classroom. The class used the orbit, flocking, and Moon-Tag activities described in Chapter 2.

Facilitation and Context

The teacher in this classroom was both supportive and excited. However, as a relatively new teacher, she had not yet mastered whole-class discussions; students did not help each other out-of-turn. There was some evidence of an IRE structure to coverage of formal scientific material. However, the teacher was excited to learn how to create a more engaged and connected classroom.

The author was primary facilitator of the VBOT activity. The teacher helped students with technical issues, organized the classroom, and assisted in classroom management. In addition, two of my fellow researchers aided the teacher and me by addressing technical issues and organizing data collection.

Data Collection

Each student completed a pre-test and a post-test. The tests were designed to assess the students' complex systems fluency and technical fluency. The pre-test and post-test were similar; they consist of sections on robotics, complex phenomena, and flowcharts (meant to be roughly analogous to the visual programming environment). The specific content of some sections was changed slightly from pre-test to post-test to avoid identical responses. The post-test also included VBOT behavior circuit building questions in which the students were required to draw 4 visual circuits on a piece of paper given contextual information and a local goal.

The social network data collection was designed to isolate and describe the interaction patterns of the students. The social network links were recorded in two ways. On the baseline (non-VBOT) days (Day 0), a researcher recorded every verbal interaction in the classroom during the hour-long lesson through field-notes. These interactions were verified through video analysis. The network included only bi-directional conversations, and did not include the teacher, who lectured to the entire classroom and spoke to most of the students individually. This data was collected to understand the interaction patterns in the classroom for further analysis; interaction patterns are discussed in Chapter 6. On VBOT activity days (Day 2 and Day 3), the links were recorded by the students themselves (in a daily questionnaire) and partially verified through video analysis. The change in recording methods was because of the significantly higher volume of student interactions during the VBOT lessons, compared to the baseline lessons.

Both during the activity and on the baseline (non-VBOT) days, the students left their seats only at the beginning or end of the lesson, or to go to the bathroom. Furthermore, the seating arrangement was not modified during the data collection. Thus, it was possible to base the social network diagrams on this seating arrangement; the social network diagrams show students in the same relative positions that they occupied in the classroom itself. All activities with VBOT were logged on a central server and videotaped by a researcher. Two cameras were used at all times, one of which was stationary, one of which captured activity in the classroom and interviews of students. The interviews were designed to elicit information on students' technical and complex systems fluencies as they acted in the VBOT activities. In the interviews, researchers ask students questions about both the technical process and the student's reasons for their actions.

At the end of each of the three lessons, students completed a short (~5 minute) questionnaire. The daily questionnaire was designed to elicit information about the students' social interactions, evaluate the progress of the VBOT activity, and diagnose any material problems with the activity. These questionnaires asked students to describe one element of the VBOT activity that they had enjoyed and one they had disliked during the day. We also asked the student to list the students with whom s/he had discussed VBOT and to list other students from whom s/he had received technical help. Answers to this question were used to create the social network diagrams discussed above. The final question focused on learning by asking students to describe something that s/he had learned in the course of the VBOT activity.

Classroom patterns of participation in P2

Of particular interest in P2 was the change in classroom participation; this led us to try to organize the social space similarly in the final implementation. In P2, there was a highly significant difference in the type and count of verbal interactions in the classroom (n=21, p<0.01). In the observations of the classes both before and after our implementation, the teacher was the primary focus of the class. Students were encouraged to ask questions of each other, but few did. The explicit teacher-driven goal of the students during these classes was to "take good notes which will be graded later." These notes would also provide the basis for a test that the students were to take a week after the VBOT activities ended.

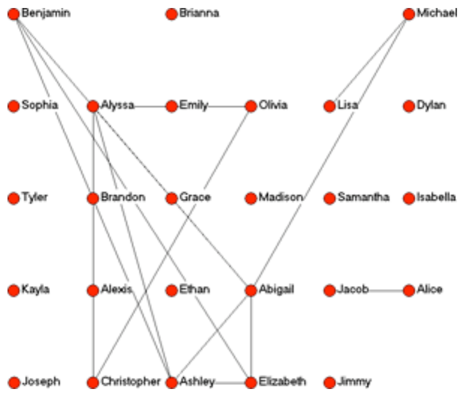


Figure 4-1: Day 0 Spatial Social Interaction Network

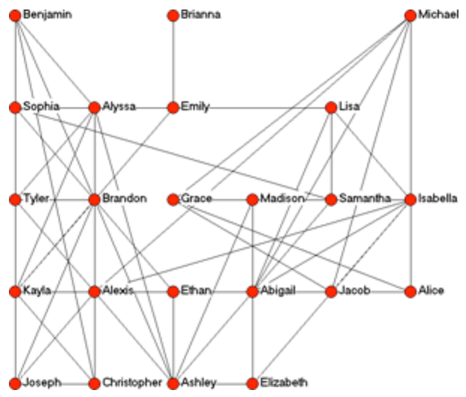


Figure 4-2: VBOT Day 2 Spatial Social Interaction Network

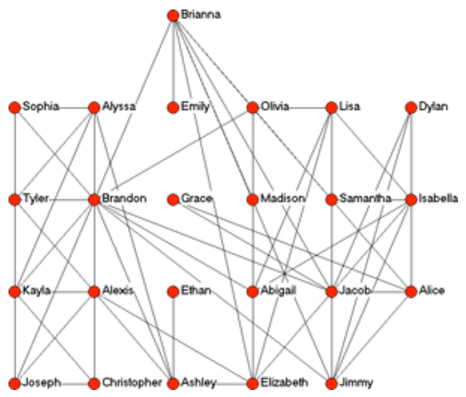


Figure 4-3: VBOT Day 3 Spatial Social Interaction Network

Figure 4-1 shows verbal activity classroom for the class the day before the students began the VBOT activities (D0). Figure 4-2 shows verbal activity during the second day of VBOT activities (D2). Figure 4-3 shows verbal activity on the third (and last) day of VBOT activities (D3). Again, all of the links are two-way conversations.

The classroom dynamic changed in the two situations. In D0 situation, students asked questions only of those students they judged to be "experts." In the D2 and D3 situations, students talked to many of their fellow students surrounding them. Table 4-1 shows the relative density of links on the three days. A density of 1.0 would mean that every student talked to every other student at least once. A density of 0.5 means that half of the possible links were made, where a link is defined as at least one verbal interaction between two students. A density of 0 means that no student talked to another student.

Day	Density of Network
Day 0	0.03
VBOT Day 2	0.29
VBOT Day 3	0.37

Table 4-1: Relative Network Density

Unfortunately, we were not able to collect this data in the final implementation due to the difference in scale. Surveying, identifying, and verifying social network data at this scale in the final implementation would not have been feasible, as collecting it in this manner requires a ratio of roughly 4 students per researcher to collect and verify conversation data over time. Anecdotally, the patterns appeared similar in the final implementation.

4.3 FI – Final Implementation

The transition from P2 to the final implementation involved refinement of the assessments based on our data from P2, refinement of the virtual VBOT system, revision of VBOT activities, and, most importantly, addition of the physical VBOT architecture and robots. We did several brief tests at Northwestern with small numbers of graduate and college students on the physical VBOT architecture, but the final implementation was its first exposure to the classroom. Unless otherwise stated, data collection and assessment were unchanged between P2 and the final implementation.

In this study, we used the VBOT system in two Chicago public schools with contrasting contexts. We worked with two 8th grade classes from a neighborhood middle school on Chicago's northwest side, heretofore referred to as Old Grove. We also worked with two 8th grade classes from a major public high school on Chicago's southeast side, heretofore referred to as Bayville. At each school, one class enacted the virtual VBOT and the other used physical VBOT.

The two schools have significantly different atmospheres and makeup. We selected these schools for their diversity to help demonstrate the applicability of this project to a variety of settings. In this section, we will describe the two schools, the teachers at the two schools, and our data collection methods in the two environments.

FI – Data Collection and Methodology

The primary instrument in this research was classroom implementations using the VBOT system. A researcher co-taught the class with the regular teacher. Each of the four enactments lasted for 5 school days each. This was designed to be a complete weeklong drop-in unit for a middle school science teacher. There were two schools, each of which had one class using the virtual VBOT system and one class using the physical VBOT robotics system. One school had separate teachers conducting the different classes, and the other school had only one teacher for both classes. No students overlapped. Each class was made up of 20-30 8th grade students. Two of the classes worked for one week on physical VBOT systems, and the other two spent the week on virtual VBOT systems.

The students had taken pre-algebra before this pilot, because of the amount of mathematical understanding required to successfully build VBOT circuits. No additional technological expertise was needed or expected. The fill-in unit is designed to teach computational and complex systems fluencies through the mediating subject matter, although no technological changes were made. The teachers led discussions connecting the subject matter to the biological systems content that they were otherwise teaching. Like print fluency, computational and complex systems fluencies address a variety of content domains. Just the act of reading is distinct from the content that is read, the present study does not require the content of the lesson to reflect the apparatus.

Due to the heavy involvement of the research as facilitator of the activities, there was only minimal teacher professional development prior to the enactments. Neither the teacher nor the researchers explicitly taught students to build their robots, but rather helped them to understand the techniques and explore possibilities. The teacher and researcher both worked to encourage as much diversity as possible in building behaviors. The role of the teacher involved organization, management, and facilitation of discussion and reflection. These tasks did not require extensive training with the software, the methods of using VBOT, or leading the activities.

The data collected for this study consists of: 2 videotapes of each implementation day, full activity logs, pre-tests, post-tests, daily questionnaires, pre-interviews of 4 students per class, post-interviews of those same 4 students per class, pre-interviews of all teachers, post-interviews of all teachers, and 3 sets of researcher field notes per class.

The interviews were conducted individually with four students from each class, both before and after the activity. The students were selected to represent a range of gender, ethnicity, and scholastic performance. With this in mind, one student in each class represented each quartile of scholastic performance. The interviews were semi-clinical interviews. The interviewer, proceeding from a set of questions (listed in Appendix 2), allowed the student to digress, but only briefly. Each interview lasted between 20 and 30 minutes. Both pre-interviews and post-interviews consisted of one question on computational fluency, one on complex systems fluency, one on VBOT, and one on the relationship between these topics. In the pre-interview, the VBOT question was adapted such that it is answerable without knowledge of the VBOT GUI.

The classroom enactments were videotaped using two cameras at all times. One camera focused on the activity of the classroom as a whole, observing what is being discussed within the classroom, which student is talking, and the actions of the teacher and facilitator. This is the reference videotape. The other camera focused on one group of students at a time, recording the behaviors and interactions of the students who are using the system in the activity. The second camera was used to conduct "*in situ* interviews," in which a cameraperson asked students individually to describe and narrate their actions as they participated in the activities. For instance, when students used VBOT, the activity interviewer repeatedly prompted the student to describe not only every action taken, but also the motivations for taking those actions. Through the activity interview, one can glean some understanding of how the student is working. While this method is useful, it is limited, because the questions may influence the students' behaviors. However, the information gained was valuable to decipher the other actions taken by the student when examining the activity logs. Furthermore, activity interview data is often self-explanatory to non-expert observers attempting to understand how students are working.

In addition to collecting data through videotaping, researchers took some field notes. These field notes are used orient to the videotape and activity log data but are not a primary source data. Our field note data is not as useful in these implementations because of the high volume of verbal and technical information and the use of the camera and activity logs.

The last form of qualitative data collected was in the form of a daily questionnaire. The daily questionnaire had several uses: collecting self-report data on student learning; self-report data on conversations the students had with VBOT; potential bugs in the technology; and potential problems with the lesson or facilitation. These daily questionnaires were similar to the questionnaires used in P2. This questionnaire is provided in Appendix 2.

The majority of the quantitative data came in the form of activity logs. Every action taken with the VBOT system was recorded and logged to a central server, where an action is defined as any change the state of the student's vbot or VBOT GUI. The students were informed of the logging of the data. The other quantitative data collected were the pre-activity tests and the post-activity tests. As in the pre-activity and post-activity interviews, the questions maintained the same content domain and structure between the pre-activity and post-activity test, although the actual content of the questions differed. Every participating student took both tests. The tests were about 20 minutes long.

FI – Daily Schedule

Each day of the enactments was held in a full classroom for an hour. In both the virtual and the physical VBOT classrooms, we started the day with a short introduction (less than 5 minutes), and then used the technology for the rest of the day. During the activities, students were allowed to talk and interact to the limit of the discretion of the teacher. We list the daily schedules below:

	Virtual Classes	Physical Classes
Day 1	Single-user VBOT. We taught familiarity with the system, basic operation, and did a basic activity we call "orbit." In orbit, all students programmed their vbots to independently orbit a light source. The students learned about light sensors, motors, and mathematical operations.	Single-user PVBOT. We taught familiarity with the system, basic operation, and did a basic activity we call "orbit." In orbit, all students program their robot to independently orbit a light source in the physical world. The students learned about light sensors, motors, and mathematical operations.
Day 2	Multi-user orbit and flocking. We started the networking features, and students tied these into what they knew. Students learned to orbit and flock as a group by learning how to use the robot sensors.	
Day 3	Simple tag. The students built several circuits and used their vbots to play a game called "simple tag." In simple tag, you get points for touching other robots, as many as possible.	Multi-user orbit and flocking. Students learned to orbit and flock as a group by learning how to use the touch sensors with the light sensors.
Day 4	Moon tag. This is game described in Chapter 2 in which students must organize to move rocks from the outside border to the center of the screen.	Bot soccer. This is game described in Chapter 2 in which students must organize to play bot soccer as a team.
Day 5		

Table 4-2: Syllabus overview

Context – Old Grove School

The research was conducted in two classes at Old Grove School, encompassing roughly 50 8th graders between the two classes. One class used only the virtual VBOT environment, and the other class used the physical VBOT robots. Old Grove exhibits strong international diversity. According to the school's website, 26 nationalities are represented in the middle school grades. Among our participants, we had multiple students from every inhabited continent but Australia. This reflects, in large part, the diversity of the northwest side of Chicago. The school is approximately 50% White (Non-Hispanic) with 40% of middle school students classified as low income. According to the principal, the school has risen from the bottom quartile in academic performance of non-magnet middle schools in the city to the third quartile in the last few years. The school is a small K-8 school with approximately 350 students. Both of the teachers that we worked with had been at the school for more than 30 years and were considered leaders in the school, according to informal discussions with the principal and other teachers.

The school serves grades K-8, and it has a warm feel. There is an overall T-shape to the entire school, and everyone must pass through the junction to get from class to gym or lunch. Students of all ages meet in the hallways in what appears to be a friendly and open atmosphere, however open discussion in the halls was discouraged, as it could get congested and loud. The stem of the T holds the gym, library, cafeteria, and science laboratories. One wing of the T holds the older students and the front office, and the other wing holds the younger students. There is art created by students on every open surface, and I had the opportunity to witness presentations by students in the main area. Given the frequency with which these occurred during my short time in the school, they appear to be a normal part of the school culture.

While we were there, there was another research group in the school from University of Chicago, exploring “the most diverse” schools in the city. We had little contact with them, but the school seemed to be research friendly.

Old Grove Class One (Virtual) – Mr. Wilson

Mr. Wilson taught the class that used the virtual VBOT system. He’s been a teacher at Old Grove for more than 40 years, serving in almost the exact same position throughout his tenure. He’s among the most senior teachers at the school, and has been friendly to research projects in the past. In the pre-interview, he described a deep affection for both gadgets and educational technology. He has some plans to retire soon, though he looked and acted younger than retirement age.

One of our first observations about Mr. Wilson’s classes was the high frequency of student discussion, sometimes appearing to nearly get out of control. His students were allowed to talk freely, provided the talk was on topic. He rarely punished his students, but he discussed obvious transgressions with them. Overall, his class seemed highly motivated, and they were excited by the prospect of using robots and games in the classroom. Two students brought a robot of their own to show me on the second day of the implementation, but it was confiscated after they acted inappropriately with it, by running the robot during one of Mr. Wilson's later lectures.

The classroom had information everywhere. Often, a TV and computer were on, while posters about 4 or 5 different domains, either created by students or commercially published, covered every available space. During the implementation, posters fell off of the wall and onto our cameras or camera-people more than once, although no data was lost. In non-VBOT classes, students would occasionally consult a poster or a reference book, but more often, students would hop out of their seats to check something on the computer (if Mr. Wilson was available to supervise).

Old Grove Class Two (Physical) – Mr. Cleveland

Mr. Cleveland has also been a teacher at Old Grove Elementary for about 30 years. For 25 of those years, he taught 3rd grade. He recently made the switch to 8th grade, but he teaches in a manner more familiar to 3rd grade than 8th grade. His class is formal and orderly, and students do not talk out of turn. However, the class also engages in whole class discussions mediated by Mr. Cleveland. Mr. Cleveland considers himself pro-technology, and he spends time keeping up with classroom technology, but I did not witness any use of computers. He liked to use video demonstrations, and he played computer games on his laptop on free periods. Moreover, he had people in his 3rd grade classroom work with educational robotics several years prior, and he had enjoyed using them. He freely indicated that he was excited about VBOT coming into the class.

Mr. Cleveland had a sparse, organized, focused classroom. There was little information on the walls, in stark comparison to Mr. Wilson. From my observations, his students tended to be engaged, although there were a few students who clearly bristled at the structure.

Like Mr. Wilson, Mr. Cleveland indicated his plan to retire within the next 10 years, and he was also quite youthful appearing and quick to laugh and make jokes with fellow teachers. Mr. Cleveland has a clear dichotomy in his persona – he's tough but fair in the classroom, and he's easy going and funny with adults.

Context – Bayville School

Bayville School provided a contrasting environment to Old Grove. The school is a high school rather than a middle school, and it encompasses grades 7 through 12. The school is situated in an urban setting on the southeast side of Chicago. It is known in the city as a good non-magnet school with solid academic performance. The school is approximately 95% African-American, with 70% low income. Halls at Bayville are either lined by lockers or giant murals, though there is an area in the middle school reserved for winners of an art contest. The school is large, even by Chicago standards, with over 1000 students.

All doors are guarded by a metal detector and armed guards, though, by most accounts, there is little violence. One of the students in our study was robbed during school hours inside the school, but all students said this was rare. One of our classes was postponed as students gave depositions to the police regarding this incident, but, by all accounts, this event was unusual.

Bayville Classes One and Two (Virtual and Physical) – Ms. Adams

Ms. Adams taught both the virtual VBOT class and the physical VBOT class at Bayville. She has an unmistakable rapport with her students, who seem almost protective of her. She hasn't been teaching as long as her colleagues, as she became a teacher after a mid-life career change; she has teenaged children who attend the school. She appeared to represent a mentor to many of her students. All of her classes were held in a traditional biology laboratory, with faucets, burners, and lab tables. The room smelled faintly of formaldehyde, although no chemicals had been used recently in the classroom.

She was excited about the possibilities of technology in her classroom. Ms. Adams had actually used LEGO Mindstorms robots before in a summer class she had taught at Bayville. She "never figured them out entirely" and didn't feel entirely comfortable with them at first, although she was pre-disposed to like them. She indicated that she was excited at the possibility of new ideas for her classroom. Moreover, another learning sciences research group at Northwestern was also working with her at the same time in another one of her classes. Classes that I observed before our implementation felt much like a typical 11th grade biology classroom. Students had a much material to cover, and were working diligently to complete it. The purpose of the class, according to Ms. Adams, was to prepare students who would not otherwise be prepared for a tough high school biology class anywhere in the city, including magnet schools. It appeared that the discussions were at a relatively high level compared to the Old Grove classes, with much less student interaction and discussion. I observed occasional labs, in which students measured a physical or biological phenomenon and produced a rudimentary lab report.

4.4 Pre-test and Post-test

In a study of this small size, it is difficult to make claims regarding the statistical validity of performance evaluations. Pre-/post-test measures are used to gain insight into individual students' learning and to find patterns in how those students interact with the VBOT system. Chapter 5 investigates the results of these post-tests and Chapter 6 investigates these patterns. In this section, we describe the pre- and post- test instruments. Moreover, as this study has learning goals of computational and complex systems fluency, our instruments are designed to focus on these areas individually and, as such, we do not aggregate the scores of the different questions. Instead we look for patterns within and across questions. The full text of all questions can be found in Appendix 2.

Question 1 (Agent/Aggregate Understanding)

Question 1 was designed as a basic complex systems competency question. The question was designed by using Jacobson & Wilensky's (2006) hierarchy of complex system understanding. The question presents a picture of birds flying in a V-shape (see Figure 4-4); one of the birds as an arrow pointing to it with the name "Shelby." The name Shelby was chosen because as a gender-neutral name. The description of the picture in the question reads, "When birds fly south for the winter, they often form a V-shape. You might have seen this in the sky. This is a picture of flock of birds flying in V-shape."

The first question that follows the text (Q1.A.) reads, "How does Shelby, the bird, know where to fly in the V-shape?" The use of the word "know" implies there's a conscious decision on the part of Shelby the bird to fly in the V-shape; this is meant to be subtle discouragement from answers such as: "s/he doesn't know", "it is instinct", or "s/he just does." Wilensky & Resnick (1999) showed that most people respond to questions of biological systems from a deterministic-centralized perspective; that is, they posit that a leader controls the system.

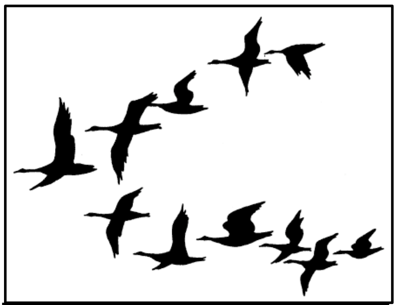


Figure 4-4: Shelby flocking

Q1.B asks, " How do the birds know where to go when they are flying in a V-shape?" Note the perspective difference between asking how Shelby flies and asking how birds fly; this question is designed to elicit answers that consider that aggregation, distinct from the agent-based perspectives in Q1.A.

Q1.C asks, "Why do bird fly in V-shapes?" This question was designed to elicit students' opinions on the value of aggregation to an individual agent. Levy & Wilensky (2008) describes student understanding of the relative utility of acting as an agent in an aggregation. Her work informs the design and assessment of this question by categorizing middle school student understanding of the different facets of agent, aggregate, and "mid-level" understandings in biological systems.

Question 2 (Flowcharting)

The VBOT system uses a flow metaphor towards circuit-based logic programming (as discussed in Chapter 2). Toward that focus, Question 2 is a flowchart question in which students had to both interpret and modify a simple flowchart. This question was designed to measure competency with flow diagrams in general. Question 2 was designed to evaluate a students' ability to do simple branching logic (see Chapter 2) and program logic. This question begins with self-reportage by the student of whether s/he has ever "seen a flowchart before." We can evaluate a students' perception of his or her competency along with his or her actual competence.

Question 2.B. asks the student to read the flowchart as a list. If the student understands flowcharting, the problem is simple. Indeed, students either clearly succeeded or clearly failed on this question. Less that 2% of answers were judged partially correct.

Question 2.C. asks the student to modify the flowchart and add one operation and one leaf to the flowchart. This question is designed to determine a student's ability to modify flowcharts. To complete the answer, a student can add one directional arrow and one hexagon. We hypothesized that the different ways in which students incorrectly approached this problem might lead us to better understand patterns of student interaction with the VBOT system. Again, however, there was little ambiguity with less than 2% of answers receiving partial credit. The vast majority of students either answered the question entirely right or entirely wrong. The wrong answers mostly stemmed from an inability to connect the flowchart "wires" properly. This is a skill directly addressed with VBOT.

Question 3 (Sensors & Motors)

Question 3 was designed to test student understanding of sensors and motors. However, due to poor wording, the students' answers proved unhelpful to test any competence or understanding. Less than 10% of students' answers mentioned sensors or motors in any way. It is included in the Appendix 2 only for reference.

Question 4 (VBOT)

Question 4 requires students to build a VBOT circuit on a paper facsimile of an actual VBOT circuit board. Question 4 describes four different scenarios and asks students to sketch out the circuit they would build for those scenarios as seen in Appendix 2. The four questions were:

- 4.A. Wire up a vbot to go in a loop around the screen.
- 4.B. Wire up a vbot that would make a smaller loop.
- 4.C. Wire up a vbot that makes either loop using NO LIGHT SENSORS.
- 4.D. Wire up a vbot that makes either loop using NO VBOT/BUMP SENSORS.

Although the transition from the technical environment to the paper post-test is difficult, a student who can program advanced circuits in VBOT should be able to answer all four questions. On average, students answered two questions right and two questions wrong. In this question, there was only limited partial credit available. Only one of two or three circuits would be correct answers for each of these questions. If the answer corresponded to those circuits, students earned credit; if not, they did not. On every answer that was non-standard (around 10% were neither obviously right nor obviously wrong), the circuit was tested in VBOT. Only three of these circuits earned partial credit.

Chapter 5: Student Tinkering and Student Sharing

In this chapter, we examine the ways that tinkering and sharing are interrelated with performance and activity in VBOT. Towards this end, we will examine four students in more depth, reviewing the ways that the student shared or tinkered, and evaluating that relationship. There is a fundamental argument that by sharing and tinkering, these students came to understand VBOT better, came to succeed on post-test measures, and showed reliable understanding of the target material.

The chapter is structured in the terms of computational fluency described in Chapter 3. In that chapter, computational fluency is described as being cognitive, material, and social in nature. Cognitive fluency is the ability to think computationally; material fluency is the ability to manipulate computational artifacts; and social fluency is the ability to communicate effectively about computation. Toward computational fluency, we evaluate students on their ability to build circuits in VBOT (material fluency), to use VBOT understanding in disparate tasks (cognitive fluency), and to communicate with both their teachers and fellow students about VBOT (social fluency). Throughout the chapter, computational fluency will be specifically highlighted and evaluated.

This chapter uses three main sources to evaluate claims of relative computational fluency:

student logs, interviews, and pre-/post-test data. Case studies are constructed from a combination of interview data, log data, and supporting classroom video data. The case studies are intended to highlight aspects of social fluency through communication and cognitive fluency through process. Process data is generated by following the students' thinking as they attempt to solve tasks; this is used to describe the relevance to their cognitive fluency.

The four case studies represent particularly rich data sets in which tinkering or sharing is clearly manifests. No argument can be made for the "typicality" of these students. Recall from Chapter 4 that we selected students to interview based on a range of teacher-perceived aptitude and gender. In this chapter, we only discuss students from the middle and lower end of the teacher-perceived aptitude. As this is a biased selection, we present statistics about all of the students in order to show the relationship between these data and the remaining students. It is interesting to note that we found little significant correlation between pre-test and post-test scores, and no correlation between gender and performance.

The case studies should be read as exemplars of particular modes of using VBOT. The first case study shows Patrick, who tinkers in VBOT. Patrick's activity shows a path through the VBOT activities that maximizes tinkering with technology in order to understand it. His methods of using VBOT show the ways in which using VBOT can help one to think computationally. Joshua, on the other hand, shares much more than he tinkers. Joshua's case is an example of one way that a student that is not normally very social might use VBOT as an apparatus to disseminate and collect information, in turn gaining social computational fluency. Alicia and Dania are then presented to show two relationships between tinkering and sharing and the ways that tinkering might affect sharing and vice versa. In each section, statistics are presented to show the correlations between tinkering, sharing, and performance for the classes.

The vast majority of students learned how to program in VBOT during the VBOT intervention. No student knew how to program in VBOT on day 1, and only two students showed any familiarity with even the most basic computer programming skills. By the final day, all of the students could build functional circuits in VBOT. While the data presented demonstrate these learning gains they do not necessarily suggest the causal argument that tinkering and sharing increased performance. Instead, the data show the ways that tinkering and sharing are different methods by which students exhibit and gain computational fluencies.

5.1 Tinkering

The term tinkering derives from Hancock (2003), who describes different methods in which students use live-programming systems. This, in turn, is derived from Turkle & Papert (1991), who detail a plurality of methods, such as *bricolage*, with which students can program effectively. Turkle, Papert, and Hancock argue that the methods of tinkering and degree to which students tinker reflects the manner in which they understand the task of programming. In that way, by analyzing and measuring the tinkering of students, we are evaluating the cognitive fluencies of those students, as described in Chapter 3.

We define tinkering as any manipulation of the circuit that results in a semantic change in the circuit. A semantic change is any change that produces different behavior in a vbot. Since some of the students spent far more time on task and built far more circuits (Avg. circuits built: 1127, $\sigma = 906$), it was not appropriate to measure tinkering in terms of number of circuits built. Thus, tinkering was measured in terms of the number of semantic changes divided by the number of total interactions with the system. Measuring tinkering in this way illuminates the rate that students were changing circuits over the entire time that they were using the system. That is, as each group used the system differently and for different amounts of time, it was ineffective to look at raw measures instead of averages.

However, looking at averages negated the value of "more free play." Unfortunately, the design of the study prohibits any clear separation of "free play" and "directed play," as discussed in Chapter 3. As such, raw time on task is factored out. To illuminate this point, consider that in virtual class, one group produced over 4000 individual circuits over the course of the week, while another produced only 1000. Both groups took part in all activities. The range of circuits produced, and the amount of circuits produced varies widely. This tinkering metric averaged 0.97 tinkers per circuit, with a standard deviation of 0.18.

Patrick, Tinkerer – Virtual Class, Old Grove, Mr. Wilson

Patrick is used as an example of a student tinkerer. Patrick was a shy, curly-haired, short student who smiled often and laughed quietly. He was one of the few white, American students in the mostly international class. Patrick is slightly overweight, and every day of the activities, he wore essentially the same outfit: jeans with a t-shirt with an open plaid button-down over the t-shirt. His manner implied mischief, but he didn’t seem interested in attention, and he was enthusiastic about our efforts. Patrick often grins in a knowing manner, but he was relatively reticent when questioned directly. Patrick is one of the lowest achieving boys in the Mr. Wilson's class.

In response to pre-test question 2 (flowcharts), Patrick was the only student in his class to mark that he did not have any experience but also correctly finish and modify the flowcharts. Unlike Patrick, most students reported experience with flowcharts, but only correctly answered fewer than half of the flowchart questions. This implies that while Patrick lacks confidence, he is comfortable with the types of logic used in flowchart diagrams.

During the pre-interview, Patrick showed an aptitude for detail. His pre-interview task was organized as such: Patrick received three yellow notecards marked "IF", "THEN", and "OR ELSE," which corresponded with three blank blue notecards. He was then asked to teach a robot how to spread out. His answer was "IF you are greater than or equal to 5 feet from any object, then spread out, or else do not spread out." When I asked him how this hypothetical robot would or would not spread out, he said that the robot "could move away from the object until it was 5 feet away from that object." His answer was among the most "programmatic" of all of the pre-interviews, in that it gave a specific set of instructions to a robot that were triggered on a discrete condition. His pre-interview and post-interview both show that he thought carefully before answering each question. His answers were specific in a way that implied he was familiar with computers, though he did not have any apparent background knowledge about robots or programming.

Throughout the intervention, Patrick's group was unique in that they spent the first four days tinkering with a few high level operations, exploring several unique circuits with each operation. Out of the groups in his class, his had the second highest percentage of high-level operations per circuit and the second lowest percentage of unique operations per circuit. Most of the operations that he added to the board were high-level operations, and there were generally few operations on the board at a given moment. That is, he would try out the more advanced operations, such as IF, with only a few other operations on the board at the time. This implies that he was engaging in a careful, controlled exploration of these operations, as he would create low-density circuits that focused on testing out the use of an operation. These circuits would generally be less useful, because high-level operations in general do not work independently; they are logical operations that require other operations to be useful.

Across the students we generally see two types of tinkering; students either ‘accreted’ or ‘tested’ operations. When students accrete operations, they add them to an existing circuit. When students test operations, they must use a nearly clear or newly wiped board (for more explanation on this, see Chapter 2). Patrick's group tested operations repeatedly and rarely accreted them. I'll further discuss these classifications of students in the next chapter.

Patrick improved the effectiveness of his tinkering by sharing his circuits. On the final two days of the simulation, he began to share his ideas with his two sets of friends in the classroom, all of whom were boys. The friends designed a set of behaviors together to solve a set of challenges that were posed by the game. Working together, they came up with circuits that were both appropriate for their local goals and included advanced VBOT operations and features. The groups' circuits all changed together in similar ways while remaining independent and unique for the majority of the lesson. Once another group, made up of girls, started doing well in the game, Patrick went to them to ask how they designed their circuits. Showing good comprehension by this point, Patrick followed their example in building a new type of circuit more targeted to the lesson. Though the teams only communicated once about the design of their circuits, both eventually mastered an ad hoc language in order to verbally communicate their circuit design.

Patrick did well on his post-test. He got 3 out of 4 VBOT questions correct, and he correctly answered all of the flowchart questions. Even though he was regarded as a sub-par student, his interest and the adaptability of the tool to his apparently natural tendency to “tinker” allowed him to excel. On his post-interview, Patrick expressed that VBOT had been "really cool!"

Tinkering and performance

Using our specific tinkering metric, we found that tinkering was positively and significantly correlated (n=33, p< .05, F=5.0) with post-test VBOT programming performance (question 4). We performed a linear stepwise regression of post-test performance against pre-test performance, the tinkering metric, classroom-effect, teacher-effect, and school-effect. In that case, only tinkering was positively correlated. All 33 groups across the 4 classrooms were included in this regression. Pre- and post-test performance were averaged by group.

Essentially, this implies that students who tinkered more were better at building VBOT circuits, regardless of their teacher, classroom, school, or pre-test scores. This does not show that tinkering caused these students to do better, but that there exists a more robust reason for how and why they tinkered than a student's previous performance or the effect of the teacher. Indeed, the standout students weren't necessarily the students who were identified as being "good students." Patrick serves an example of this. Moreover, Patrick's example provides a possible argument as to how tinkering might improve a student's computational fluency materially, cognitively, and socially. Patrick improved materially: he did very well on the post-test, building working circuits. His cognitive fluency was indicated by his scientific approach to the circuit building, as seen in his testing out of complex operations in simple circuits. His social computational fluency is exemplified by his improved communication with his fellow students when describing and discussing circuits.

5.2 Sharing

Sharing in VBOT is defined as two groups building circuits with the same semantics, whether intentionally or unintentionally. Our sharing metric measures how many of a student's fellow students in the class had a circuit with the same semantics. For instance, if student A created a circuit in which the left motor was set equal to the right light sensor, and student B created a circuit in which the left motor was set equal to the right light sensor plus zero, the two students would have the same circuit, semantically, and they would be counted as "sharing." However, it was intractably difficult to measure how many students actually "shared" circuit by discussing them or looking at other people's circuits, in classrooms – the crosstalk of 30 students in a classroom is indecipherable. Furthermore, even this analysis would overlook the number of students who looked at each other's screens. Without expensive and sophisticated equipment, no "human" model of sharing was measurable. Thus, we use the computational analysis of semantically identical circuits to analyze this.

Some educators might argue that this is "cheating" rather than "sharing". To rebut this, we argue that we encouraged the copying of circuits, if only because the nature of the activities was that any two agents will need to have a different set of behaviors to do well in any of the tasks given. Only on day one and part of day two would it be beneficial to have the same circuits as everybody in the classroom. However, variations on circuits are consistently useful to the students. To this end, sharing implies adoption of a circuit paradigm rather than a direct copy.

Joshua, Sharer – Physical Class, Old Grove, Mr. Cleveland

Joshua is a student who shared circuits extensively during his VBOT experience. His story illustrates how sharing might organize the VBOT experience. Joshua is an athlete who often wore basketball jerseys and spoke to the researchers of his experience on the middle school basketball and baseball teams. Joshua is of middle-eastern descent, and he has curly hair. His slight build and height makes him seem meek. During the VBOT intervention, he did not often speak informally to his classmates, though he was consistently good-natured. During the intervention, he endured constant teasing, often because he was the only student from his religious minority in his class, which is a fact that he also mentioned in the interview. He expressed feelings of alienation from both school and his peers, and was referred to as "a slower student" by his teacher, who said that he got B's by "trying really hard." During the pre-interview, he was reticent, and almost every answer he gave was a simple "yes" or "no." As a result, he did not successfully finish the IF/THEN question on the pre-test.

Joshua's pre-test performance was poor; not only did he fail to complete the IF/THEN programmatic question but he missed every flowchart question. This was likely reflective of his attitude during the initial day and the pre-interview: he was hesitant, and when asked whether he was enjoying himself, he indicated that he did not understand why we were using robots.

Joshua's group was made up of students who did not appear to be his friends. They were a group of the "odd" students in the class – two male students had long hair and another wore black tshirts with gruesome horror movie scenes on them. Joshua stood out from his group by looking “normal.”

As such, it was notable that this group of “misfits” shared with other groups significantly more than any other group. Not only did they share circuits, they often employed a rotating member to talk to share ideas with other groups. This worked well in the physical robots classroom, as it took more time to build a physical circuit, and students could discuss the circuits before downloading them onto robots. On the third day, the students learned how to use IF, at which point, they started testing out their own IF circuits. Joshua discussed the task with his group, leading them to attempt a novel way of using separate IF statements for each of the two motors. After learning this method, they then shared their progress with the group of “popular” girls sitting across the room from them. The group of girls then learned how to use both constants and sensors in their multiple IFs and relayed this information back to Joshua’s group.

Joshua excelled on the post-test. Every flowchart question was correct, and the following three VBOT programming questions out of four were entirely correct: question 4A (light-following), question 4B (about using the CONSTANT operation), and question 4D (creating a multi-tiered circuit). All of the answers were canonical and appropriate. During his post-interview, he expressed that the activity was "pretty fun." He built several correct circuits, and described them in detail.

Sharing and performance

Joshua provides an example of a student that, through his motivation to share his circuits, excelled in VBOT and on post-test performance measures. Although Joshua learned VBOT through sharing, this was not necessarily the case for every student. There was no significant correlation between sharing and performance given tinkering. We performed linear stepwise regressions of post-test performance against pre-test performance, the tinkering metric, the sharing metric, classroom-effect, teacher-effect, and school-effect. The effect of sharing was not significant. However, tinkering and sharing were positively and significantly linearly correlated across student groups ($n=33$, $p<.01$). As stated in the last section, tinkering was correlated with post-test performance, but tinkering and sharing are correlated enough to make them effectively co-linear when regressed against performance, so the sharing metric effectively did not factor in post-test performance.

The implication here is that students that shared more, tinkered more and, hence, did better on post-test evaluations. This has positive and negative implications: the positive implication is that the activities stimulated learning these fluencies, which helped students excel; the possibly negative implication is that it is a student-to-student finding and our findings possibly reflect only those students most "interested" in the material. In truth, the data set is far too small to resolve these positions adequately. In the next chapter, the relationship between sharing and patterns of behavior will be explored in the context of virtual and physical robotics.

5.3 Tinkering, Sharing, and Circuit Quality

Relative material computational fluency can be understood in terms of the relative qualities of students' circuits. However, measuring the "quality" of a circuit that was built by a student is problematic because it is necessarily extremely contextual. There is no way to judge whether a circuit is the optimal circuit for a given task in an open task environment because there is no acceptable way to determine a students' local task. Thus, we measure quality in terms of whether it successfully moved a vbot, the complexity of the wiring and number of operations in the circuit.

Developing a unified "quality" metric is problematic, as there are infinite valid measures of the "quality" of a circuit. To be more helpful in determining the actual expertise involved in the circuits, several different metrics were used, each with a particular focus. They shared one similar feature: they all only counted those circuits that successfully moved a vbot. In the VBOT system, one builds many intermediary circuits while simultaneously building useful circuits, which corresponds to the "tinkering" discussed earlier in this chapter.

"Density" (OP_DENSITY), which is the first quality metric that we use, is the number of wired (i.e. working) operations on a vbot circuit board divided by the number of circuit boards. It is a simple metric, but it gives us an idea of how "big" a circuit has been built. In the case of the basic light-finding circuit described in Chapter 2 ("RM = LL ; LM = LR ;"), the operation density is 4, as there are four wired operations on a working circuit. OP_DENSITY could potentially provide insight into the manner in which a student creates a circuit. Very high-density circuits (circuits with a high OP_DENSITY) are circuits that accrete operations over time out of previously working circuits. It takes a relatively long time to build a dense circuit in VBOT; the environment privileges smaller circuits in the UI (see Chapter 2 for more explanation). Therefore, a student with high OP_DENSITY is building circuits in with a higher relative "time per circuit." However, as the student's median time is exactly 3.0 seconds per unchanged circuit, there is no actual measure of "time per circuit." (The 90% trimmed mean is 6.70 seconds per circuit, as there are a significant number of outliers.) This is because that "unchanged circuit" metric only shows the time between any modifications of the board during play.

"Operation Difficulty" (OP_DIFF) measures the number and amount of the "high-level" operations that a student uses per circuit. Logic operations (e.g., IF/THEN) are weighted double that of arithmetic operations (e.g., ADD), which are weighted double that of sensors and motors. Therefore, logic operations are four times more "difficult" than sensors or motors. OP_DIFF is a metric designed to measure the "adventurousness" of a student. Students with a high OP_DIFF are likely to have tried out all of the operations and used high-level operations frequently. OP_DIFF could be a measure of the depth of material fluency with the project. Students with a high OP_DIFF are using complicated circuits to achieve local goals. A student with a high OP_DIFF and a high OP_DENSITY would likely be using several complicated operations simultaneously. A student with a low OP_DIFF and a high OP_DENSITY is building a complicated circuit out of simple components.

“Unique Operations” (OP_UNIQUE) is the number of unique operations per circuit. For instance, a circuit with the left light sensor (LL) attached to the right motor (RM) through an ADD would have an OP_UNIQUE of 3. A circuit that routed signal through two ADD operations would still have an OP_UNIQUE of 3 (ADD = 1, LL = 1, RM = 1; 1+1+1=3). OP_UNIQUE adds focus to OP_DIFF and OP_DENSITY. A student with a high OP_UNIQUE and a high OP_DIFF is creating a complicated circuit. A student with a low OP_UNIQUE and a high OP_DENSITY is creating a complicated circuit out of simple repeated elements.

These metrics help identify not only metrics of performance, but also patterns of student behavior. In the following section, we will refer to each of these metrics and detail how they relate to performance in the activities and on the post-test.

Alicia – Virtual Class, Bayville, Ms. Adams

Alicia is a quiet and polite student. She is short, mildly overweight, and African-American. She dresses in muted colors, and she talks so quietly that it is hard to hear her speak in class. Only one member of her class was not African-American. Alicia was so quiet that few teachers noticed that she was failing all of her classes. Alicia’s teacher described her as “sometimes getting near [to failing] but then pulling back, always getting a D.” This information fell in line with her academic record, which was situated consistently in the D/F range. According to her teacher, Alicia was at the bottom of her 8th grade science classes, and there was significant concern that she would be required to repeat the grade or drop out of school.

Notably, she was both cheerful and interested in the robotics questions in her pre-interview. She did not fit the traditional description of a failing student. In fact, she seemed interested, bright, and thoughtful. She gave careful attention to every question, and she answered each of them correctly. For example, her pre-test IF/THEN answers were not as programmatic as Patrick's, but they were clear and showed an understanding of basic computer skills, though she missed every flowchart question on the pre-test.

Alicia's experience presents a different model of tinkering that we have seen previously. On her first questionnaire (see Chapter 4), she remarked that she had enjoyed the entire activity, especially “creating the wires.” Indeed, early on, her group tested wires and operations that the rest of the class had not yet investigated. Alicia much preferred playing with the value of the CONSTANT operation in order to guide the vbot directly instead of creating a circuit in which the vbot responded to sensor data programmatically. In VBOT, it is possible to change the value of a CONSTANT operation on the fly. As such, a student can wire a CONSTANT to each motor and change the value of the constants frequently to create the desired behavior. This, however, is taxing and much slower that creating a programmatic circuit. Alicia’s group was the first group to understand this change. At that point, this idea of creating programmatic circuits spread around the classroom, and other people tried it. However, while most of the other teams worked further on programmatic circuits, Alicia's group returned to the CONSTANT method, on subsequent days.

Alicia's group shared their circuits with their friends with some mutual benefit. After some time, Alicia's circuit was working well enough to convince her friend Tanisha to use a similar circuit, and the two of them used this new circuit during the majority of day 3. Of particular interest on day 3, during a game called "simple tag," Alicia's team performed poorly. In simple tag, one vbot (V1) begins play tagged, and when V1 touches another vbot (V2), V1 gets a point, and V2 becomes tagged as well. At this point, both V1 and V2 attempt to tag more people. A goal of the game is to be tagged early and then tag other people. Another team, led by a student named Maple (not the team initially tagged), won the first game handily with a simple circuit. At that point, Maple's seating neighbors asked what circuit they used, and the circuit spread quickly around the classroom, with various groups adding small modifications to their own circuits. At the end, only Alicia and Tanisha were using an entirely different circuit, and Alicia's team was unable to excel. Alicia reflected on this in the post-interview as "confusing," remarking that she did not enjoy this game. Thus, in this case, we see a lack of sharing resulting in student frustration.

During the last two days, Alicia's team opted for an entirely different set of strategies. They created simple circuits, involving only a MULTIPLY operation, a CONSTANT, and the sensors.

These circuits were relatively simple as they did not employ logical operations, many simultaneous operations, or many wires. They did not again use an IF after being taught it, but they ended up doing well in the sticky tag game, even winning a round. Again, she shared her circuits with her friends and they did likewise. While she had initially only talked to her group-mate and immediate neighbors, she actually communicated down the row of computers with the more reticent groups during the final days of the study. In the end, she did not tinker much beyond her two different circuit models, but only two other groups shared their circuits more than hers, as she had shared so extensively on the final days of the study.

While Alicia's post-test was uneven, she clearly distinguished herself as reasonably competent and engaged. Her answers to flowchart questions were entirely correct, and she was able to build the simple circuits in question 4. However, she had some problems with the difficult circuits. Her post-interview was encouraging – she had done well, her mistakes were much more technical than conceptual, and she was both animated and excited. She understood and was interested in the assignments, but lacked the math fluency to excel on the tests. And, although it is probably unrelated to this study, Alicia also managed to pass 8th grade.

Dania – Physical Class, Bayville, Ms. Adams

Dania provides another example of the complex relationship between tinkering, sharing, and circuit qualities. Dania was a basketball player, and she claimed that as her primary identity. She is tall and has a strong build. She's African-American, as was every other member of her class. During the activities, she repeatedly made references to basketball and adopted an attitude indicating that, though she may not know much about technology, she could easily beat anyone on the basketball court. Indeed, she was tall and exuded an unmistakable athletic self-confidence. Her teacher called her an "average-at-best student who does not try hard," and, despite her self-confidence, Dania often tended to be overshadowed by her more boisterous class. She was keen to unite the class and expressed disinterest in the "drama" of her classroom.

As such, Dania was the girl chosen to work in a group with two problematic boys and one high-achieving boy. She was immediately designated "team leader," but she clearly vied for control with the high-achieving boy, Abdul. Abdul and Dania often worked together and both took time to explain the programming to their two group-mates, although the group-mates were generally resistant.

Dania did poorly on the pre-test. She was one of only 3 people in her class who did not get any part of the flowcharting question correct. From the pre-interview:

> *Interviewer: You can say 'IF something THEN something else OR ELSE something else' and your goal is to create a spreading out robot, a robot that spreads out, what would you tell it to do?*
> *Student: Can this be more than one word?*
> *Interviewer: Yeah, absolutely.*
> *(10 second pause. Student is staring at the notecards.)*
> *Interviewer: There's no right answer or anything. This is not a test… just think aloud for me. What are you thinking of?*
> *Student: It's kind of difficult trying to start with IF or even start with any of the words because the only thing I can think of is 'If you're in a bunch, then…'*

Interviewer: That's OK.
Student: But then it wouldn't make sense with THEN.
Interviewer: Why?
Student: Because I want to say 'If you're in a bunch, then…' .. Oh! OK. I could, I guess.
(She writes on the notecards: "IF you're in a bunch, THEN you have to move to an empty space.")
Student: I do not need an OR ELSE.
Interviewer: OK.

From the pre-interview, we can see that she had some intuitive understanding of the nature of branching logic, but that she did not have the correct terminology nor confidence in her understanding.

Dania's tinkering stands in contrast to the other student groups in the class, as her group tinkered mostly with basic, simple circuits. This tinkering was of a different sort than Patrick's or Alicia's groups. The group would hone a small number of circuit paradigms and test them in various scenarios. Furthermore, they would modulate a small number of circuits built around CONSTANT operations (literal numbers) rather than sensors. Then they would tune those numbers for the task at hand. In the final two days, in which we played competitive robotics games, Dania's group switched back and forth between two basic circuits: one using two IF operations and one using two CONSTANT operations. They did well in the competitions, even winning one. Only on day 3, after Dania’s group experienced difficulty implementing the IF operation, did Dania's group share her circuits with any other group. Unfortunately, Dania talked to the members of the nearest group, who also failed to master the IF operation. Consequently, both teams built incorrect IF statements for the remainder of day 3. On day 4, Dania and Abdul worked out the IF operation together and were able to do well in the "Bot Soccer" activity that they played on days 4 and 5. However, Dania's group did not share this information with their neighbors, and the neighbor group did poorly in Bot Soccer. In this case, sharing was not necessarily beneficial.

Dania's post-test scores were good. She did better than any of her group-mates on the flowcharting questions, missing none despite being one of the only students who had missed them entirely on the pre-test. She also did reasonably well on the VBOT programming questions on the post-interview, getting two of the four questions entirely right: question 4A (light-following) and question 4C (about advanced operations). She received partial credit on question 4B (about using the CONSTANT operation) and missed the question 4D entirely (creating a multi-tiered circuit), though only two people in her class received any credit on the question 4D.

Oddly, in the post-interview, when asked to create a circuit with an "IF," she claimed that she did not know what an "IF" operation was even though she had both used them extensively and correctly answered the related question on the post-test. When asked simply to "try it out," she exclaimed, "OH!" and proceeded to make a correct circuit with an "IF" operation. This hesitation indicates that she was intimidated by the terminology more than the ideas (as she implied in the pre-interview).

Circuits and performance

Tinkering was positively and significantly correlated with the sharing metric, post-test performance , and OP_DIFF (n=33, p<.05, F=7.7), when regressed against post-test performance, pre-test performance, school effect, teacher effect, class effect, sharing metric, the average difficulty of the circuits (OP_DIFF), the use of many unique operations (OP_UNIQUE), and the density of operations on the circuit board (OP_DENSITY).

The correlation of OP_DIFF and tinkering implies that students who attempted to use more difficult operations also changed their circuits more in the process. Intuitively, this suggests that there was a set of students who tinkered and tried to create difficult circuits, and, as a result, did well on the post-test. This is unsurprising. What is, perhaps, more surprising is that these were the same students who shared their circuits more, and that these circuits were more similar to other people in the class. That is to say, one conclusion might be students who used VBOT do not conform to the stereotype of the "solitary programmer" and that the environment supported social computational fluency. This theory is strengthened by the finding that tinkering was a more powerful predictor than sharing, as seen above. Another conclusion might be that students who shared more learned more quickly, and that the learning motivated the tinkering. This is seen in the case of Joshua above. In both of these scenarios, VBOT is encouraging either tinkering or sharing toward post-test performance. It is important to note that, again, pre-test performance was a poor predictor of post-test performance.

Post-test question 2 (flowcharting) was significantly correlated with both pre-test question 2 (n=78,p<0.05) and the OP_DENSITY (n=33, p<.01), in a linear stepwise regression with post-test performance, pre-test performance, school effect, teacher effect, class effect, sharing metric, OP_DIFF, OP_UNIQUE, and OP_DENSITY. The difference between the pre-test and the post-test on Question 2 was highly significant (n=78, p <.00). The mean for the pre-test was 3.7, and the median was 4 out of 6. The mean for the post-test was 4.8 and the median was the full 6 points out of 6.

These statistics are perhaps less suggestive than they might appear, as the median score for post-test two was 100%. On the whole, students did very well on the post-test flowcharting question. Post-test performance on flowcharting is the only instance in which pre-test performance is correlated with post. The correlation with OP_DENSITY can be understood by considering that more operations require more wires (see Chapter 2 for more details). Wires in VBOT are similar to connector arrows in a flowchart. Hence, a student that used more wires was better at connecting flowchart items.

5.4 Teacher, Schools, and VBOT

The statistics and examples presented here appear to hold true across contexts; there was no measurable correlation of any metric of achievement between teachers or classrooms. In every case where there was a linear correlation between teachers, it was accounted for by the difference between the virtual and the physical classes. This held true for circuit metrics, post-test metrics, and qualitative metrics. Statistically and qualitatively, most differences occurred between virtual and physical implementations and not between teachers or schools. This is notable due to the apparent differences between the school environments and teaching styles. Bayville is a south-side Chicago school, 97% African-American, and designed to operate as a high school. Old Grove is a north-side Chicago school with a hugely diverse international population, which is designed to operate as a middle school. Furthermore, the same teacher taught both of the Bayville classes, whereas two different teachers taught the two Old Grove classes. Whether measured across teachers or classes, the differences in performance were not measurable when the virtual/physical class distinction was taken into account.

5.5 Gender and Performance

Furthermore, the different student learning gains do not appear to be related to gender. There was no significant difference between male and female students in general pre-test to post-test performance measures (Avg. male post-test: 19.6, $\sigma = 4.37$; Avg. female post-test: 19.4, $\sigma = 4.51$). There were both good and bad male and female students.

In this work, the lack of gender differences may be partially attributed to the “newness” of the experience for the students. Students do not necessarily require background knowledge or background motivation to do well in VBOT. As is stated by several of the students in their respective post-interviews, the experience was “different than [they] expected.” Only 2 of the students across all four classrooms had had any meaningful contact with “robots” or programming before.

Even on the pre-test of relevant content, there was not any significant difference between genders or classrooms. The only question that showed a significant difference was Question 2A (“Have you had prior experience with flowcharts?”) This question was weighted towards the female students (who were more likely to claim familiarity, out of 4 degrees of familiarity possible, female students avg.: 2.90; male students' avg. 2.58). That answer did not correlate with student performance on the flowchart questions (pre-test or post-test, showing a slight negative regression slope of –0.157, $r^2 = 0.003$). As such, it has been discounted.

5.6 Understanding Sharing, Tinkering, and Performance

In this chapter, different cases were presented to highlight the ways that different students shared and tinkered in VBOT, and these cases were compared to the performance of the entire population of students. The relationship between sharing, tinkering, and performance appears to be relatively robust. There are two salient conclusions that one might draw from these data: that tinkering and sharing were causally related to performance, or that students predisposed to tinkering and sharing were predisposed to perform better. Although this question cannot be conclusively answered, there is significant evidence that the students who performed well in VBOT were not those students who performed particularly well on the pre-test, pre-interview, or in their science classes prior to VBOT.

Dewey (1897) notably suggested that common methods of schooling often privilege a certain mode of thinking. This study further suggests that situating VBOT in a tinkering and sharing frame privileges different students than traditional school often does, as we could not reliably predict performance from any given measure. It appears that female students did not necessarily share more and that male students did not necessarily show more technical interest or performance. Teacher and school effect were effectively non-existent, and 100% of the students learned at least basic programming fluency in the span of 5 days.

These data prompt one to ask why these factors were not particularly important in this study. One significant possible reason is the simple “newness” of the material; students did not have any background for the context that was given to them and they did not know how to position themselves. A pessimistic response is that these data simply overlook some crucial factor, or that our metrics were insufficient for evaluation. An optimistic reviewer might posit that students did well because it was “a fun game.” While “fun” is notoriously hard to quantify (Csikszentmihalyi, 1990), the students were nearly constantly both competing and collaborating actively with

VBOT although they were not required to be active at all. In the next chapter, we begin to address this question by discussing the relationship between the design of VBOT and the experiences of the students, specifically focusing on the contrast between the virtual and the physical VBOT systems.

The virtual-physical distinction was strong. The next chapter will address the relationship between the virtual, the physical, and the social as they reflect the design of constructionist learning environments. The experience of the classrooms was perhaps more different than was expected, given that the study was designed to maximize parity. According to the pre- and post-tests, the circuit metrics, and the sharing and tinkering metrics, the experience, even given the same software and similar or identical tasks, appeared different, indicating that the different media support different behavior and different comprehension. The theoretical framework of computational and complex systems fluencies will be used highlight the different implications of the media.

Chapter 6: The Virtual and the Physical

Though this study was designed to optimize parity between virtual and physical environments and activities, the different sets of classes performed differently on the assessments, created different types of circuits independently, and differently understood the relationship between the agent and the aggregate. In this section, we will describe the ways in which the virtual and the physical classes differed, give reasons as to why they might have differed, and describe their contrasting patterns of behavior.

Virtuality and physicality engender different behaviors; it is difficult to interact with a physical object as one interacts with a virtual world. As I have hands, feet, and eyes that usually exist in a purely physical world, I am accustomed to use them in a purely physical way. Obviously, we can learn to use computers, but industrial design has not yet reached a point where the keyboard feels as natural as speech. While it is difficult to tease out exactly what minute perceptual and actual differences exist between virtual and physical interactions, it is possible to measure the large-scale differences in effect and action.

This chapter is designed to highlight salient differences in the use of VBOT by virtual and physical students. Case studies, student use logs, and performance evaluations show very different patterns of activity and performance in the two groups. Contrasting the virtual and physical classes highlights the ways in which the two systems supported learning goals differently. Comparing the two classes highlights the common benefits in teaching these fluencies with constructionist robotics. The case studies provide insight into why the two systems support the learning goals differently; the logs provide a backbone of those case studies and provide a portrait of day-to-day patterns; and the performance evaluations highlight the aspects supported and provide some evidence of progress towards learning goals.

6.1 How did virtual and physical students play differently?

The difference between the patterns of activity in a virtual and a physical classroom are immediately apparent. Less than a minute of video would be sufficient for an informed observer to distinguish the classes, even if no physical or virtual robots were immediately visible. In this section, we will examine how students' computational and complex systems fluencies were supported in the classroom, highlighting those differences with particular attention to social aspects of the fluencies, as described in Chapter 3.

The virtual classes are characterized by constant action on the computers punctuated by short discussion. There is a constant low rumble of discussion between students at nearby computers. The class feels traditionally teacher-oriented until the start of those activities in which everybody looks at the common screen in the front of the classroom. At that moment, the class explodes in yelling, movement, and students begin to tinker with their virtual vbots. The first comments heard, usually, are “there I am!” as students locate their individual vbot on the projected, shared screen. At particularly decisive points in the action (near the end, often), students often analyze the actions of their fellow students: “Alice, move your bot! We need to cluster!”

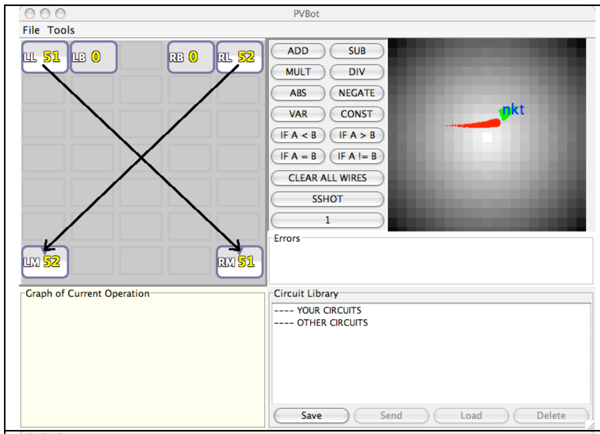
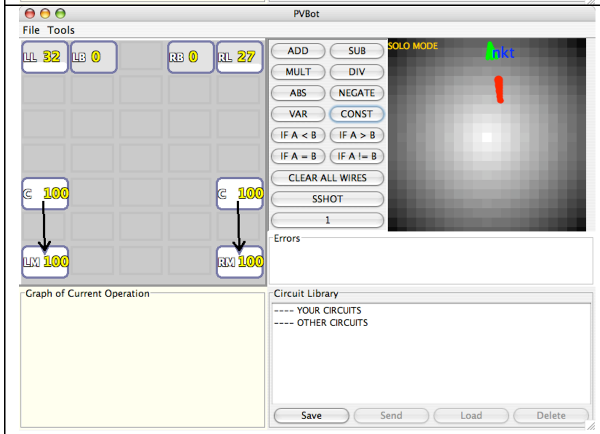
The physical classes feel much less traditional. Mr. Cleveland’s classroom was the most traditional classroom in both organization and teaching method. Despite his attempts to quell noise, however, students walked to other tables to discuss the design of their physical vbot behaviors. These actions result in an environment that feels more like that of an undergraduate engineering laboratory than middle-school classroom. As described in Chapter 4, students work on their circuits for a pre-determined amount of time (“Everybody gets 5 minutes to finish these circuits!”), and each student subsequently downloads her circuit to her robot. The robots are then collected in the center of the classroom and all students simultaneously activate their robots. The beginning of this time for shared space is the quietest moment as people hush each other and wait to observe the performance of their vbots. As the robots start to move, the class gets progressively louder until the climax, at which point students cheer goals or particularly deft moves of their teammates’ vbots.

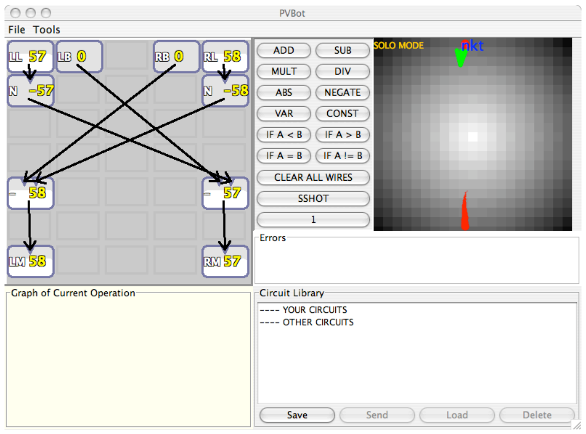
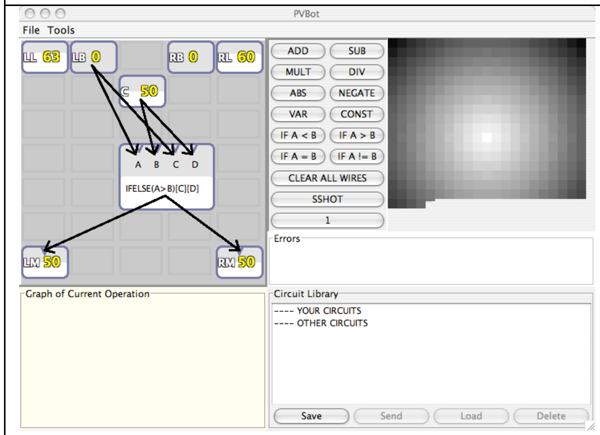
The differences between virtual and physical classes show up clearly on the activity-by-minute graphs (see Figure 6-2 below). Due to their constant work with the computers, the virtual classes produce far more circuits than their physical compatriots. However, the activity in the physical classrooms occurs in many short bursts, whereas the activity in the virtual classrooms resembles a constant roar that peaks around three quarters of the way into class as the culminating activity is reaching its end. The activity then tails off as students fix their circuits for the next day or work on some final, smaller activity.

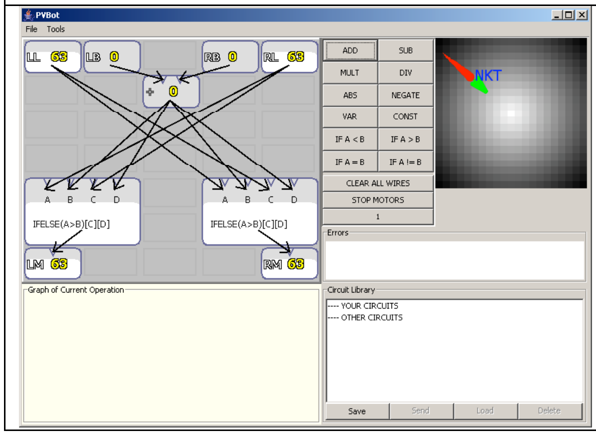
6.2 A circuit progression in virtual and physical Classes at Old Grove

On the final day of the enactments, in both the virtual and physical classes, students competed in a game-activity ("Moon-Tag," described in Chapter 2) designed to test their fluency with VBOT and their real-time programming skills. This section investigates a representative working circuit from the two groups that accumulated the most points in their respective classes during Moon-Tag; one group is from a virtual class and one is from a physical class. By comparing them, we can highlight differences between the virtual and physical circuits that students typically created. These differences highlight the ways that the cognitive (processing) aspects of the fluencies were supported and how the material aspects of the fluencies developed.

Claudia's Group, Physical Class, Old Grove

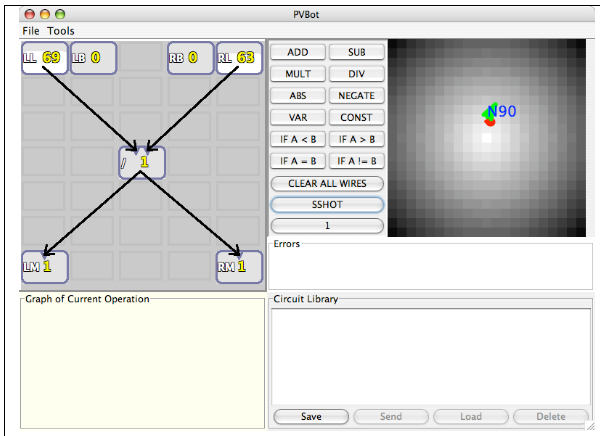
 PVBot File Tools LL 51 LB 0 RB 0 RL 52 LM 52 RM 51 ADD SUB MULT DIV ABS NEGATE VAR CONST IF A < B IF A > B IF A = B IF A != B CLEAR ALL WIRES SSHOT 1 nkt Errors Graph of Current Operation Circuit Library ---- YOUR CIRCUITS ---- OTHER CIRCUITS Save Send Load Delete	Day 1 – This is the circuit with which we start teaching VBOT. It turns toward the light, making a direct path towards the light. The first day of the PVBOT class is spent inventing and investigating this circuit and testing it on the physical robots. Both the virtual and the physical classes start with this circuit.
 PVBot File Tools LL 32 LB 0 RB 0 RL 27 C 100 C 100 LM 100 RM 100 ADD SUB MULT DIV ABS NEGATE VAR CONST IF A < B IF A > B IF A = B IF A != B CLEAR ALL WIRES SSHOT 1 SOLO MODE nkt Errors Graph of Current Operation Circuit Library ---- YOUR CIRCUITS ---- OTHER CIRCUITS Save Send Load Delete	Day 2 – This is a circuit that all students built in order to determine the relative power of their bots. This student's day 2 circuits mostly involved various permutations of constants and light sensors. She also created a circuit identical to the day 2 circuit, substituting bump sensors for light sensors. The circuit shown here is designed to move the VBOT forward at full speed with both motors working 100%. However, due to the vagaries of physical robotics (detailed in Chapter 2), each student's individual robot makes an arc rather than a perfectly straight line, and the arc of the path is contingent on the power of the motors attached to it. The virtual robots do not model this in any way. For the virtual robots, this circuit would drive them directly forward unerringly, and, as such, it holds less interest.

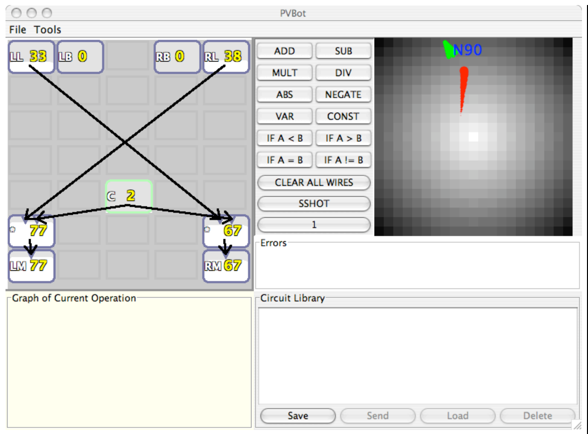
 PVBot File Tools LL 57 LB 0 RB 0 RL 58 N -57 N -58 58 57 LM 58 RM 57 ADD SUB MULT DIV ABS NEGATE VAR CONST IF A < B IF A > B IF A = B IF A != B CLEAR ALL WIRES SSHOT 1 SOLO MODE nkt Errors Graph of Current Operation Circuit Library ---- YOUR CIRCUITS ---- OTHER CIRCUITS Save Send Load Delete	Day 3 – We played our first game activity on day 3. This was a game of "tag," detailed in Chapter 2. Optimally, the robots would swarm the light and knock other robots away. Claudia's circuit did this exceptionally well. This circuit logic is as such: move towards the light until bumped; if bumped, move backwards away from the light. This is an extremely complicated and complex circuit, though the circuit is actually appropriate for the task that this student was attempting. It is indeterminable whether she understood the implications of this circuit. Her later circuits are not as complex, suggesting that she might not have entirely understood the concepts involved in this circuit. This circuit is suited only to physical robotics for two reasons: Virtual robots have no mass and do not "bump." The virtual world is seamless and toroidal, and, as such, virtual robots rarely need to "back up."
 PVBot File Tools LL 63 LB 0 RB 0 RL 60 C 50 A B C D IFELSE(A>B)[C][D] LM 50 RM 50 ADD SUB MULT DIV ABS NEGATE VAR CONST IF A < B IF A > B IF A = B IF A != B CLEAR ALL WIRES SSHOT 1 Errors Graph of Current Operation Circuit Library ---- YOUR CIRCUITS ---- OTHER CIRCUITS Save Send Load Delete	Day 4 – This circuit is notably simpler than the circuit she created on day 3. It moves the robot directly forward at half speed until bumped. When bumped, the robot moves forward at full speed. This circuit helped determine the winner of a contest on day 4. This circuit was designed to push away the other students' robots, and it does so relatively successfully. In stark distinction to the VVBOT activities, this "tackle" design often enabled the team to win by interrupting the opposing team's "scoring" circuits. In many ways, it functions like a

	“pick” in basketball, and Claudia’s teammate Sean mentioned this.
 PVBot File Tools LL 63 LB 0 RB 0 RL 63 + 0 IFELSE(A>B)[C][D] IFELSE(A>B)[C][D] LM 63 RM 63 ADD SUB MULT DIV ABS NEGATE VAR CONST IF A < B IF A > B IF A = B IF A != B CLEAR ALL WIRES STOP MOTORS NKT Errors Graph of Current Operation Circuit Library ---- YOUR CIRCUITS ---- OTHER CIRCUITS Save Send Load Delete	Day 5 – This circuit is more similar to the day 4 circuit than it might appear. It follows the light until bumped, at which point it moves forward at full speed. This “tackler/scorer” circuit effectively won the game for its team by both pushing the ball towards the opposing team’s goal-light and knocking both the ball and the other robots away.

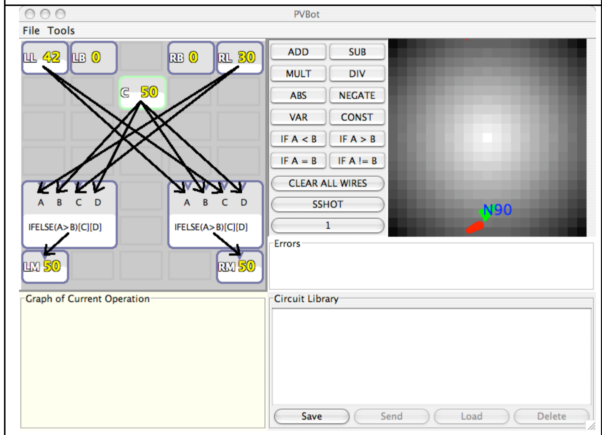
All of Claudia’s group’s later circuits would only have been effective in a physical robotics setting. They all rely on notions of physicality, as they were designed to knock away other robots. This strategy was effective, and it gave Claudia’s group significant confidence. Their robot was repeatedly described as “tough,” and, indeed, they treated it as if it had a personality.

Maribel’s Group, Virtual Class, Old Grove

 PVBot File Tools LL 69 LB 0 RB 0 RL 63 / 1 LM 1 RM 1 ADD SUB MULT DIV ABS NEGATE VAR CONST IF A < B IF A > B IF A = B IF A != B CLEAR ALL WIRES SSHOT N90 Errors Graph of Current Operation Circuit Library Save Send Load Delete	Day 1 – This circuit does not make sense. As it is the first day of the implementation, that is to be expected. This was the culminating circuit in a long chain of similarly incorrect circuits, showing that Maribel’s group did not yet understand the circuit logic. This circuit compiles, although it always moves the vbot directly forward, slowly, under any circumstances. The existence of a circuit wholly unrelated to any we had taught shows that the group was tinkering and testing more than mimicking the teacher’s examples.

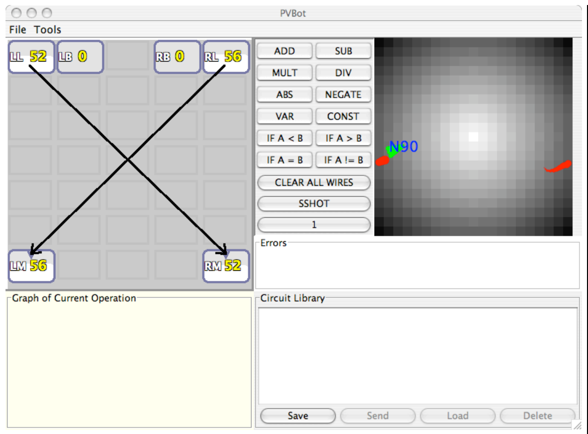
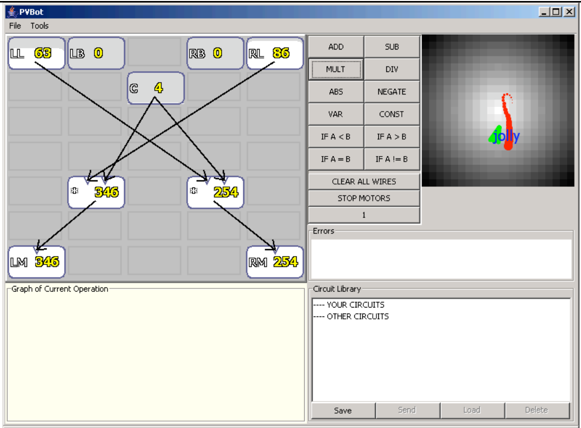
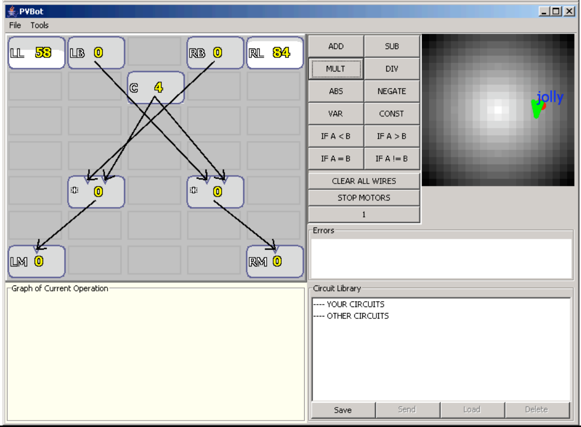


Day 2 –
This circuit was taught on day 2, and this group tested it further. They returned to this circuit several times, using it and tinkering with variations. This circuit causes the vbot to search out the light somewhat faster than the crossed wires circuit described in Chapter 2. Although the group produced several circuits that day, day 2 marked few innovations, and this circuit shows little comprehension due to its conceptual and technical similarity to those circuits exhibited by the facilitators. This circuit works similarly in both the virtual and physical contexts.



Day 3 –
Again, day 3 exhibited circuits relatively similar to the circuits that the facilitators provided, given some tinkering with the value of the constant. This circuit compares the output of light sensors to a constant. As such, the vbot follows the light only when near the light and moves directly forward in all other circumstances. Maribel's modifications were technical and conceptually good, but they show only minor invention.

This is an example of a circuit that would work much better in the virtual context. For this circuit to work with a physical robot, the constant would need to be tuned through repeated trials, which would be both relatively boring and subject to change in different contexts (such as a contest with other robots). The virtual robots can use this circuit reliably in all contexts.

 PVBot File Tools LL 52 LB 0 RB 0 RL 56 LM 56 RM 52 ADD SUB MULT DIV ABS NEGATE VAR CONST IF A < B IF A > B IF A = B IF A != B CLEAR ALL WIRES SSHOT N90 Errors Graph of Current Operation Circuit Library Save Send Load Delete	Day 4 – This is the simple crossed-wires circuit described in Chapter 2. It is notable, however, that this circuit did well in the context of the game activity that the students were playing. Since the goal of the activity was to tag students, Maribel's vbot stayed near the light and tagged all the students that passed by it (tagging automatically occurs when to vbots touch). This circuit works identically in virtual and physical contexts, and it was used by all students in all classes at some point.
 PVBot File Tools LL 63 LB 0 RB 0 RL 86 C 4 346 254 LM 346 RM 254 STOP MOTORS jolly YOUR CIRCUITS OTHER CIRCUITS  PVBot LL 58 LB 0 RB 0 RL 84 C 4 0 0 LM 0 RM 0 jolly	Day 5 – Day 5 provided further evidence that Maribel's group understood her circuits in context. They saved the two circuits shown at left and switched quickly between them depending on the emerging action of their classmates' vbots. The circuits, respectively, moved quickly towards the light and towards other vbots. Few other students used this combination of circuits on the final day, and, although the circuits were uncomplicated, they were effective. Her effectiveness at using the circuits in context suggests an understanding of the relative value and logic of the two circuits. These two circuits would work identically in virtual or physical VBOT, but the ability to switch between them quickly exists only in VVBOT, as PVBOT students would have to download the circuits, which takes time and is effectively impossible during a game.

Comparing circuit progression of Maribel and Claudia's groups

Day	Claudia – Physical VBOT (PVBOT)	Maribel – Virtual VBOT (VVBOT)
One	Make a correct "crossed-wires" circuit to move their bots towards the light.	Evidence of tinkering, but they produce non-functional circuits.
Two	Circuit designed to test physical capabilities of the robot, to tune it and move it directly forward.	Shows tinkering with working circuits provided by facilitators. Little innovation.
Three	Very interesting, innovative circuit designed to navigate the robot and avoid other robots and obstacles.	Modification of another circuit provided by the facilitators.
Four	Simple "bouncer" circuit designed to move forward and disorient other robots.	Using simple circuits in context. The first circuit taught was successfully deployed in an activity.
Five	Variation on day 4 and day 5 circuit was innovative and appropriate. Contextually well deployed.	Used a set of two circuits, switching between them as appropriate. Both circuits are relatively uninteresting, but the novel use of them in context excels in the activity.

Table 6-1: Comparing day-to-day circuit progressions

In this comparison, the VVBOT students tinkered and modified more frequently than the PVBOT students. This was largely expected due to the constraints of the different environments. We also see the PVBOT students making use of non-existent functionality in the virtual setting – that is they are able to use their robots as physical entities with weight, pushing and tackling one another. This shows awareness of the identification of the robot as a physical body in space. The VVBOT students, on the other hand, were more likely to demonstrate awareness of the relationship between the goal of the activity and their circuits. We see this awareness in the above example when Maribel's group created simple circuits to accomplish the project goals.

6.3 How did virtual and physical students perform differently?

The difference in patterns of behavior emerged from the different ways that students built circuits. As described in the last chapter, we evaluated a student-group's circuits on the basis of "uniqueness" (OP_UNIQUE), "difficulty" (OP_DIFF), and "density" (OP_DENSITY). Contrasted on day-by-day data, there existed significant differences between the virtual and the physical classes in the number of unique operations per circuit (OP_UNIQUE, n=108, p<.01), the difficulty of the operations (OP_DIFF, n=108, p<.01), and the density of the operations on the breadboard (OP_DENSITY, n=108, p<.01). Below we can see how they differ.

Means	OP_UNIQ**	OP_DIFF**	OP_DENSITY**
V	1.1181	1.9125	.2751
P	1.6261	2.7737	.4374

Table 6-2: Averaged day-by-day circuit metrics in virtual/physical classes

As shown in the above table, students in the physical classes typically create more unique, difficult, and dense circuits than students in the virtual classes. These three metrics are not necessarily correlated with positive performance, however. As seen in Figure 6-1, OP_UNIQUE and OP_DENSITY decreased each day over the course of the activities in both virtual and physical classes, while OP_DIFF (difficulty) stayed roughly the same across all days. Figure 6-1 shows the graphs of these three metrics plotted against normalized time. In this section, we will investigate several explanations for this.

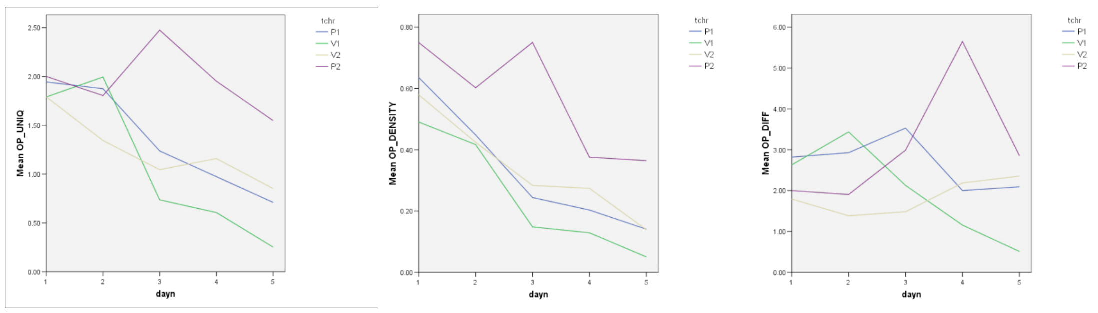


Figure 6-1: Day-by-day graphs of circuit metrics in each class

As seen above, as students understood the system better over the course of the days, their circuits became more targeted and, often, simpler.

It appears that students also came to realize that they could not as easily control excessively dense/unique circuits, due to the number of variables. Thus, as the activities became more fast-paced and intricate, the circuits became simpler. That is not to say that all simple circuits were good circuits, but in Maribel's group above, we can see that optimal placement of a simple circuit did better than the significantly more complex circuits elsewhere.

VP Mean	MULTIPLY	ADD	SUBTRACT	DIVIDE	ABSOLUTE VALUE	CONSTANT **	IF/ THEN *
P	.30	.18	.05	.02	.03	1.14	.29
V	.47	.14	.04	.03	.02	.67	.21

Table 6-3: Mean # of circuit operations per circuit in virtual/physical classes

The physical classes were more likely to use constants and branching logic (IF's). These relatively difficult operations did help them in the post-test performance metric, in which they did significantly better than their virtual class counterparts (post-test question 4.B., n=33, p<0.05). However, the simplicity of the virtual class's circuits ostensibly helped them perform better on the slightly more difficult question 4.C. that addresses complex movement around the shared space (post-test question 4.C., n=33, p<0.05). That is, performance on this complex test question implies that the simplicity of the students' circuits in the virtual classes seems to belie the students' understanding of the complex programmatic concepts they were learning.

VP Mean	TINKERING	SHARING	# of CIRCUITS**
P	.8942	2.74	982.27
V	1.0075	2.73	1200.36

Table 6-4: Action metrics in virtual/physical classes

As we can see, the tinkering metric shows that the virtual class was tinkering slightly more frequently than the physical class, but not significantly more. The sharing metric was remarkably equivalent between classes; given the difference in the atmosphere of the two classes, and the different ways in which they shared, this is particularly notable. Though the distribution is different, the two sets of classes share almost exactly the same amount per circuit. The number of circuits built varied widely; the only surprise was that they did not vary more. Virtual class students spent the entire time at their desks building circuits – the physical classes spent only part of their class time doing so. Nonetheless, the number of circuits built was only about 20% more, on average.

Overall this creates the impression that the virtual class learned concepts physical class did not and vice-versa. These differences are suggested by the profiles of students' circuits, the specific circuits that the groups built, and the differences in the post-test.

The physical class used their understandings of the robot as a physical object as this knowledge as a basis for their primary mode of programming. Claudia's group designed most of its circuits for the specific goal of navigating and bumping other robots. The students' understandings of the capabilities of a robot were mediated by the physical presence of the robot. As such, PVBOT students used more CONSTANT operations and more IF/THEN operations, both of which are more easily adaptable to sensor noise. Sensor noise, as a concept is, most likely, fundamentally new to the students, and understanding this concept enhances both their material and cognitive computational fluency skills, as discussed in Chapter 3. Virtual students, on the other hand, worked more contextually. Students in VVBOT classes built more, simpler circuits that work best in context. As shown, Maribel's group took into consideration the roles that her fellow students played and used those circuits that would be most appropriate. This implies significant conceptual complex systems cognitive and social fluency – understanding the emergent nature of the system and using simple agents to exploit the system rather than acting alone. As such, we can see the ways that the different design decisions with PVBOT and VVBOT, close as they may be, affect the behavior of the students using them.

6.4 Contrasting computational and complex systems fluencies in virtual and physical classes

Virtual and physical students seemed, therefore, to group into distinct patterns reflecting the relationships between computational fluency and complex systems fluency, both between classes and within classes. We call these patterns "cross-levels," after the "mid-levels" work by Levy & Wilensky (2008), in which she describes the cognitive transitions implicit in complex systems fluency. The "cross-level" patterns of the two fluencies can be categorized as such:

	Agent-based <u>complex systems</u> fluency	Aggregate <u>complex systems</u> fluency
Agent-based <u>computational</u> fluency	Unique agent understanding e.g., Designing a single-user computer application	**Multi-agent systems understanding e.g., Designing an agent-based ecology simulation such as Wolf-Sheep Predation (Wilensky, 1997b; Wilensky & Reisman, 2006)**
Aggregate <u>computational</u> fluency	**Team-based systems understanding e.g., Concurrent systems or swarm robotics (Bonabeau, Dorigo, & Theraulaz, 1999)**	Systems dynamics understanding e.g., Designing formula-based ecology simulations using systems such as STELLA (2000)

Bolded cells define "agent-aggregate fluency"

Systems modeling work has often focused on unified mathematical models of a uniform set of actors (e.g., Maani & Cavana, 2000), which we are calling "aggregate-aggregate fluency," meaning “aggregate systems fluency” and “aggregate computational fluency.” Computer science training has traditionally foregrounded serial or single-agent work, which we are calling “agent-agent fluency,” meaning “agent-based systems fluency” and “agent-based computational fluency.” Our hypothesis is that P/VBOT teaches students to focus on the opposing cross-level axis: multi-agent systems understanding and team-based systems understanding, which we are calling “agent-aggregate fluency” (as seen in the table above). The corollary hypothesis is that virtual robotics and physical robotics support agent-aggregate fluency differently. The following sections describe the ways in which the data do and do not support these hypotheses.

These differences are primarily differences in perspective; students approached problem solving differently in the virtual and physical classes. One way to understand these differences is in terms of the relationship of an agent’s action to the aggregate and the relationship of an aggregate’s action to the agent. In the case of a crowd of people leaving a stadium, each agent (person leaving the stadium) is fundamentally constrained by the movement of the aggregate crowd. An individual in the crowd cannot decide to change the direction of the crowd, nor can that individual choose exactly where she moves. She can head towards a gate, but the salient activity is that of the crowd itself. A group of officers directing the crowd, however, has the converse situation: each action of each individual officer determines the movement of the aggregate crowd.

Unique agent understanding is a fluency in the computational and systems aspects of dealing with the actions of a single agent working independently. This can be thought of as the actions of an unconstrained independent individual. A system can consist of an individual agent if the system is internal to the agent. For example, human psychology can be described as a complex system (see Minsky, 1985, for one example). Indeed, attempting to understand the myriad connections internal to an agent is inherent to VBOT programming. A VBOT circuit consists of a set of connected operations from which the semantics of the circuit emerge. Independent operations are only marginally independent as agents, but the connections between them create a complex system of meaning. VBOT, as deployed in both the virtual and physical classes, did not address programming a solitary agent in significant depth, as the majority of both programming and computational fluency work already focuses on programming as an independent activity designed to create single-user programs (Cole, 1996; DiSessa, 2000). Nonetheless, unique agent understanding is essential to computational fluency in the broader sense. The post-test metric privileges unique agent understanding and material computational fluency more than the activities did by asking students to draw circuits on paper in isolation, rather than on a computer contemporaneously with the rest of the classroom. This was our intent, as many computer scientists and physical scientists will recognize and understand the validity of a metric based on unique agent understanding more readily than a systems-based metric due to the nature of the field and historical precedent. Notable here is the contrast between, on one hand, the relatively small disparities on the post-test overall between virtual and physical classes, and, on the other hand, the disparities between their material unique agent skills as exhibited by their behavior on the different problems on the post-test. As seen above, physical classes performed better on question 4.B., which focused on unique agent understanding and material computational fluency, but the classes performed similarly overall. This suggests that the nature of the activity was sufficiently unique-agent-based for all students to be able to perform at the baseline required for the post-test.

Team-based systems understanding is a fluency with the design and manipulation of targeted aggregates. For instance, a baseball coach might be able to direct the flow of a team better than the specific actions of each individual player. A bee-keeper might know how to deal with a swarm of bees but have no idea how an individual bee behaves. A fluid dynamics research needs to know less about the chemical properties of water than the aggregate properties of water flow in certain conditions. From a computational fluency perspective, it manifests as an understanding of ways to manipulate groups of agents toward a specific objective, regardless of the degree of independence exhibited by each agent. Both virtual and physical classes were required to pursue team-based understanding to an extent – the activities were designed to ask students to collaborate towards team-based goals, thereby supporting social computational and complex systems fluencies. However, the virtual and the physical classes did so differently and with contrasting levels of success.

Although the physical curriculum was arguably more team-based (see the description of "Bot Soccer" in Chapter 2), the virtual students worked more cohesively in teams. Maribel's example is one of many in which students would change roles in "Moon Tag" (see Chapter 2) in order to manipulate the swarm or use the swarm to their advantage. This data is supported by the log results that show simpler overall circuits, as simpler circuits often suggested a more contextual approach to circuit programming, as seen above. The post-test results also support this claim, in that the virtual students performed better on Question 4.C., which addressed the movement of an individual vbot in and around a swarm of other vbots.

However, the physical students also performed team-based logic and communication. Although the virtual students' curriculum spent more time devoted to contextual strategies, the physical students developed more formal strategies. In doing so, both the virtual class and the physical class exhibited social aspects of computational and social complex fluencies. Claudia's example details her team's strategies at Bot Soccer, and, indeed, the students in the physical class were more likely to collaborate and communicate individual on planning strategies. This finding is supported by other research with "robot soccer" in which students design explicit strategies (e.g., Sklar, Parsons, & Stone, 2003).

Systems dynamics understanding describes a fluency with aggregate understandings that is notably backgrounded by VBOT activities. In the baseball metaphors above, this corresponds, perhaps, to the designer of a stadium. This fluency describes the design of system-level constraints and flow without necessary consideration of the action of any specific individual. However, there was evidence that students came to understand certain aspects of aggregate systems flow through the programming itself. Pre-/post-test question 3 (flowcharting) showed significant improvement between the pre-test and the post-test in both the virtual and the physical classes. As shown above, the difference was relatively large. The large difference between the pre-test and post-test in all groups suggests that the VBOT program interface and paradigm (common to all groups) is responsible to some extent, and that the students' cognitive complex systems fluency was effectively supported by the system. The VBOT circuit programming method is a likely source of this improvement, as it is structurally analogous to a flowchart (as seen in Chapter 2).

Multi-agent systems understanding is a fluency related to the methods of designing individual agents to perform some emergent task. This corresponds most closely to the individual in the crowd; each individual has agency, but the salient understanding comes from those aspects that emerge from the system, not from the specific movement of any agent. Although this is closely related to team-based systems understanding, the salient difference is the focus on the behavior of the agent over the behavior of the aggregate. NetLogo (Wilensky, 1999), on which VBOT is based (see Chapter 2), is a tool designed to foster complex systems understanding through the use of agent-based systems. The relative value of team-based understanding and agent-based understanding towards high-level complex systems fluency is still under debate (see Levy & Wilensky, 2008). A canonical method of agent-based systems understanding is to frame more traditionally aggregate systems problems in terms of their component agents, such as simulating traffic flow in terms of the behavior of cars rather than aggregate traffic flow. This approach has been usefully employed by a variety of academic disciplines, ranging from physicists (Bar-Yam,

1997) to economists (Axtell & Epstein, 1996).

Our results here appear surprising. Although virtual systems have been successfully used to teach multi-agent systems, our data suggest that the physical students tended to understand the systems from a more agent-based perspective. The virtual students, on the other hand, tended to understand the systems from a more aggregate perspective. Evidence for this finding stems from log data, video data, interaction patterns, and pre-test/post-test differenced.

The circuit-based log data explained above show that physical students created more complex and more difficult circuits overall, though the virtual students created more circuits. This implies that the physical students attempted to design circuit behaviors in which the robot would move and act independently. The virtual students, on the other hand, created circuits to be contextually relevant and often created these circuits quickly to respond to the actions of the other vbots.

This evidence is corroborated by the post-test results, which show that the physical students performed significantly better on Question 4.B. (about directing agent behavior), while the virtual students performed better on Question 4.C. (about the behavior of a vbot in the context of other vbots). As stated earlier, both classes performed relatively similarly overall on Question 4 (programming vbots). These differences suggest that while both sets of classes were building material fluencies, they did so differently.

The contrast between Maribel's and Claudia's experiences highlights the differences between their multi-agent systems understanding. Maribel's group worked with fellow groups as a member of a swarm, determining appropriate behaviors contextually. As a contrast, Claudia's group worked to create an independent agent that worked on a team (as a "tackler"). These strategies underscore the ways in which the students' social computational and complex systems fluencies related to their overall performance and perspectives.

Some of this difference is likely attributable to the different patterns of behavior in virtual and physical classes. Figure 6-2 below shows all student board changes, minute to minute, in one school on day 5 in that school's virtual and physical classes.

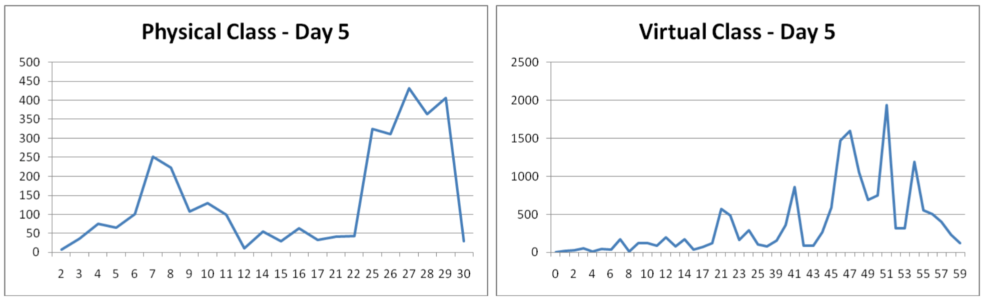


Figure 6-2: Board changes per minute at Bayville, Day 5

The physical class is bi-modal, and the virtual class is multi-modal; note that the physical class ends after 30 minutes, and the rest of the time was spent testing their creations together in various contests. In the physical class, students would spend chunks of time working on their robots, and then test them together *en masse*. Note that this behavior was not required by the teacher or the setting, but rather, determined by the students; indeed, the teacher in these two classes was the same teacher (see Chapter 4 for more details). The virtual class was constantly and consistently changing their circuits. By working on their circuits individually and testing them *en masse*, the physical students had more opportunity to hone the individual behaviors of their robots. In contrast, the virtual students more often honed the contextual behavior of their vbots.

Using cross-level fluencies to characterize the virtual and physical classes underscores the ways in which the two classes exhibit different aspects of computational and complex systems fluencies differently. While both virtual students and physical students exhibited significant improvements in agent-aggregate fluency, the virtual classes did so from an agent-based perspective, while the physical classes did so from an aggregate perspective. The tables below summarize this difference and the ways that the fluencies affected virtual and physical classes differently:

	Physical	**Virtual**
Cognitive	Enables aggregate thinking from an agent-based perspective	Enables agent-based thinking from an aggregate perspective
Social	Must communicate with group to work out contextual plans	Motivates communication about emergent patterns
Material	Programming an agent in a shared space	Programming an agent in an aggregate

Virtual/Physical Class Differences in Complex Systems Fluency

	Physical	**Virtual**
Cognitive	Enables aggregate logic from an agent-based perspective	Enables agent logic from an aggregate perspective
Social	Motivates communication about strategies	Communication is more transparent and intrinsic to activity
Material	Enables difficult circuit-building skills	Enables contextual circuit-building skills

Virtual/Physical Class Differences in Computational Fluency

This chapter highlighted the ways that virtual and physical students exhibited different aspects of computational and complex systems fluencies. The differences emerged from both design decisions and inherent features of the virtual and physical media. The next chapter explains how specific design decisions affected student behavior and performance, and how well the project achieved stated design goals.

Chapter 7: Design Principles

In this chapter, we evaluate our work on the basis of proposed design principles and whether the design goals and principles were met, given the results of the study. As discussed elsewhere (Chapter 2), the design of VBOT stems from studies performed with NetLogo (Wilensky, 1999), Flogo (Hancock, 2003), and Logo (Papert, 1980). These studies provide us with both baseline design principles and contrasting material. The contrast illuminates the benefits and deficits of both VBOT and the study design. Thus, we use this contrast and analysis to evaluate interface design, language design, and relative learning gains in VBOT.

7.1 Motivating complex systems and computational fluencies

Chapter 3 evaluated sharing and tinkering as organizing principles for understanding complex systems and computational fluencies. In this section, we describe the elements of the interface designed to motivate complex systems and computational fluencies, and evaluate the design elements for effectiveness. Based on Chapter 5's focus on the relationship of sharing and tinkering to performance, this chapter will address the relationship of design to sharing and tinkering. The questions to be addressed, then, are:

- In what ways did we intend the design to motivate sharing and tinkering?
- How well were sharing and tinkering motivated in the enactments?

- What design goals other than sharing and tinkering affected complex systems and computational fluencies?

Hancock (2003), Braitenberg (1984), and DiSessa (2000) make the argument that real-time programming motivates both sharing and tinkering. This study is built on the belief that sharing and tinkering arise naturally from a real-time game-like programming environment. This can be seen from a different perspective on the recent serious games literature. For example, Gee (2004) and Squire (2004) both argue that people involved in networked games not only engage significantly with them as learning tools, but learn how argue and communicate about the games themselves. In my study addressing complex systems fluency, communication and engagement about complex systems material is essential for teaching social aspects of complex systems fluency. Squire (2004) suggests that it is in the nature of a "good game" that people want to play with it and communicate about it. Given the video-gaming literature, we hold that the degree to which students tinkered and shared while working with VBOT is directly tied to the real-time aspect of the system.

In addition to learning sharing and tinkering methods, students learned how to program in the VBOT language. By the final day of the enactment, every group in every class could create programs that showed understanding of the target objectives: every group played the game. Moreover, 80.5% of the students created at least half of the circuits correctly on the post-test evaluation.

Complex systems are systems in which meaning emerges from networks of independent agents. This concept is inherent to the design of the breadboard; a circuit is a system in which semantic meaning emerges from a network of independent operations. As they were building circuits, the students were simultaneously learning how to create complex networks of meaning to achieve emergent goals. In other words, each student created networks of VBOT operations out of which emerged a behavior for their vbot. Though several studies (Chi, 2005; Hmelo-Silver & Pfeffer, 2004) show that students have trouble correctly intuiting the emergent meaning of a network, this study suggests that, provided that a network is comprehensible, the students can form robust understandings of the emergent network. This understanding is borne out by the pre-test to post-test performance on question 2 (flowcharts), as discussed in Chapter 4. Students, on average, did 19.0% better interpreting the static flowchart.

The complex system in VBOT is both at the circuit design level and at the agent collaboration and competition level. The notion of "levels thinking" comes from Wilensky & Resnick (1999). He argues that robust complex systems understanding must come from an understanding of the different "grain sizes" of the phenomenon and their inter-relationship. Out of the aggregations of agents in the VBOT shared space emerges a level of complexity above that of the circuit design, much like the flock in pre-test question 2 emerges from aggregations of individual birds. This complexity of the vbot "flock" is harder to predict and understand than that of the circuits themselves. A shared virtual VBOT screen often shows more than ten individual agents, and a shared physical VBOT space (e.g., the floor) has multiple physical robots interacting in constant motion. Students' ability to understand the network of the relationships of the agents shows up in the differences between their pre-test and post-test answers about agent-aggregate interaction. As described in Chapter 4, question 1 is designed to test student understanding about the flocking of individual birds. Students in the post-test change their answers about the leadership of the birds only slightly (6.7% overall), but students are more likely to give answers about the nature of individual bird understanding.

In our study, students used complex systems and computer science fluencies to effectively use VBOT and to perform on the post-tests. We view this as a success. Students using physical and virtual VBOT understood and could communicate their understandings differently. Therefore, the design of the systems accounts for some of the differences in understanding. The sensitivity of the learning gains to the difference in design is itself a notable result.

7.2 Evaluating Design Principles

Supporting programming in equivalent virtual and physical spaces

The system is built both as a virtual and physical environment, and students did not have a choice as to whether they used virtual VBOT or physical VBOT. Virtual and physical spaces were supported differently, attempting, but not necessarily attaining, equivalence. Although VBOT and the supporting activities were designed to optimize equivalence, making all of the activities identical would have been to the detriment of both the virtual and physical environments.

The virtual space was built first and tested more extensively in the pilot studies because it was easier to implement. As a programmer, building a virtual system came more naturally to me. Furthermore, I was initially more familiar with literature on virtual collaborative spaces than physical ones, due to my work in the Center for Connected Learning and Computer-Based Modeling (such as Berland, 2006). However, physical educational robotics has been studied extensively, as detailed in part in Chapter 2. Many of these studies show that students were engaged and motivated, and that the systems supported computational fluency. Conversely, virtual robotics environments have not been tested as extensively, as described in Chapter 3. Furthermore, all of the researchers had access to computers but no access to a bank of physical robots. These factors conspired to make virtual VBOT the predecessor of physical VBOT. This precedence might imply some favor towards virtual VBOT.

However, physical VBOT had the benefit of significant outside research and study design. LEGO Mindstorms robots (2002) are relatively popular commercial toys, and they are used around the world to teach about computers and programming. We used several well-tested software/hardware packages to make the programming and design of physical VBOT easier to implement. Furthermore, the study design for both virtual and physical VBOT was adapted from study designs by Wilensky (1999) and other constructionist physical robotics studies (e.g., Martin, 1996b). In that sense, the physical VBOT projects were more grounded than the virtual projects.

The data show little overall difference in engagement or overall ability. As described in Chapter 6, virtual VBOT students did better on some questions, while physical VBOT students did better on others. For the reasons described in Chapter 2, virtual VBOT students produced far more circuits as they were constantly tinkering. As tinkering was correlated to performance, this would seem to favor them. However, the physical students did just as much tinkering in the time that they were working with the physical VBOT. Although they often had to download the circuits and wait, they would repeatedly consider and tinker with circuits before downloading them to their VBOT. Often they had to work faster because they had less time to complete their tasks, given the amount of time dedicated to physically moving and using the robots.

If the activities were made absolutely equivalent by equalizing group size and activity structure, it would have had the immediate effect of making physical VBOT much harder and more expensive, while making virtual VBOT activities less engaging. For example, a requirement of commercial physical robotics is the time taken to download programs to the physical robot. If we paused the virtual robots to simulate a “download” before acting, it would seem both artificial and boring. Furthermore, physical VBOT requires more people per group to handle the physical and virtual aspects of the activity at any given time. While one student handles the downloading, another student can set up the physical space for testing. In his physical robotics studies, Hancock (2003) uses groups of around 5 middle-school students. Similarly, Martin (1996b) used groups of three or more students. Smaller student groups would have made the project difficult to finish suitably in the allotted time. Similarly, raising the group size of the virtual VBOT studies presents a problem, as one keyboard and monitor can only address a limited number of people; four people per keyboard can interfere with cooperative behavior.

Physical and virtual simulations differ in real-world settings, as sensors and motors work very differently in the two environments. Maes (1991) discusses the reasons for the inherent difference, but they are mostly outside the scope of this project. For this project, the difference can be reduced to the so-called "noise problem." That is, physical sensors are "noisy" – they do not produce reliable information. The positive aspect to "noise" is that programmers come to understand feedback and logic much more reliably, making simpler, more targeted circuits (Brandes & Wilensky, 1991; Papert, 1996; Wiener, 1948). Resnick & Ocko (1991) suggested that many of the learning gains from educational robotics stem in part from coping with this noise. An obviously negative aspect to noise is that it can make building robust circuits intractably hard for middle-school students. Noise alone creates a need for different types of activities to address sensors. As seen in Chapter 6, physical VBOT students often created more robust circuits, while virtual students simply created more circuits.

This design goal of relative equivalence was met – it is inherent to the project, but the necessity of equivalence between physical and virtual systems is debatable. The equivalence could be located in high-level program design (i.e., do students use the same program throughout?); low-level program design (i.e., is the only difference between virtual and physical VBOT the actual physical robots?); or activity design (i.e., do student have the same day-to-day intermediate goals?). If the goal is equivalence in high-level design, then the project was an unqualified success. If the goal is equivalence in low-level design, the project was, perhaps, a qualified success; the students used much of the same mechanical knowledge, but the activities were technologically different. If the goal is equivalence in activity design, the success is tempered by the differences between the virtual and physical technologies. One result of this study is that virtual/physical equivalence is not strictly possible at the moment in a real-world school setting with a limited budget. Another result might be that these (forced) differences are illuminating.

Supporting “low threshold, high ceiling” computation

Papert (1980) and Wilensky (1999) explicitly advocate a “low threshold, high ceiling” (LTHC) philosophy; an environment must be easy to enter as a novice (“low threshold”), but deep enough for expert users (“high ceiling”). Flogo (Hancock, 2003) is designed to be LTHC, as novice users can use the graphical interface, segueing eventually to the high ceiling text-based interface to create more complex programs.

None of the students in our settings had had any relevant programming experience. We supported these students by designing a program with an extremely low threshold, potentially lower than either of the thresholds in NetLogo and Flogo. This low threshold came at the expense of a high ceiling, specified by the LTHC philosophy. This means that every student was able to successfully interact with the system but it does not easily support the more complex programming tasks (as discussed above). The system was designed to respond to every input with action, allow only limited input, not to allow explicit errors, and to make all action in the system explicitly lead to a reaction in behavior. Our design decisions served to limit the ceiling of the project. For instance, by making all possible actions immediately visible and labeled, the number of them had to be limited by computer screen space. Flogo avoids this problem by allowing the user to create new functions and describe them textually. Though VBOT allows some flexibility of operations in each activity, the contextual operations are pre-defined for all users of the activity, rather than individually as in Flogo. For instance, in VBOT a teacher could decide that a new type of sensor is necessary. All students in the class would then be able to use the new type of sensor. An individual student cannot create a new sensor independently.

The success that students experienced when working with VBOT (discussed in Chapter 5) implies that our design successfully created a low threshold. The decision to lower the ceiling grew out of the purpose of this program. While both Flogo and NetLogo are used in non-research settings, VBOT was designed for this purpose. This success came at the expense of a higher ceiling. This is not a novel finding, but it is an important one to reiterate. Perhaps a better designer could have achieved such a low threshold without lowering the ceiling, but, as described here and in Chapter 6, the current set of programs designed at this target threshold do not.

Designing a low barrier to entry

Scalability is extremely difficult; it is one of the most obvious failings of many otherwise good learning environments. VBOT has not been tested at the large-scale implementation level. However, whether or not VBOT is scalable, it is possible to deploy in public schools with relatively little professional development. For virtual VBOT, nothing need be purchased. This is of absolute import as the poorest schools are in greater need of this technology. Many schools now feature computers, but they are used for little more than email and web surfing (Cuban, 1995). Perhaps by using more constructionist activities such as those in VBOT in these schools would improve the quality of science education.

Physical VBOT requires the purchase of LEGO Mindstorms Robots. LEGO Mindstorms Robotics Systems are already featured in many schools (LEGO, 2002), and the systems are relatively inexpensive (a classroom can be outfitted for about the price of one computer). A significant benefit using LEGO Mindstorms is the installed user base and many educational users. Indeed, while none of the teachers had taught classes using LEGO Mindstorms, they had all seen them before.

Using groups of students makes it easier to deploy in crowded classrooms with limited resources (Cuban, 1995). Although every class initially claimed to have sufficient resources to accommodate each student on an individual computer, in every classroom, some of the computers that were provided simply did not work. Hundreds of students using a limited bank of computers every day results in broken computers. In one case, we had to borrow computers from another computer lab, as more than half of the computers in the classroom were non-functional.

Thus, with caveats around professional development, we could claim that this program is scalable, at least financially. However, questions about scaling the pedagogy remain difficult to address, due to the size and scope of the studies performed. Another large-scale study is, at least, possible to achieve with the current technology. That would be necessary to evaluate the possibility of large-scale public school deployment. To be deployable with minimal professional development and out-of-pocket cost has long been an important objective for learning environments (see Cuban, 1986, for historical examples). The scalability problem is compounded by the different resources necessary for design and deployment. Nonetheless, it remains important to address it in the design, as teachers often ask about scalability issues in enactments.

Designing “game-like” programming

Real-time programming was considered a priority because of its relationship to engagement and complexity. Gee (2004) and Hancock (2003) both show how real-time activity, or “liveness”, enhances a user’s feeling of engagement. Wilensky & Stroup (1999a) show how the simulation of complex systems in real-time enhances students’ ability to comprehend the systems.

Virtual VBOT is more specifically “real-time” than both NetLogo and Flogo. That is, all actions have immediate effects in the shared space, which does not “pause” except at the end of activities. Physical VBOT is not “real-time” in the same sense; students build circuits but must download those circuits to their robots. Virtual VBOT is more similar to online video games than other programming environments. In games such as World of Warcraft, every action has an effect in a shared game world that is immediately seen by all other players in the spatial vicinity. In fact, a few students made informal references to these games when using VBOT.

Virtual VBOT was effectively real-time, but physical VBOT was not. Although students using physical VBOT could lightly tinker with circuits before downloading them, the action was not effectively real-time.   There are physical robotics systems, such as the Cricket system (Hancock, 2001), that support real-time active physical robotics, but they are more expensive, and they cannot be purchased from commercial retailers. An original design for physical VBOT used the real-time robotics, but that plan was abandoned due to considerations about scalability, cost, and expertise.

Motivating programming as an aspect of both computational fluency and complex systems fluency

Programming is, at its core, building; to learn by doing is not the same thing as learning to do. These two points are sometimes conflated in constructivist literature, but in this study they are not the same thing. One can learn to program without programming: a former colleague of mine had been trained to program at a university that did not have a computer lab. One can also learn to program by implementing a predetermined and prewritten set of programs. My personal experience leads me to believe that those approaches tend to be both dull and ineffective. Learning to program creatively by programming creatively is often the best option – this is a tenet of constructionism (Papert, 1980).

But just as thinking about thinking is thinking about thinking about something (Husserl, 1913), learning to program is learning to program about something. Every programming language carries within it a metaphor for understanding computers. Boxer (DiSessa, 1997), Logo (Papert, 1980), NetLogo (Wilensky, 1999), and, indeed, Prolog (1972) all show us that when one is programming, one is thinking and learning in that language specifically. Non-programmers often think of programming as inherently “mathematical” or “logical”, but there is a vast distance between considering programs as interactions between agents (NetLogo) and existence predicates (Prolog). In the same way, programming in VBOT is considering networks of emergent meaning. This is emphasized by the content of the activities, which are also about networks of emergent meaning and emergent paths to local or global goals.

Therefore, the question becomes: how well does the syntax of the language enable complex systems and computational fluency? VBOT does not approach NetLogo in motivating complex systems fluency, as would be expected. Standard NetLogo syntax addresses group of agents, and nearly all commands are organized to enable users to design emergent systems from groups of agents. VBOT, even at the level of basic syntax, is design in this way. VBOT syntax itself, is, as described in this Chapter and in Chapter 2, an emergent network. However, a VBOT user could theoretically program VBOT independently. The activity structure of VBOT, however, requires groups of students collaborating and competing to fulfill goals. Similarly, VBOT cannot approach Boxer in motivating computational fluency. Boxer provides a model metaphor for understanding computation. VBOT provides only the simplest circuit metaphor for computation. However, inherent to both systems is a novel syntax designed around targeted programming. Towards both fluencies, the simplicity of VBOT is both its strength and weakness.

Motivating students toward intrinsic goals

This goal mediates the other goals. If I am a student, why should I use this system? "Teacher said so" is a reliably problematic answer. It is a fundamental tenet of constructivism and constructionism that people learn based on what they know (Piaget, 1972); people learn by mentally participating. A student who hates what she is doing will do little. These arguments have been verified many times in many ways, from Dewey (1913) to Pintrich & Schunk (1996). The question remains: how do we make learning environments and activities enjoyable and usable? Gee (2004) provides several concepts on what makes a game usable and engaging.

Simplicity is fun. During the pilot studies, it became clear that we did not need so many buttons, so many features, or so much clutter for this program. No student built an operating system in the last half-hour of class, and we did not need to prepare a system in which that was possible. By paring down the interface repeatedly, all elements existed on the same page at the same time. This greatly enhanced both usability and comprehension. Lowering the threshold meant that students could program immediately without hours of explicit training. Students reported that they excelled at building circuits, and hence material computational and complex systems fluencies, when they did not feel intimidated by the language or interface.

Complexity is fun. Why read a book if you already know an ending? The activities were made increasingly complex to satisfy the students. When I first presented the final activity ("Moon-Tag") at a research presentation, many colleagues argued that it was too complex and too involved for middle school students. One teacher argued that they would leave frustrated and dejected after failing miserably. The opposite occurred. Students in every class could describe the game to their fellow students within a matter of seconds; this provides some encouraging evidence of students building social fluencies.

Action is fun. It goes without saying that active play is fun. Sports are fun for some; video games are fun for some; and painting is fun for some. All of those involve some form of active play. It seems odd, then, that traditional school often includes so little active play. This is something that we know that students enjoy and independently engage in, so we should exploit it as much as possible. In Physical VBOT, the students had to stand up and walk to a central space for their robots to compete.

Understanding is fun. A significant percentage of negative questionnaire comments concerned the incomprehension or misunderstanding of an operation or circuit. Students like to understand; building understanding is building cognitive fluencies. We have theorized (Berland, 2006) that one reason academically poor students do well in HubNet and VBOT is that they feel like they understand concepts at the class level – they are not "left behind." Similarly, students liked the circuits and games that they felt that they understood well. Simplicity and understanding are synergistic; the simplicity of VBOT enabled students to understand the robots that they controlled.

Having is fun. Possession of a vbot, whether physical or virtual, actively engages a student in learning. Even the most sullen student identifies with a vbot bearing his or her name on the screen or on the floor. Wilensky & Stroup (1999a) noted that in a HubNet activity, students typically yell out potential actions for individual agents by name: "Bobby, move your vbot!" The same effect in VBOT motivated the students to take some interest in learning. Few of the middle school students wanted to be the only student left out of an activity, and some of the activities required all students to participate in order to reach an emergent goal. In those cases, the students were almost forced to have fun, but the forcing came from their peers, rather than from a teacher. It is rather novel to hear an 8th grade student remark that her compatriot needs to "learn to program!" These statements are concrete indications of the ways that social computational and complex systems fluencies are being supported by the system.

Being is fun. Papert (1980), Hancock (2003), and others argue that embodiment and body syntonicity are essential for good constructionist design. They argue that more concrete activities and metaphors are easier to understand. Inhabiting a space brings with it implications for engagement and understanding that remain relatively unexplored (see Abrahamson, 2006), but implies that students are building complex systems and computational cognitive fluencies. It was rare to see a student trying to understand the movement of a vbot without some body contortion simulating the movement of the individual's vbot (Berland, 2005). Gesture and movement are powerful comprehension tools (Goldin-Meadow, 2003), and using them gives us access to an underused learning tool for the modern public school.

7.3 Design Conclusions

How well did VBOT design decisions address our design goals overall, how well did VBOT design decisions address our learning goals, and what principles can we draw from our findings?

By design, a single student using VBOT cannot write a single circuit that satisfies an activity's global goal; all activities are designed for multiple actors, and the software is designed to enable this. Students wrote simpler programs than those found in the work of Hancock (2003) or Wilensky (1999). Hancock (2003) asks middle-school students to program robots to explicitly search out and return specific objects to a predetermined location. Towards this end, Flogo provides a separate, text-based interface to input more complex program code, providing a wholly different program metaphor than the circuit-based interface. VBOT provides no such facility, and it is impossible in VBOT to "delve" into the text-based program code for any reason. Thus, while tasks such as the Flogo task are feasible in VBOT, they are difficult, and no task of that complexity could have been undertaken in the classroom time allotted. In this sense, the design of VBOT was starkly different from Flogo: our design foregrounds tinkering, sharing, and "liveness" (from Hancock, 2003) over more complicated programming. In this way, the design of VBOT promotes complex system fluency through computational fluency.

Wilensky & Stroup (1999a) discuss the benefits and deficits of using HubNet system in the classroom. As described in Chapter 2, VBOT is a descendent of HubNet. HubNet, contrasting with VBOT, requires no programming. HubNet is designed to support complex systems fluency by allowing students to act as an agent in a participatory simulation. Compared to other HubNet simulations, VBOT foregrounds circuit building and programming over direct agent action as a way for students to interact in a shared social space. In this way, the design of VBOT promotes material and cognitive computational fluency through material and cognitive complex systems fluency.

The data suggest that the computational and complex systems fluencies were mutually reinforcing, and that the boundaries set on program complexity and agent interaction were contextually fruitful. Student programming in this study did not approach the maximum complexity of VBOT programs. Indeed, as seen in Figure 6-1, the relative difficulty of the individual operations decreased by the last day of the activity. As the activities became more complex, students created more targeted, simple circuits to achieve their goals. Hancock's study guides the students into an eventual goal of being able to program in traditional functional languages, such as LISP; this study had no such goal. Hancock's study was designed differently and targeted differently, using volunteers with previous programming experience in after-school programs. Our data suggest that even previously untrained middle-school students are capable of creating relatively complex correct programs. 80.5% of the students created at least half of the circuits correctly on the post-test evaluation. Students in the top 50% of post-test scores did not have a higher average difficulty per circuit metric (OP_DIFF). This implies that material computational fluency was not directly tied to building complex circuits. That is, the simplicity of the circuit design appears to have been borne out of the activity design rather than student abilities.

Chapter 8: Conclusions and Future Directions

VBOT is a fully integrated virtual/physical constructionist robotics-based learning environment designed to teach complex systems and computational fluency to middle school students. Four 8th grade public school classes (2 virtual, 2 physical) at two schools used VBOT and showed significant learning gains towards both fluencies. Furthermore, we examined several possible explanations for those gains and investigated how a set of constructionist design principles related to those gains. The most salient factors in those gains were tinkering and sharing, and while students in both contexts demonstrated learning gains there were interesting differences across them.

8.1 What is VBOT?

The VBOT language was shown to be a robust circuit-based robotics language. In VBOT, groups of students compete and collaborate to build several virtual and/or physical robots. There are a few salient differences between VBOT and other educational robotics packages: it is designed to work equivalently for both virtual and physical robotics; it is inherently collaborative and multi-user; behavior emerges from networks of connected operations; all virtual robotics is real-time; and, as an architecture, it emphasizes tinkering and sharing.

VBOT is based on the HubNet participatory simulation environment (Wilensky & Stroup, 1999b). In the virtual aspect of VBOT (called "virtual VBOT" or VVBOT), as in HubNet, students (or pairs of students) each control a single avatar through a game-like interface. This interface controls the actions of a single "vbot" avatar. The avatars of all of the students act in a common, shared screen-space. The students look at this shared screen in the front of their classroom to see their actions as well as the actions of their fellow students' avatars. The physical aspect of VBOT ("physical VBOT" or PVBOT) differs in that the students use their interface to control an actual robot, which is a physical manifestation of the virtual avatar described above. To control their robots, students must download their programs to the physical robots. These robots (also called vbots) then act in a shared space, which is usually a ring marked on the floor of the classroom. The physical robot used was the LEGO Mindstorms robot, described in Chapter 2 and illustrated in Appendix 1.

8.2 Why was this study necessary?

This work rests on several fundamental assertions: that computational fluency is an important goal; that complex systems fluency is an important goal; that the two fluencies are mutually reinforcing; and that a constructionist robotics learning environment provides a fruitful path toward that goal. The study with VBOT derives from a host of companion works. Papert (1980) described constructionism as a method for teaching and learning. This work is, if nothing else, an argument for increased use of constructionist methods in the classroom. Wilensky (2001) pioneered work on complex systems fluency. DiSessa (2000) pioneered work on computational fluency. Hancock (2003) and Resnick & Ocko (1991) provided works on robotics in the classroom that fundamentally informs the paths that this work follows.

Chapter 3 explored the roots and justifications for complex systems fluency and computational fluency. The call for complex systems fluency stems from the finding that American students lack basic scientific fluency upon exiting high school (AAAS, 2007; TIMSS, 2003) and complex systems fluency is posited as a novel, comprehensible mode for understanding scientific phenomena. Teaching students about the elements of phenomena and the relationships between those elements fosters understanding of some phenomena better than traditional equation-based science (Wilensky & Stroup, 1999a). The need for computational fluency stems from the growing role that computer technology is playing in the US. While more students may use computers on a daily basis, there is little evidence that students better understand computation (DiSessa, 2000).

Wilensky & Resnick (1999), Wolfram (2002), and others have long posited a strong connection between computational fluency and complex systems fluency. Their studies suggest that complex systems theory is more easily available as a mode for understanding science because of the increasing prevalence of scientific computation and visualization. However, there are few studies that test learning environments for their applicability to complex systems fluency or examine the relationship between computational fluency and complex systems fluency at the middle-school level. Constructionist virtual and physical robotics provide a solid context for teaching these fluencies because both have been individually tested before with good results, as described in Chapter 2. However, there are few comparisons of constructionist virtual and physical robotics with modern technology, however.

This work, in turn, fills several gaps in available research because it is: a comparison between virtual and physical constructionist environments; an investigation into the relationship between computational and complex systems fluencies; and a test of the effectiveness of different learning environments toward teaching those fluencies.

8.3 Why did we use these methods?

As such, the study design is a traditional “2 by 2” factorial research study (as described in Ericsson & Simon, 1984), using a VVBOT class and a PVBOT class each at two separate school, totaling 4 classes. By comparing and contrasting schools, virtual classes, and physical classes, we were able to more clearly see the differences that stem from teacher, student, class, and learning environment interaction. The “2 by 2” study allows researchers to factor out specific and individual differences on two axes. In a 2x2 study, the between-class and between-treatment differences often contrast if they exist.

Each of the four classes was an 8th grade, non-magnet public school classroom in a large Midwestern city, and each used P/VBOT across roughly 5 days for an hour per day. During the first three days, time was spent teaching students how to use VBOT operations. The majority of each class days was spent in VBOT, with the exception of the time spent for rudimentary instruction, discussion, and questionnaires. All mouse or keyboard activity in VBOT was logged to a central server. We also collected videotape data of each class day on two cameras. In addition, we administered a pre- and post-test for all students, daily questionnaires for all students, and pre- and post-interviews with 4 students per classroom.

8.4 Did we meet our goals?

There are three questions the reader should be able to answer at this point:

1. What are the benefits and deficits of using physical or virtual robotics for learning computer science or complex system concepts?
2. What aspects of design motivate this learning?
3. How well does this work?

This work succeeds to the degree that one can answer those questions from its reading. Fundamentally, this work is a study of one system (VBOT) used to teach two intertwined learning goals (complex systems and computational fluencies) to a specific population (middle-school students). This study was situated in four middle school classrooms from two different

Chicago public schools over the course of one week. From this perspective, this is an enormous task – these students have no programming experience and many of them are doing quite poorly in traditional academic settings. To teach these students about computer programming and complex systems content simultaneously might seem too big, too fast, and too much. However, students learned not only how to program in a proprietary system, but they did so in a complex systems environment, in a way that shows remarkable understanding and improvement. Our results led us to conclude that complex systems and computer science fluencies are mutually reinforcing.

8.5 What did we find?

In short, we found that students in all classes learned as a result of being able to play relatively freely with the system. The amount of time given by the individual teachers, the design of the class material, and the school environments mattered far less than simply motivating students to share and tinker with the system. Perhaps against common perception, boys did not perform better than girls, higher SES did not perform better than lower SES, and the virtual classes did not perform better than the physical classes. What we found was that students learned a great deal across the board: a significant majority of students learned how to program in VBOT, they could collaborate and compete in the classroom using VBOT, and they could take that knowledge and apply it on the post-test evaluations.

Complex systems and computational fluencies are mutually reinforcing.

We have noted the relative value of complex systems and computational fluencies several times over the course of the work to gain lenses with which to understand science, math, and computers. Many of these students entered the project with a mystifying view of robots as Martian invaders and left the project believing that they knew how to program robots. Other students came in assuming that every flock of birds had a pre-determined leader, and that every anthill was "radio-controlled" by a hidden queen. These same students came to understand that sometimes the only way to a common goal was an array of nearly identical simple agents. All of the students became familiar with the concept of networks of meaning, both in computation and in society.

Virtual and physical environments, even with equivalent activities, engender different behaviors.

VVBOT and PVBOT students performed differently on post-test questions in the types and numbers of circuits they built and in the ways that they built their circuits. Many of the statistical measures we used were significantly different between VVBOT and PVBOT students. However, neither group was significantly "better" on the post-test or in their circuit quality. Some researchers might look to a work such as this one to judge the value of physical or virtual robotics for teaching and learning; we found that they both work well, albeit differently. To an extent, the virtual environment favors agent-based modeling from an aggregate perspective, and the physical environment favors aggregate modeling from an agent-based perspective.

Tinkering and sharing helps students learn complex systems and computational fluencies.

Sharing, tinkering, and performance were strongly correlated. Our data showed that students who tinkered and shared performed better on the post-test, regardless of pre-test performance. Gender, race, teacher, and school were uncorrelated to performance. All of the students learned how to create at least passable working VBOT circuits in the span of 5 days.

8.6 How did we add to the field?

VBOT is a new paradigm and language, built from previous successful systems – this is a virtual/physical real-time robotics system that can be used and adapted further in the field. A primary addition to the literature comes through the comparison of virtual and physical constructionist environments. There are few studies that systematically compare virtual and physical robotics in a constructionist frame. Among other findings, our data suggest that virtual and physical students interpreted similar problems remarkably differently: VVBOT students tended to understand things in context, in a networked way, where PVBOT students came to a more thorough understanding of the way that computation and logic works.

Another addition to the field comes through the local definitions and elucidation of complex systems and computational fluencies. By breaking down these categories into material, cognitive, and social, as per DiSessa (2000), we have broadened the definitions, so that can be used in other studies. This is a new, adapted definition of complex systems fluency, but it has relevance outside of this work. Furthermore, we have defined a set of design principles, which can be used to teach complex systems and/or computational fluency. These design principles have been verified both in practice and in data, and they should prove helpful for future designers building with related goals.

Tinkering and sharing were shown to be significantly correlated with performance and activity within the system. Male and female students were roughly equal on every measure; race, school, and teacher effects were quite minor. This suggests that by emphasizing tinkering and sharing, we are emphasizing learning. It could be argued that students tinkered and did better because they were more motivated by the system, but this is not a negative finding: this simply shows that the more fun a task is, the better the students perform.

Prescriptions

The following section describes how to use the information in this work to design towards complex systems and computational fluency. These prescriptions are positive, offering suggestions for organizing classrooms and designing learning environments.

Making the classroom more constructionist

Constructionism has been repeatedly correlated with engagement, creativity, and robust learning (see Bruckman, Edwards, Elliott, & Jensen, 2000). Essentially, constructionism means learning through building both knowledge and artifacts. The concepts behind constructionism are both deceptively simple and deceptively complex. What has been shown here, in Papert's work (1980), and countless other places (e.g., Harel & Papert, 1990) is that a designer can exploit active building towards engagement in learning design. People like to make artifacts. People learn better and care more about what they learn by interacting with and building artifacts than they do by hearing about them. In my opinion, no gain observed in this study is independent of constructionism.

Emphasizing computational fluency in your design

Computational fluency becomes more important every day. To be an informed citizen of the 21st Century, we need not only a literary (suggested by Orwell, 1947) and scientific fluency (AAAS, 2007), but a computational fluency as well. Computational fluency involves not only familiarity with computers but also a fundamental understanding of how and why they work and what they can and cannot effectively do (DiSessa, 2000). Computational fluency is difficult to gain without actually programming computers or designing digital artifacts. Only by programming a computer can a student understand why computers look and feel the way that they do and that they are essentially simple machines.

In keeping with the constructionist philosophy, our work demonstrates that the best way to emphasize computational fluency in research design is to engage students in tasks that require it. The programming should stem from the task. Students using VBOT never questioned why we were using computers to play with robots – they were too busy playing a game with VBOT to grumble about having to learn it. From the students' perspective, the games were driving the medium; the medium was not driving the game.

Emphasizing complex systems fluency in your design

Arguing for science fluency is positive, but science fluency has not been taught effectively in this country (TIMSS, 2003). One explanation is that traditionally taught science is both boring and hard, though science itself is neither boring nor hard. It only seems boring and hard because it is presented inappropriately; Students are rarely motivated by material that is packaged for educators or scientists. VBOT students learned about complex systems science because it was necessary to play the game and because it simply flowed from the material. NetLogo (1999) contains many examples of simulations in which phenomena are understandable through complex systems methods that would be otherwise difficult or needlessly mathematical. When approaching science content, it is necessary to approach it from ¸unique perspectives and consider the elements that make up the phenomenon; it is necessary to ask why phenomena happens, rather than simply how they happen.

Using robotics effectively

Robotics is a powerful learning tool. It is novel, even to the expert; it is futuristic, even to the programmer; and artificial intelligence has significant depth. Robotics essentially offers low-cost student engagement in a subject with significant depth. Better yet, it can be applied to a variety of scientific and mathematical phenomena. Rather than programming a LEGO robot follow a line, program it to act like a squirrel. Rather than building a robot to bring you a bagel, make a robot phonograph. There are numerous works in which robotics is used in remarkably novel ways, most of which are accessible even to the tech novice (e.g., AgentSheets, from Repenning, Ioannidou, & Zola, 2000; Rusk, Berg, & Resnick, 2005).

Teaching students to share more effectively

Students share naturally; most middle-schools students, for instance, love to talk incessantly.

Here is my prescription: let them talk. Let them talk off subject, but give them topics without one right answer. There is no such thing as "cheating" in VBOT, and this helped make the project run much more smoothly. In many of the activities, two students had to fill sufficiently different roles in which talking helped both of them but copying helped neither. In a free market, cheating rarely works– two companies that sell the exact same product will both sell half as much of it. However, two companies that work in synchrony create more business for other companies. To maximize sharing, students can be considered independent entities working in a free market. Opportunities for synchrony lead to much better results than identical competition.

8.7 Where were the problems?

Many veteran teachers today have a reasonable dislike of classroom innovations due to the spotty history of the novel classroom innovation (Tyack & Cuban, 1995). Our study was big enough to get reasonable statistical significance, but small enough that there are obvious holes. There is no way to assure that it will work everywhere. Nobody has attempted to deploy VBOT in a classroom where I was not present. This remains a future goal, and it is discussed below.

A limitation of the study is that I was not able to collect and analyze the social network data collected in the second pilot (as described in Section 4.2). Although there is reason to believe that fruitful data exists in examining social networks over time, the researcher resources necessary to collect that data in a “real-time” simulation study makes that collection difficult to do. As such, we were unable to do it in this study.

Like many design research studies, issues of transfer arose (as described by Perkins & Salomon, 1988). How does one know that the local learning translates to global learning? Will these students be better at programming when they see it again? Do they show complex systems fluency gains a year later? We attempted to design the pre-test and post-test to be meaningfully different from the VBOT activity (see Appendix 2), but transfer can be notoriously difficult to determine or understand.

8.8 Future directions

So where do we go from here? In keeping with the theme of this conclusion, our path can be divided into two prongs: understanding fluencies and improving design.

The cognitive aspects of complex systems fluency are only beginning to be understood (Goldstone & Wilensky, 2008). It is not yet know what cognitive processes people use to understand complex systems, neither is it known how they translate that understanding into material action. Work could be done within this system to address this topic. While it is difficult to get speak-aloud data of step-by-step student understanding because of the real-time aspects of

VBOT, it would be possible to more carefully consider one student's experience in a VBOT classroom. With an automated notation tool for individual students and individual cameras, it might be possible to investigate all of the actions of a specific student to understand their thought-processes more fully.

There is not much new research on assessment design in this study. Much more could be done by spending research time and money on improving the automated and textual aspects of assessment. Automatic assessments, individualized assessments, and reflective assessments would be relatively easy to integrate into VBOT and would increase the value of the data towards understanding fluencies. The approach to social network data tried in the second VBOT pilot (P2, detailed in Section 4.2) provides a different and potentially very valuable perspective on constructionist classrooms. Work in automatic and individualized assessments might provide a way to collect and analyze this social network data with fewer necessary resources.

This research has several implications for design. Many of the findings and much of the data suggest that students who play with the system more enjoy it more; those students then do better on assessments and in the activities. As such, there is a wealth of new research on games and learning that would be invaluable (e.g., Gee, 2003; Squire, 2003). There are many questions about the relationship between tinkering, motivation, and performance that are unanswerable with our design. Perhaps the most exciting next step would be to implement a set of targeted "tinkering games" that could be tested in various scenarios. During the original design of the current iteration of VBOT, Ian Horswill and I discussed a VBOT-like massively multiplayer online game (MMO) in which the data could be collected from players on the Internet. There are few learning games that leverage the "liveness" or complexity aspects of VBOT, and our design would likely benefit many users.

8.9 In Summary

Learning, playing, and sharing are complex activities. By tinkering, playing, and sharing, students came to better understand a complex and complicated set of content in a short period of time. There is a significant amount of research debating the difficulty of teaching computational fluency to younger students (e.g., AAUW, 2000; Cuban, 1985; Pea, 1987), debating the difficulty of teaching complex systems fluency to younger students (e.g., Hmelo-Silver & Pfeffer, 2004; Levy, 2002), and, possibly as a result, little research on combining the two. In using a "live" game-like simulation, our students were motivated, they met our learning goals, and they improved on our assessments. The virtual VBOT students and the physical VBOT students learned different things differently, but both sets learned content that few middle school students are able to learn. In this study, context trumped content. Exciting concepts presented in boring ways often seem difficult to learn; exciting concepts presented in exciting ways often seem easy to learn.

REFERENCES

AAAS. (2007). Atlas of Science Literacy, Volume 2. Washington, DC: AAAS and National Science Teachers Association.

AAUW. (2000). Tech-Savvy: Educating Girls in the Computer Age. Washington, DC: American Association of University Women.

Abrahamson, D. (2006). The shape of things to come: The computational pictograph as a bridge from combinatorial space to outcome distribution. *International Journal of Computers for Mathematics Learning*, 11(1), 137-146.

Abrahamson, D., & Wilensky, U. (2004a). ProbLab: A computer-supported unit in probability and statistics. In M.J. Hoines & A.B. Fuglestad (Eds.), *Proceedings of the 28th Annual Meeting of the International Group for the Psychology of Mathematics Education Vol. 1* (p. 369). Bergen: Bergen University College.

Abrahamson, D., & Wilensky, U. (2004b). SAMPLER: Collaborative interactive computer-based statistics learning environment. Paper presented at the 10th International Congress on Mathematical Education, Copenhagen, July 4 - 11, 2004.

Abrahamson, D., & Wilensky, U. (2005). Collaboration and equity in classroom activities using Statistics As Multi-Participant Learning-Environment Resource (S.A.M.P.L.E.R.). The annual meeting of the American Educational Research Association, Montreal, Canada, April 11 - 15, 2005.

Abrahamson, D., Berland, M., Shapiro, R., Unterman, J., & Wilensky, U. (2006). Leveraging epistemological diversity through computer-based argumentation in the domain of probability. *For the Learning of Mathematics*, 26(3), 39-55.

Abrahamson, D., Blikstein, P., Lamberty, K. K., & Wilensky, U. (2005). Mixed-media learning environments. Proceedings of the annual meeting of Interaction Design and Children 2005, Boulder, Colorado.

Ames, C., & Archer, J. (1988). Achievement goals in the classroom: Student learning strategies and motivation processes. *Journal of Educational Psychology*, 80, 260-267.

Aristotle. (1996). *Poetics*. Malcolm Heath (Trans.) London: Penguin.

Arkin, R.C. (1998). *Behavior-Based Robotics*. Cambridge, MA: MIT Press.

Au, W.J. (2002). Triumph of the Mod. Retrieved May, 2002, from http://archive.salon.com/tech/feature/2002/04/16/modding/

Babarasi, A. (2002). *Linked: The New Science of Networks*. Cambridge, MA: Perseus Press.

Bar-Yam, Y. (1997). *Dynamics of Complex Systems*. Cambridge, MA: Perseus Press.

Beck, K. (1999). *Extreme Programming Explained.* Boston, MA: Addison-Wesley.

Berland, M., & Wilensky, U. (2004). Virtual robotics in a collaborative constructionist learning environment. The annual meeting of the American Educational Research Association, San Diego, CA, April 12 - 16, 2004.

Berland, M., & Wilensky, U. (2005). Complex play systems -- Results from a classroom implementation of VBOT. The annual meeting of the American Educational Research Association, Montreal, Canada, April 11 - 15, 2005.

Berland, M. (2006). Constructionist collaborative engineering: Results from an Implementation of PVBOT. Paper presented at the annual meeting of the American Educational Research Association, San Francisco, CA.

Blikstein, P., & Wilensky, U. (2005). Less is more: Agent-based simulation as a powerful learning tool in materials science. Paper presented at the 4th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2005), Utrecht, Netherlands.

Bonabeau, E., Dorigo, M., & Theraulaz, G. (1999). *Swarm Intelligence: From Natural to Artificial Systems*. Oxford: Oxford University Press.

Braitenberg, V. (1984). *Vehicles: Experiments in Synthetic Psychology.* Cambridge, MA: MIT Press.

Brandes, A., & Wilensky, U. (1991). Treasureworld: An Environment for the Study and Exploration of Feedback. In I. Harel & S. Papert (Eds.), *Constructionism*. Norwood, MA: Ablex Publishing.

Brooks, R. (1991). Artificial life and Real Robots, Proceedings of ECAL91, 3-10.

Brooks, R. (1999). *Cambrian Intelligence*. Cambridge, MA: The MIT Press.

Brooks, R. (2002). *Flesh and Machines*. New York: Pantheon Books.

Brooks, R., & Stein, L. (1994). Building brains for bodies. *Autonomous Robots* 1(1):7-25.

Brown, J.S., & Duguid, P. (2002). *The Social Life of Information*. Cambridge, MA: Harvard Business School Press.

Bruckman, A. (1997). MOOSE crossing: Creating a learning culture. Thesis for the degree of Doctor of Philosophy at MIT.

Bruckman, A., Edwards, E., Elliott, J., & Jensen, C. (2000). Uneven Achievement in a Constructionist Learning Environment. Proceedings of ICLS 2000. Ann Arbor, MI.

Bruckman, A., Jensen, C., & DeBonte, A. (2002). Gender and Programming Achievement in a CSCL Environment. Long talk, Proceedings of CSCL 2002, Boulder, CO.

Cassirer, E. (1979). *Symbol, Myth, and Culture. Essays and Lectures of Ernst Cassirer 1935-1945*. New Haven, CT: Yale University Press.

Centola D., McKenzie E., & Wilensky, U. (2000). Survival of the Groupiest: Facilitating Students' Understanding of Multi-level Evolution through Multi-Agent Modeling - The EACH Project. The Fourth International Conference on Complex Systems. Nashua, NH: New England Complex Systems Institute.

Chi, M.T.H. (2005). Commonsense conceptions of emergent processes: Why some misconceptions are robust. *Journal of the Learning Sciences*, 14(2).

Chi, M.T.H., deLeeuw, N., Chiu, M., & LaVancher, C. (1994). Eliciting self-explanations improves understanding. *Cognitive Science*, 18, 439-477.

Cole, M. (1996). *Cultural psychology: A once and future discipline*. Cambridge, MA: Harvard University Press.

Collela, V. (2000). Participatory Simulations: Building Collaborative Understanding through Immersive Dynamic Modeling. *Journal of the Learning Sciences*, 9(4), pp. 471-500.

Collins, A. (1992). Toward a design science of education. In E. Scanlon & T. O'Shea (Eds.), *New directions in educational technology* (pp. 15 - 22). New York: Springer.

Csikszentmihalyi, M. (1990). *Flow: the psychology of optimal experience*. New York: Harper & Row.

Cuban, L. (1986). *Teachers and machines: The classroom use of technology since 1920*. New York: Teachers College Press.

Cypher, A., & Smith, D.C. (1995). KidSim: End User Programming of Simulations. CHI'95, Denver.

DeBonte, A. (1998). Pet Park: a graphical constructionist community. Unpublished Master's Thesis, MIT, Cambridge, MA.

Dewey, J. (1897). My Pedagogic Creed. *The School Journal*, 3, 77-80.

Dewey, J. (1913). *Interest and effort in education*. Boston, MA: Houghton Mifflin Co.

Dewey, J. (1931/1985). Context and thought. In Jo Ann Boydston (Ed.), *John Dewey: The later works, Vol. 6* (pp. 3 - 21). Carbondale, IL: Southern Illinois University Press.

diSessa, A. (1997). Twenty reasons why your should use Boxer (instead of Logo). In M. Turcsányi-Szabó (Ed.), Learning & Exploring with Logo: Proceedings of the Sixth European Logo Conference, Budapest, Hungary

diSessa, A. (2000). *Changing minds: Computers, language and literacy*. Cambridge, MA: MIT Press.

diSessa, A., & Abelson, H. (1986). Boxer: A reconstructible computational medium. *Communications of the ACM*, 29(9)

Dodds, Z., & Tribelhorn, B. (2006). Erdos: Cost effective peripheral robotics for AI education. In Proceedings, 2006 AAAI, pp. 1966-1967.

Druin, A., & Hendler, J., Eds, (2000). *Robots for Kids: Exploring New Technologies for Learning*. San Francisco: Morgan Kaufmann Publishers.

Dweck, C.S., & Elliot, E.S. (1983). Achievement motivation. In P. H. Mussen & E. M. Hetherington (Eds.), *Handbook of child psychology* (Volume IV: Social and personality development, pp. 643-691). New York: Wiley.

Edelson, D.C. (2002). Design research: What we learn when we engage in design. *Journal of the Learning Sciences*, (11)1, 105-122.

Elliott, J. & Bruckman, A. (2002). Design of a 3D Interactive Math Learning Environment. *Design of Interactive Systems*, London, UK.

Elliott, J., Adams, L., & Bruckman, A. (2002). No Magic Bullet: 3D Video Games in Education. Proceedings of ICLS, 2002.

Epstein, J.M., & Axtell, R. (1996). *Growing artificial societies: social science from the bottom up*. Cambridge, MA: MIT Press.

Ericsson, K.A. & Simon, H.A. (1984). *Protocol Analysis: Verbal reports as data*. Cambridge, MA: MIT Press.

FIRST Robotics. (2006). Retrieved April 4, 2008 from: http://www.usfirst.org/what/frc/default.aspx?id=366

Gee, J. (2003). *What video games have to teach us about learning and literacy*. New York: Palgrave.

Gee, J. (2004). Learning by Design: Games as Learning Machines. Game Developers Conference, San Jose, CA.

Goldin-Meadow, S. (2003). *Hearing gesture: How our hands help us think.* Cambridge, MA: Harvard University Press.

Goldstone, R. & Wilensky, U. (2008). Promoting transfer through complex systems principles. *Journal of the Learning Sciences* (Manuscript in press).

Gutierrez, K., Rymes, B., & Larson, J. (1995). Script, Counterscript, and Underlife in the Classroom: James Brown versus Brown v. Board of Education. *Harvard Educational Review*, 65(3), 445-471.

Hancock, C. (2001). Children's Understanding of Process in the Construction of Robot Behaviors. Paper presented at the symposium "Varieties of Programming Experience." The Annual Meeting of the American Educational Research Association, 2001, Seattle.

Hancock, C. (2003). *Real-time programming and the big ideas of computational literacy.* Unpublished doctoral dissertation, MIT.

Haraway, D. (1991). "A Cyborg Manifesto: Science, Technology, and Socialist-Feminism in the Late Twentieth Century," in *Simians, Cyborgs and Women: The Reinvention of Nature*. New York: Routledge.

Harel, I., & Papert, S. (1990). Software design as a learning environment. *Interactive Learning Environments*, 1(1), 132.

Harvey, B. (1994). Is Programming Obsolete? Retrieved April, 2005 from http://www.cs.berkeley.edu/~bh/obsolete.html

Heath, S. B. (1999). Imaginative actuality: learning in the arts during nonschool hours. In E. B. E. Fiske & President's Committee on the Arts and the Humanities Washington DC. & Arts Education Partnership Washington DC. (Eds.), Champions of Change: The Impact of the Arts on Learning (pp. 115). District of Columbia.

Hmelo-Silver, C. & Pfeffer, M.G. (2004). Comparing expert and novice understanding of a complex system from the perspective of structures, behaviors, and functions. *Cognitive Science*, 28, 127-138.

Holland, J. (1995). *Hidden Order: How Adaptation Builds Complexity*. Reading, MA: Perseus Books.

Husserl, E. (1913/1982). *Ideas Pertaining to a Pure Phenomenology and to a Phenomenological Philosophy -- First Book: General Introduction to a Pure Phenomenology*, trans. F. Kersten. The Hague: Nijhoff.

Ioannidou, A., Rader, C., Repenning, A., Lewis, C., & Cherry, G. (2003). Making Constructionism Work in the Classroom. *International Journal of Computers for Mathematical Learning*, 8(1), 63-108.

Jacobson, M. & Wilensky, U. (2006). Complex systems in education: Scientific and educational importance and implications for the learning sciences. *Journal of the Learning Sciences*, 15(1), pp. 11-34.

Jakobi, N., Husbands, P., & Harvey, I. (1995). Noise and the Reality Gap: The Use of Simulation in Evolutionary Robotics. ECAL 1995: 704-720.

Jones, J. L. (2004). *Robot programming: a practical guide to behavior-based robotics*. New York, London: McGraw-Hill.

Kafai, Y. (1995). *Minds in play*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Kahn, K. (1996). ToonTalk - An animated programming environment for children. *Journal of Visual Languages and Computing*, 7(2): 197-217.

Kauffman, S. (1995). *At home in the universe: The search for the laws of self-organization and complexity*. Oxford: Oxford University Press.

Khoo, A. (2003). *Implementing efficient joint beliefs on multi-robot teams*. Unpublished PhD Dissertation. Northwestern University Computer Science

Klassner, F. (2002). A case study of LEGO Mindstorms suitability for artificial intelligence and robotics courses at the college level. In Proceedings of the 33rd SIGCSE Technical Symposium on Computer Science Education, 8-12

Koda, T. & Maes, P. (1996). Agents with Faces: The Effects of Personification of Agents. Proceedings of HCI, 1996.

Latour, B. (1987). *Science in action: How to follow scientists and engineers through society.*

Cambridge, MA: Harvard University Press.

Lawhead, P., Duncan, M., Bland, C., Goldweber, M., Schep, M., Barnes, D., & R. Hollingsworth. (2003). A Road Map for Teaching Introductory Programming Using LEGO(c) Mindstorms Robots. ACM SIGCSE Bulletin 35(2).

LCSI (1999). *Logo Philosophy and Implementation*. Quebec: Logo Computer Systems Inc

LEGO Creator (2002). Retrieved April, 2005, from http://www.lego.com/creator/

Levy, S.T. (2002). Casting an anchor before floating off: The shift from simple to complex reasoning. Paper presented at the 32nd Annual Meeting of the Jean Piaget Society, Philadelphia, PA, USA.

Levy, S.T., & Wilensky, U. (2007). How do I get there... straight, oscillate or inch? High-school students' exploration patterns of Connected Chemistry. Paper presented at the 2007 annual meeting of the American Educational Research Association, Chicago, IL.

Levy, S.T., & Wilensky, U. (2008). Inventing a "mid-level" to make ends meet: Reasoning through the levels of complexity. *Cognition & Instruction*, 26(1), 1-47.

Levy, S.T., Mioduser, D., & Talis, V. (2001). Concrete-abstractions stage in kindergarten children's perception and construction of robotic control rules. Paper presented at the PATT-2001 conference: New Media in Technology Education, Haarlem Holland.

Maani, K., & Cavana, R. (2000). *Systems thinking and modeling: Understanding change and complexity*. Auckland: Prentice Hall.

Maes, P. (1991). A Bottom-up Mechanism for Behavior Selection in an Artificial Creature, Proceedings of the first International Conference on Simulation of Adaptive Behavior, Meyer J.A. & Wilson S. (eds). MIT Press.

Mandinach, E. B., & Thorpe, E. T. (1987). The systems thinking and curriculum innovation project: Technical report, part 1. (TR-87). Educational Technology Center. Harvard Graduate School of Education. Nichols House, Appian Way. Cambridge MA 02138.

Manguel, A. (1997). *A history of reading*. London: Flamingo.

Martin, F. (1989). 6.270: The MIT LEGO Robot Design Project Competition. Retrieved November, 2004, from http://fredm.www.media.mit.edu/people/fredm/projects/6270/.

Martin, F. (1996a). Ideal and Real Systems: A Study of Notions of Control in Undergraduates Who Design Robots. In Y. Kafai, M. Resnick (Eds.), *Constructionism in Practice*. Mahwah, NJ: Lawrence Erlbaum.

Martin, F. (1996b). Kids Learning Engineering Science using LEGO and the Programmable Brick. In the Proceedings of the Annual Meeting of the American Educational Research Association, April, 1996.

Martin, F. (2000). *Robotic explorations: A hands-on introduction to engineering*. Upper Saddle River, NJ: Prentice Hall.

MindRover (2001). Retrieved June, 2003, from http://mindrover.com

National Research Council (2002). *Scientific research in education*. Washington, DC: National Academy Press.

NQC (2005). Retrieved April 4, 2008, from http://bricxcc.sourceforge.net/nqc/

Orwell, G. (1947). *The English People*. London: Collins.

Papert, S. (1980). *Mindstorms: Children, Computers, and Powerful Ideas*. New York: Basic Books.

Papert, S. (1991). Situating Constructionism. In I. Harel and S. Papert. (Eds.), *Constructionism*. Norwood, NJ: Ablex.

Papert, S. (1996). *The Connected Family*. Atlanta: Longstreet Publishing.

Parker, L.E., Schneider, F., & Schultz, A. (eds.), (2005). *Multi-Robot Systems: From Swarms to Intelligent Automata, Volume III*. Berlin: Springer.

Parsons, S. & Sklar, E. (2004). Teaching AI using LEGO Mindstorms. AAAI Spring Symposium 2004 on Accessible Hands-on Artificial Intelligence and Robotics Education.

Pea, R. D. (1987). Programming and problem-solving: Children's experiences with Logo. In T. O'Shea & E. Scanlon (Eds.), *Educational computing (An Open University reader)*. London: John Wiley & Sons.

Penner, D.E., (2001). Cognition, Computers and Synthetic Science: Building Knowledge and Meaning Through Modeling. *Review of Research in Education*, 25: 1-36

Perkins, D.N., & Salomon, G. (1988). Teaching for transfer. *Educational Leadership*, 46(1), 22-32.

Petrarch, F. (1948). *The Renaissance Philosophy of Man*, Eds. E. Cassirer, P.O. Kristeller, Jr., J.H. Randall. Chicago: University of Chicago Press.

Piaget, J. (1952). *The child's conception of number*. London: Routledge and Kegan Paul.

Piaget, J. (1972). Intellectual evolution from adolescence to adulthood. *Human Development*, 15, 1-12.

Pintrich, P.R., & Schunk, D.H. (1996). *Motivation in education: Theory, research, and applications*. Englewood Cliffs, NJ: Prentice Hall.

Pollack J. B., Lipson H., Funes P., & Hornby G. (2001). Three Generations of Co-evolutionary Robotics. *Artificial Life*, 7,3.

Portsmore, M. (1999). ROBOLAB: Intuitive Robotic Programming Software to Support Life Long Learning. *APPLE Learning Technology Review*, Spring/Summer, 1999.

PROLOG (1972). Logic programming language.

Raymond, E. S. (1999). *The Cathedral and the Bazaar*. Sebastopol, CA: O'Reilly & Associates.

Repenning, A. (1993). *Agentsheets: A Tool for Building Domain-Oriented Dynamic, Visual Environments*. PhD thesis, University of Colorado.

Repenning, A., Ioannidou, A., & Zola, J. (2000). AgentSheets: End-User Programmable Simulation. *Journal of Artificial Societies and Social Simulation, Vol. 3.*

Resnick, M. (1994a). *Turtles, termites and traffic jams: Explorations in massively parallel microworlds*. Cambridge, MA: MIT Press.

Resnick, M. (1994b). Learning about life. *Artificial Life Journal*, vol. 1, no. 1-2, pp. 229-241.

Resnick, M. (1998). Technologies for Lifelong Kindergarten. *Educational Technology Research and Development*, vol. 46, no. 4.

Resnick, M., & Ocko, S. (1991). "LEGO/Logo: Learning Through and About Design." In (ed. by I. Harel and S. Papert (Eds.), *Constructionism*. Norwood, NJ: Ablex.

Resnick, M., & Wilensky, U. (1998). Diving into Complexity: Developing Probabilistic Decentralized Thinking through Role-Playing Activities. *Journal of Learning Sciences*, 7(2).

Resnick, M., Martin, F., Sargent, R., & Silverman, B. (1996). Programmable bricks: toys to think with. *IBM Systems Journal*, 35(3&4), 443-452.

RoboCup (1997). Proceedings of RoboCup, 1997.

RoboLab. (2005). Software. Retrieved April 4, 2008, from http://www.ceeo.tufts.edu/graphics/robolab.html

RobotWar. (1981). Software. Retrieved April 4, 2008, from http://www.the-underdogs.org/game.php?name=RobotWar

Rocky's Boots. (1982). Software. Retrieved April 4, 2008, from http://www.warrenrobinett.com/rockysboots/

Rusk, N., Berg, R. & Resnick, M. (2005). Rethinking Robotics: Engaging Girls in Creative Engineering. Proposal to the National Science Foundation. Retrieved April 4, 2008, from http://llk.media.mit.edu/projects.php?id=1942

Salomon, G., & Perkins, D.N. (1989). Rocky roads to transfer: Rethinking mechanisms of a neglected phenomenon. *Educational Psychologist*, 24(2), 113-142.

Schank, R. & Cleary, C. (1994). *Engines for education*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Schank, R.C., Fano, A., Bell, B., & Jona, M. (1994). The design of goal based scenarios. *Journal of the Learning Sciences*, 3(4), 305--346.

Second Life (2006). Retrieved April 4, 2008, from http://secondlife.com/

Sharlin, E., Watson, B.A. , Kitamura, Y., Kishino, F. & Itoh, Y. (2004). On humans, spatiality and tangible user interfaces. *Pervasive and Ubiquitous Computing*, 8, 338-346. Theme issue on Tangible Interfaces in Perspective.

Shaw, A. (1996). Social constructionism and the inner city: Designing environments for social development and urban renewal. In Y. Kafai and M. Resnick (Eds.), *Constructionism in practice: Designing, thinking, and learning in a digital world* (pp. 175-206). Mahwah, NJ: Lawrence Erlbaum Publishers.

Sklar, E., Eguchi, A., & Johnson, J. (2002). Robocupjunior: Learning with educational robotics. In Proceedings of the 6th RoboCup Symposium.

Sklar, E., Parsons, S., & Stone, P. (2003). Robocup in higher education: A preliminary report. In Proceedings of the 7th RoboCup Symposium.

Squire, K. & Barab, S.A. (2004). Replaying history. International Conference of the Learning Sciences (ICLS), Los Angeles, CA.

Squire, K. (2003). Video games in education. *International Journal of Intelligent Simulations and Gaming*, (2) 1.

Squire, K. (2004). Sid Meier's Civilization III. *Simulations and Gaming*, 35(1): 135-140.

Stager, G. (1996). Laptops, Logo and Learning. Retrieved April 4, 2008, from http://www.stager.org/articles/laptopsandlogo.html

Steinkuehler, C. A. (2004). Learning in massively multiplayer online games. International Conference of the Learning Sciences (ICLS), Los Angeles CA.

STELLA v 6.0. (2000). Software. Hanover NH: High Performance Systems.

Tiffin, J., & Rajasingham, L. (1995). *In search of the virtual class: Education in an information society*. London: Routledge.

TIMSS (2003). International Report on Achievement in the Mathematics Cognitive Domains: Findings from a Developmental Project. Retrieved April 4, 2008, from http://timss.bc.edu/timss2003.html

Tisue, S., & Wilensky, U. (2004). NetLogo: Design and Implementation of a Multi-Agent Modeling Environment. Paper presented at the Agent2004 Conference, Chicago, IL.

Travers, M. (1988). *Agar: An animal construction kit*. MS thesis, MIT Media Laboratory.

Turkle, S., & Papert, S. (1991). Epistemological Pluralism and the Revaluation of the Concrete. In I. Harel & S. Papert (Eds.), *Constructionism*. Norwood, NJ: Ablex.

Turkle, S. (1995). *Life on the screen: Identity in the age of the Internet*. New York: Simon and Schuster.

Tyack, D., & Cuban, L. (1995). *Tinkering Toward Utopia: A Century of Public School Reform.* Cambridge, MA: Harvard University Press.

Vygotsky, L.S. (1978). *Mind in society*. Cambridge, MA: Harvard University Press.

Wainer, J., Feil-Seifer, D., Shell, D., & Mataric, M. (2006). The role of physical embodiment in human-robot interaction. In IEEE Proceedings of the International Workshop on Robot and Human Interactive Communication (pp. 117-122). Hatfield, United Kingdom.

Watts, D.J. (2003). *Six degrees: The science of a connected age*. Cambridge, MA: W.W. Norton.

Wiener, N. (1948). *Cybernetics, or Control and communication in the animal and the machine*. New York: Wiley.

Wilensky, U. (1997a). What is normal anyway? Therapy for epistemological anxiety. *Educational Studies in Mathematics*, 33(2), 171-202.

Wilensky, U. (1997b). NetLogo Wolf Sheep Predation model. http://ccl.northwestern.edu/netlogo/models/WolfSheepPredation. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.

Wilensky, U. (1999). NetLogo. http://ccl.northwestern.edu/netlogo. Center for Connected Learning and Computer-Based Modeling. Northwestern University, Evanston, IL.

Wilensky, U. (2000). Modeling Emergent Phenomena with StarLogoT. Retrieved December, 2000, from CONCORD.org.

Wilensky, U. (2001). Modeling Nature's Emergent Patterns with Multi-agent Languages. Proceedings of EuroLogo 2001. Linz, Austria.

Wilensky, U., & Reisman, K. (1998). Learning Biology through Constructing and Testing Computational Theories -- an Embodied Modeling Approach (HTML). In Y. Bar-Yam (Ed.), Proceedings of the Second International Conference on Complex Systems. Nashua, NH: New England Complex Systems Institute.

Wilensky, U., & Reisman, K. (2006). Thinking Like a Wolf, a Sheep or a Firefly: Learning Biology through Constructing and Testing Computational Theories -- an Embodied Modeling Approach. *Cognition & Instruction*, 24(2), pp. 171-209.

Wilensky, U., & Resnick, M. (1999). Thinking in Levels: A Dynamic Systems Perspective to Making Sense of the World. *Journal of Science Education and Technology*, 8(1).

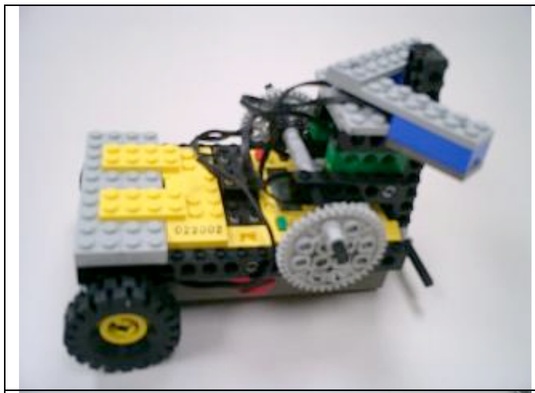
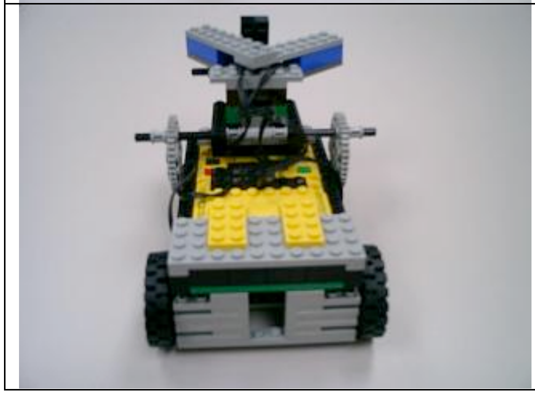
Wilensky, U., & Stroup, W. (1999a). Learning through Participatory Simulations: Network-Based Design for Systems Learning in Classrooms. Computer Supported Collaborative Learning (CSCL'99). Stanford University: December 12 - 15, 1999.

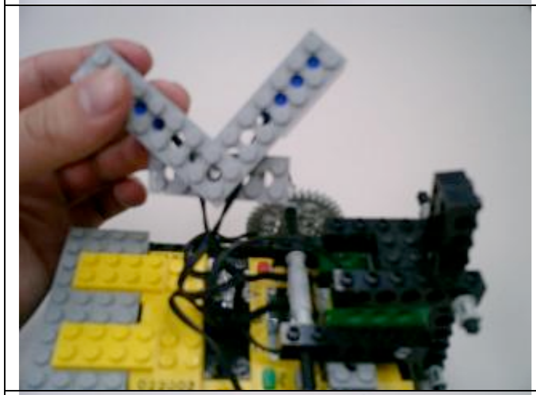
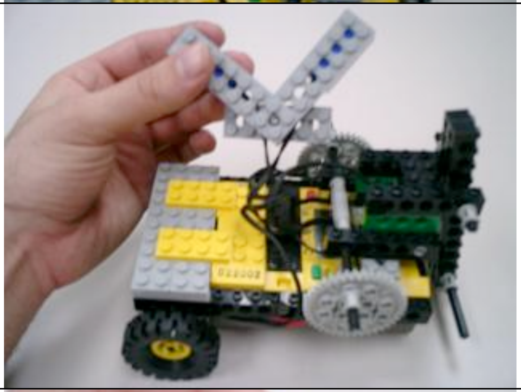
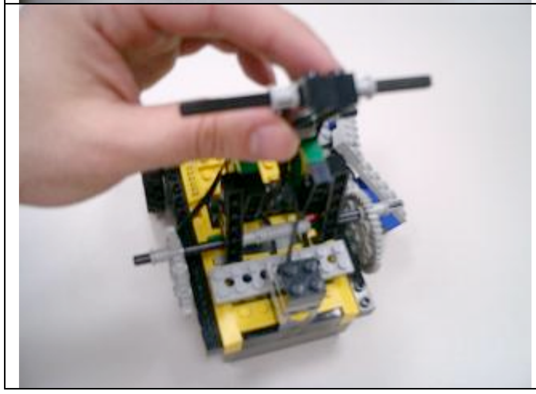
Wilensky, U., & Stroup, W. (1999b). HubNet. http://ccl.northwestern.edu/netlogo/hubnet.html. Center for Connected Learning and Computer-Based Modeling. Northwestern University, Evanston, IL.

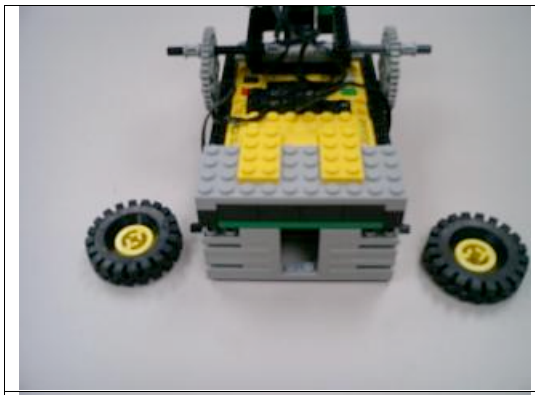
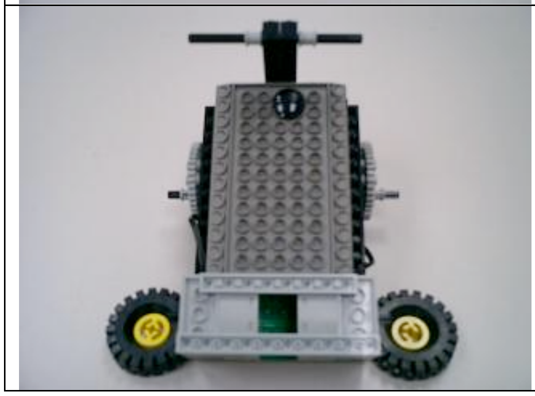
Wilensky, U., & Stroup, W. (1999c). NetLogo HubNet Gridlock model. http://ccl.northwestern.edu/netlogo/models/HubNetGridlock. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.

Wolfram, S. (2002). *A New Kind of Science*. Champaign, IL: Wolfram Media.

APPENDIX 1 – PHYSICAL VBOT

	Physical VBOT – Side view
	Physical VBOT – Side view
	Physical VBOT – Rear view

	Physical VBOT – Front view
	Physical VBOT – Light sensors
	Physical VBOT – Light sensors
	Physical VBOT – Bump sensor

	Physical VBOT – Wheels and motors
	Physical VBOT – Bottom view

APPENDIX 2 – ASSESMENT TOOLS

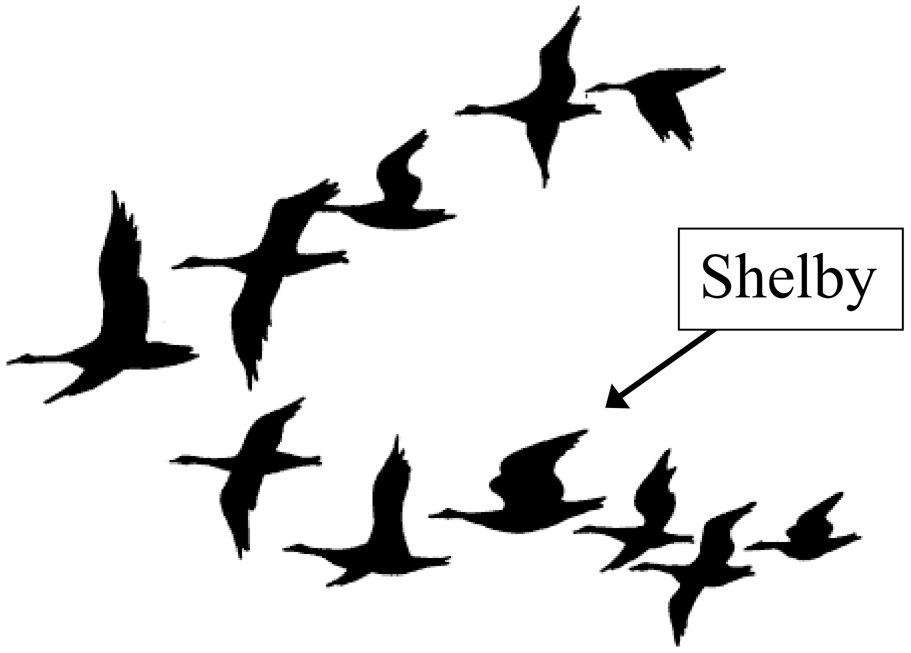
Pre-Questionnaire

Name:

Date:

Just write what you think! Answer each question with a sentence or two.

Thank you!



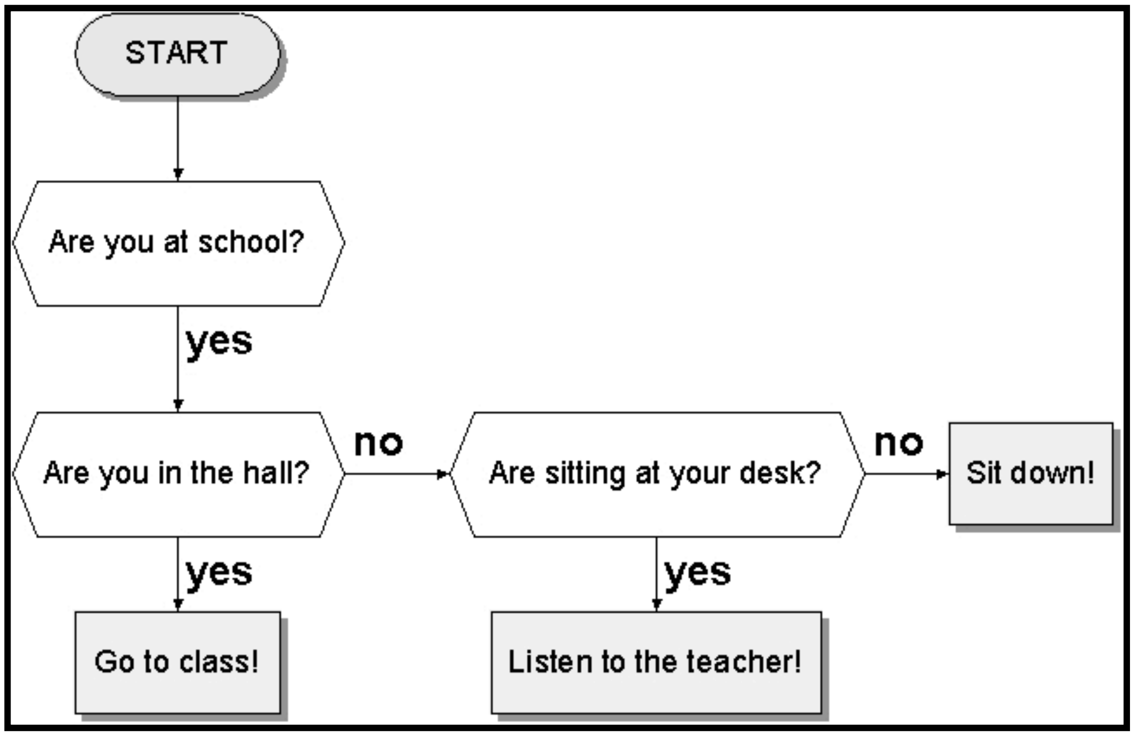
Q1

When birds fly south for the winter, they often form a V-shape. You might have seen this in the sky. This is a picture of flock of birds flying in V-shape:

How does Shelby, the bird, know where to fly in the V-shape?

How do the birds know where to go when they are flying in a V-shape?

Why do birds fly in V-shapes?



Q2

Imagine that this "flowchart" describes your day at school. You can follow the flowchart by answering questions about your day. Depending on your answers to the questions, it tells you what to do next. Start at "START" (in the image above) and follow the arrows.

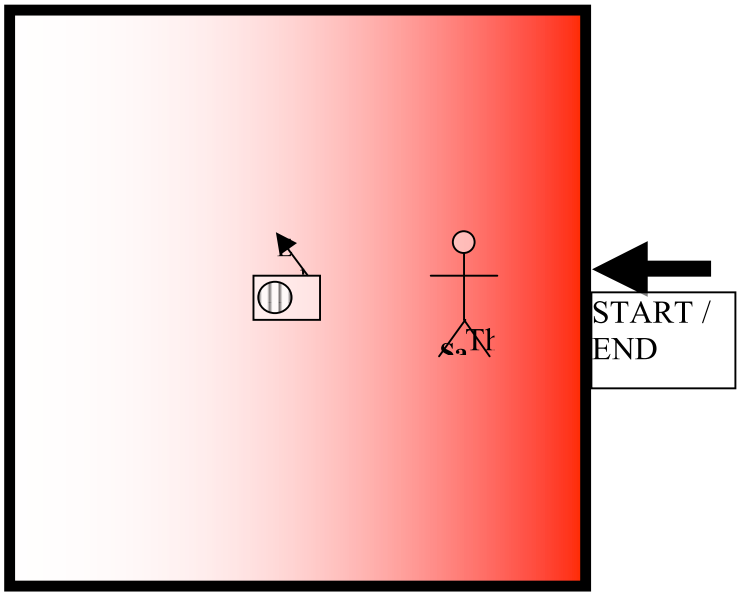
Have you ever seen a flowchart before? YES NO

Using the chart, list the things that happen during a day of school.

Change the flowchart so that if you are not at school, you read a book. (*Draw on the picture above.)*

Q3

You are in a pitch-black square room with heated walls. You can't see ANYTHING. In the center of the room, there is a radio. If you touch the middle of any wall you are about 5 feet from the radio. You can hear the radio and feel the heat from the wall to varying amounts from anywhere in the room.



How would you walk all the way around the radio at least once if you couldn't see ANYTHING? You start walking at START (on the right), and end at the same place (END). You can hear how far away you are from the radio and you can feel how close you are to the wall because of the heat from the walls. Do not touch the walls.

How would you make a path closer to the radio without touching the radio or the walls (if you still can't see anything)?

If the RADIO WERE TURNED OFF, how would you make a smaller loop around the radio without touching it or the walls (if you still can't see anything)?

If BOTH the RADIO and the HEAT FROM THE WALLS WERE TURNED OFF, how would you make a loop around the radio without touching the radio or the walls (if you still can't see anything)?

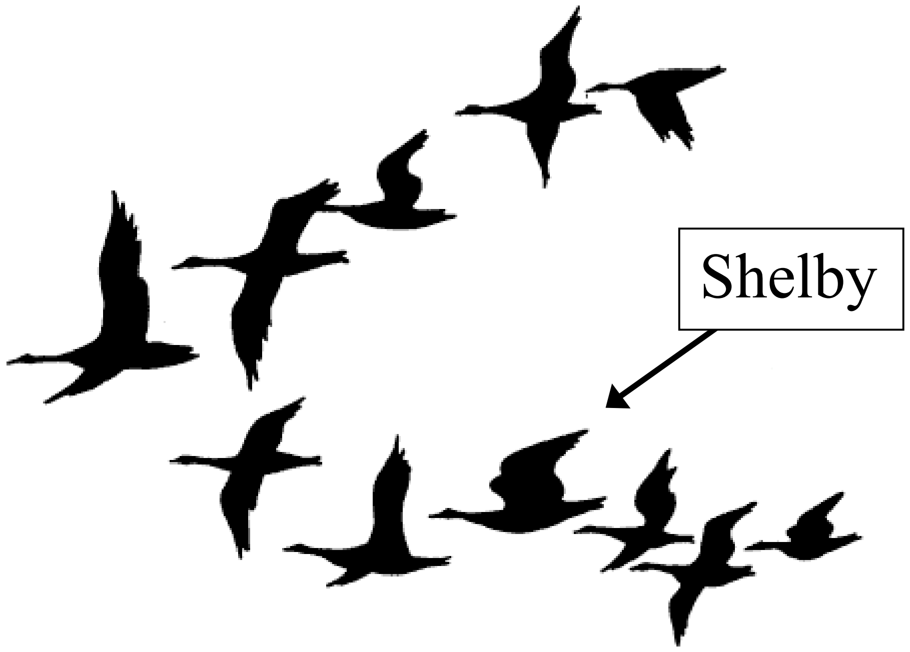
Post-Questionnaire

Name:

Date:

Just write what you think! Answer each question with a sentence or two.

Thank you!



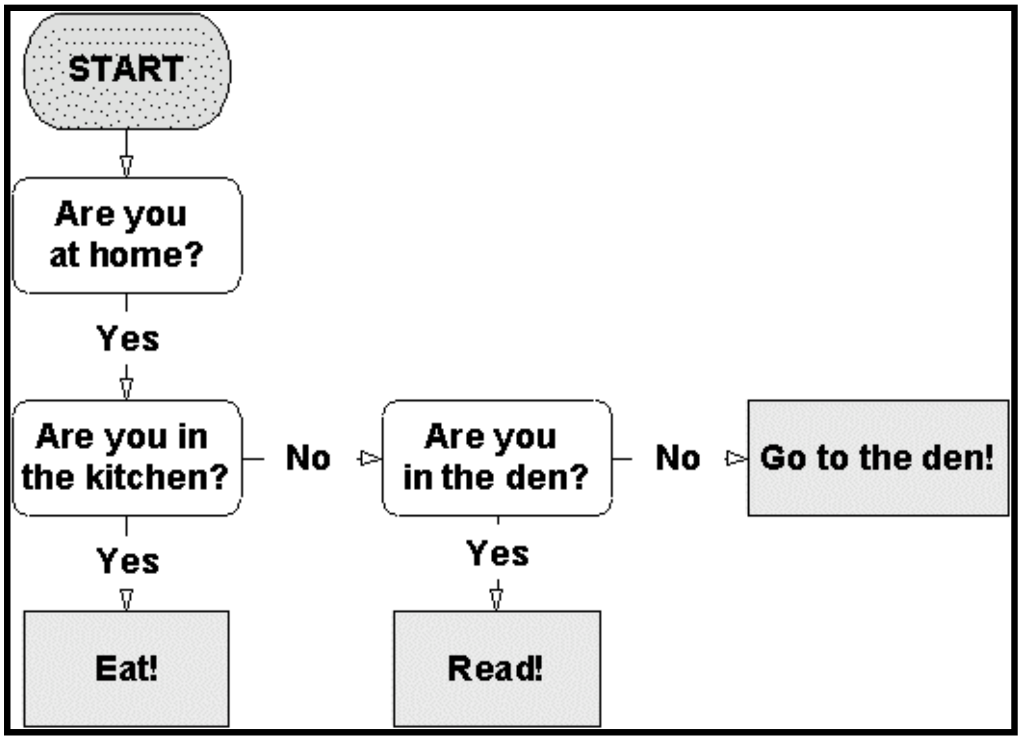
Q1

When birds fly south for the winter, they often form a V-shape. You might have seen this in the sky. This is a picture of flock of birds flying in V-shape:

How does Shelby, the bird, know where to fly in the V-shape?

How do the birds know where to go when they are flying in a V-shape?

Why do birds fly in V-shapes?



Q2

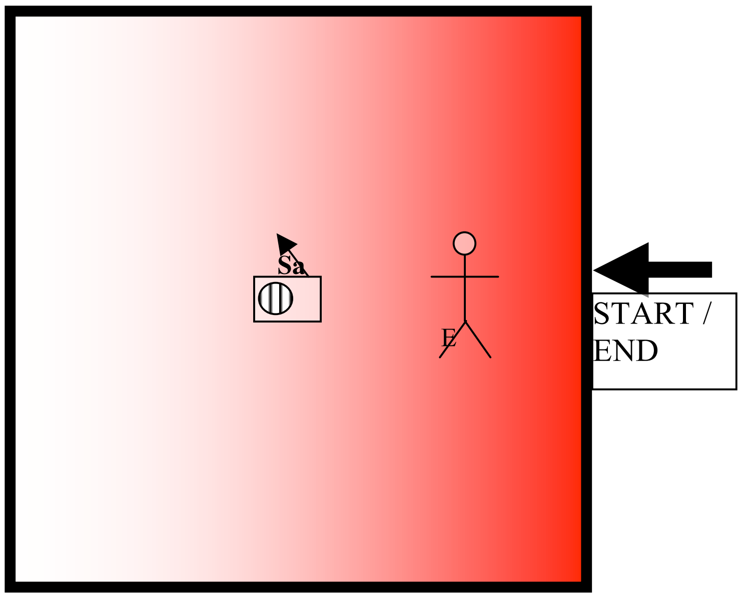
Imagine that this "flowchart" describes a day at home. You can follow the flowchart by answering questions about that day. Depending on your answers to the questions, it tells you what to do next. Start at "START" (in the image above) and follow the arrows.

Using the chart, list the things that happen during this day at home.

Change the flowchart so that if you are not at home, you go home. (*Draw on the picture above.)*

Q3

You are in a pitch-black square room with heated walls. You can't see ANYTHING. You can feel the heat from the walls when you touch them, and when you put your hand near them. In the center of the room, there is a radio. It is turned on. You can hear the radio pretty well. If you touch the middle of any wall you are about 5 feet from the radio.



How would you walk one full loop around the room if you couldn't see ANYTHING? You start walking at START (on the right), and end at the same place (END). You can hear how far away you are from the radio and you can feel how close you are to the wall because of the heat from the walls. **Do not touch the walls or the radio.**

How would you make a path closer to the radio without touching the radio or the walls (if you still can't see anything)?

If the RADIO WERE TURNED OFF, how would you make a smaller loop around the radio without touching it or the walls (if you still can't see anything)?

If BOTH the RADIO and the HEAT FROM THE WALLS WERE TURNED OFF, how would you make a loop around the radio without touching the radio or the walls (if you still can't see anything)?

Q4

Your vbot is the airplane in the picture. The arrows in the middle are other people's vbots, and they are stopped in the middle of the screen. There is a light source in the middle of the screen.

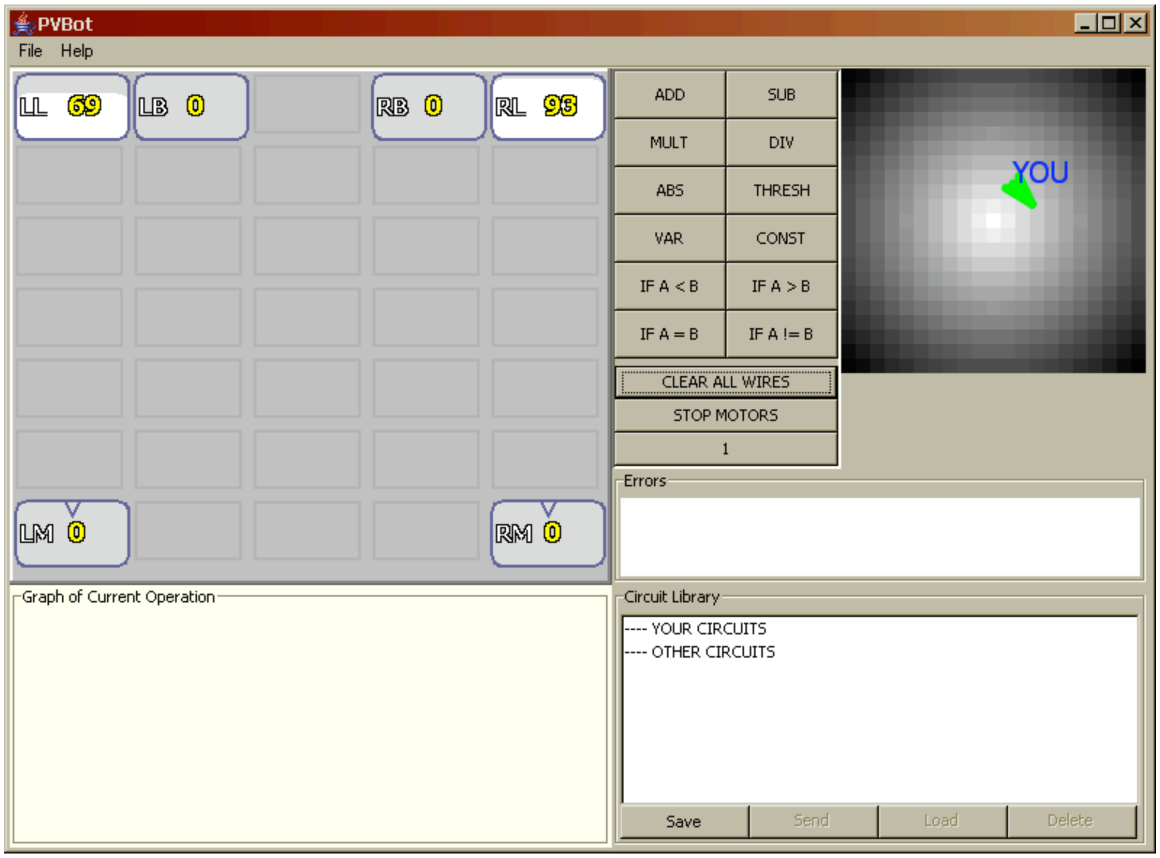
4.A. Wire up a vbot to go in a loop around the screen.
(Use attached vbot breadboard. ***Circle Q 4.A. at the top of the sheet.)***

4.B. Wire up a vbot that would make a smaller loop.
(*Use attached vbot breadboard.* ***Circle Q 4.B. at the top of the sheet****.)*

4.C. Wire up a vbot that makes either loop using NO LIGHT SENSORS (LL and LR)?
(*Use attached vbot breadboard.* ***Circle Q 4.C. at the top of the sheet****.)*

4.D. Wire up a vbot that makes either loop using NO VBOT SENSORS (LB and RB)?
(*Use attached vbot breadboard.* ***Circle Q 4.D. at the top of the sheet****.)*

Circle one:
Question 4.A.
Question 4.B.
Question 4.C.
Question 4.D.



Daily Questionnaire Name:

Circle one:

`Monday      Tuesday      Wednesday      Thursday      Friday`

Please list everybody in the class who you remember talking to during the VBOT lesson.

Please list everybody in the class who you remember working with during the VBOT lesson.

What was your favorite thing about today's VBOT lesson?

What was your least favorite thing about today's VBOT lesson?

In a couple of sentences, describe one thing you learned today.