Oats
(not just for breakfast anymore)

Megan Galbraith

Fall 2002
MAS.714J Systems and Self

Prof. Mitchel Resnick
Prof. Sherry Turkle
T.A. Michelle Hlubinka

The bicycle without a rider balances perfectly well.
With a novice rider, it will fall.
This is because the novice has the wrong intuitions about balancing
and freezes the position of the bicycle so that
its own corrective mechanism cannot work freely.

Thus learning to ride does not mean learning to balance,
it means learning not to unbalance,
learning not to interfere.

Seymour Papert illustrating Piaget's idea about the development of learning, in "MindStorms, Children, Computers, and Powerful Ideas"

Table of Contents

Educational tools developed for the computer are primarily designed for children. Bright colors, cartoon-like characters, animals, associations with toys, silly sounds, and graphical elements construct the interface for a myriad of programs designed as exploratory spaces. Through these programs, children can learn about and understand the complex world of computation. These programs are powerful because they provide a stepping stone for children to explore tasks in relation to computation, mathematics, science, identity, and, in some cases, social responsibility.

Software programs exist for adults that provide control capabilities for hardware and development spaces for software, but these programs are typically not for novice users. An inexperienced adult interested in building a project with a microcontroller has a variety of options to chose from, but the truth is that most programs are ineffective tools for learning the underlying concepts. The programs are difficult to set up, difficult to understand, and extremely intimidating. Some provide high level languages to control the devices, but still require a basic fluency in writing procedural programs. Others require a deep understanding of the circuity in order to write programs, so users must rely on dense maps and complicated circuit diagrams that show the connections between pins, buses, and other elements.

Adults with weak mathematics or science backgrounds think very differently about computers and computation than adults who come from stronger technical backgrounds. This is not to say that "non-technical" adults will not (or do not) find computers useful in their daily life, work, or hobbies. In general, people are becoming more adept at using computer programs or familiarizing themselves with operating systems and simulation environments. Many adults use computers to accomplish tasks such as word processing, email, or commerce. The catch is that these adults operate in abstraction levels far removed from the internal structure, design, and operation of the computer as a machine. "The study of people and the study of what they learn and think is inseparable." (Papert, describing ideology of both McCulloch and Piaget, in MindStorms)

Hobbyists and computer enthusiasts have a concept of the computer as a machine and therefore a desire to mess around with it's functionality. My father is one of these enthusiasts, and in his spare time he likes to add internal devices, reinstall operating systems, or fuss with his scanner and digital camera. His approach is methodical and calculated. He is fearful of damaging the computer, so he reads the manual and instruction booklets with fervor. His progress is slow, precise, and safe. Adults like my father have the confidence to open up machines and toy with their controlling mechanisms, but they often have no space in which to do so with complete freedom or lack of fear. An educational tool like the ones built for children would be a fabulous way to help my father and other adults delve into more complex ideas about computation and programming. Yet no appropriate tool exists.

Educational tools generally available to people of any age, but adults are less likely to use the programs because of age-related taste differences, irrelevance to their occupation, inflexible schedules, or the perceived need (or lack thereof) of such tools and knowledge in their life. No clear line exists between the output of these programs and the work adults need to accomplish in their career or with their family. There is little motivation for adults to conceptualize useful projects to build with computational devices, and part of this is because the appropriate tools do not exist for anyone without an engineering degree to use. Computationally educational programs such as MicroWorlds and Squeak are associated with children's toys. "Real work" can't be accomplished with a child's video game or play calculator, let alone their educational computer program. An attorney, for instance, would probably not feel comfortable bringing a Lego robot to the office when they are expected to behave professionally with clients. How, then, are adults supposed to learn about and use embedded devices if there is no useful resource for them to turn? Its infeasible to expect them to learn everything in a college classroom, and without a peer familiar with the work to guide them, the task is daunting enough to be avoided altogether.

For my final project I chose to analyze a program I am writing for my thesis. The system is a web based application that lets users program embeddable hardware devices, such as the Rabbit2000 microprocessor or PIC chip. I named it Oats, which stands for "Open, Accessible Teaching System." It could also be called Floats because Oats uses fuzzy logic reasoning rather than procedural programming to control the devices. The system is intended to provide a gateway for the inexperienced to work on and learn about technical projects. With Oats, users can explore and learn about hardware without being burdened by the many technical road blocks that accumulate during the use of current environments and processes.

In the context of my thesis, Oats will be used by fashion designers interested in constructing computational garments, but who don't have the background to build the technical components of such projects. For this paper, however, I constructed a generic form of the system in order to focus on the experience of programming with fuzzy logic. I wanted to avoid introducing conceptual ideas about computational garment design and instead explore the thinking process behind users while they interact with the system. I found two adults with limited technical background and guided them through a task where they used Oats to program a small, simulated hardware device.

In conjunction with the interviews, I analyzed different components of the software in light of the discussions we've been having in class. The discussions I focus on are about Multiple Styles of Thinking (October 22) and Computational Literacy (November 19). I came to the conclusion that technically inexperienced adults are quite scared of breaking their computers or doing something "wrong." Therefore, they are extremely hesitant to explore new programs or test ideas that confuse them. There is little incentive for adults to explore computational worlds, to build computational projects, or to conceptualize uses for technology without a direct correlation to their immediate life and work. The women I interviewed found Oats to be inviting and intuitive as a tool to accomplish the tasks I gave them. As the designer of the system, I know that it lacks immediate functionality outside its current simulation stage. However, it is successful as a testing space for observing adults as they approach, experience, and understand the process of fuzzy logic programming.

Constructionist ideology resonates with my desire for computational designers to have the skills and access to build any component of their work, for them to learn about their relationship to technology and to develop depth in their projects as a result of exploration during the development process. "Computers, like finger paint and beads, should be used as a "material" for making things," (Resnick, 2001).

Seymour Papert is interested in the kinds of computational models that lead to better thinking about powerful development processes (Papert, 1980) and in my work I am trying to construct such a model for hardware development by artists and designers. My research in embedded system design has been influenced and informed by many existing educational programs available on the market and within academic research settings, as well as the collection of papers and books I've read for this class. These include papers by Seymour Papert, Mitchel Resnick, Sherry Turkle, Alan Kay, and Evelyn Fox Keller, to name a few.

Concepts about transparency and the aesthetics of science are important to my work as well. The target audience for my work will be interested in building garments and finding the beauty in computational components. The instruments they use should reflect these aesthetic standards. As Resnick states in his paper, *Beyond Black Boxes: Bringing Transparency and Aesthetics Back to Scientific Investigation*,

> "The merits of the instrument-building tradition go beyond the immediate needs of research. Indeed, one element of that tradition is a design philosophy that emphasizes elegance and beauty in the material objects of scientific work.

Several systems have been designed by John Maeda, Ben Fry, and other students in the Aesthetics + Computation Group at the MIT Media Laboratory that focus on the aesthetic merits of scientific work. Design By Numbers (DBN), and Proce55ing are two systems that introduce basic ideas of computer programming within the context of graphic design. In DBN, dots, lines, and fields are drawn using computational concepts like iteration, repetition, variables, and conditional statements. Proce55ing is a learning program and environment for creating systems in JAVA with real time three-dimensional graphics, color, and other features that DBN lacked. The spirit of Proce55ing is to act as an electronic sketchbook where people can learn the fundamentals of computer programming within the context of the electronic arts (Reas, 2002).

> "It is (Maeda's) belief that the quality of media art and design can only improve through establishing educational infrastructure in arts and technology schools that create strong, cross-disciplinary individuals" (Aesthetics + Computation Group, 2002).

DBN and Proce55ing are powerful educational tools designed for adults, but they only allow users to create programs that manifest themselves on the screen. With Proce55ing, they can create beautiful graphics that are both interactive and dynamic. Other systems such as Boxer and Squeak are also limited to the screen. Squeak is a program designed for children to express ideas about math and science. "Squeak is an idea processor for children of all ages" and on the website it declares that its threshold is set low enough for five year olds (Squeakland, 2002). With Squeak, users can build computational systems. Boxer is a computational medium based on a literacy model and designed for building screen based tools with ease.

In each of these programs, users are unable to extend the concepts into the physical realm of hardware control. Hardware control is instead explored through educational tools created by members of the Lifelong Kindergarten Group (LLK) at the MIT Media Laboratory. LLK created Crickets, which are tiny computers that control two motors and receive information from two sensors. They have the ability to communicate with infrared. The blocks can be used for robotics or other investigations, such as body-monitoring and data collecting. Crickets are controlled using a dialect of the Logo programming language, called LogoBlocks. LogoBlocks is a procedural language that includes constructs like if, repeat, and loop, among others (Lifelong Kindergarten Group, 2002).

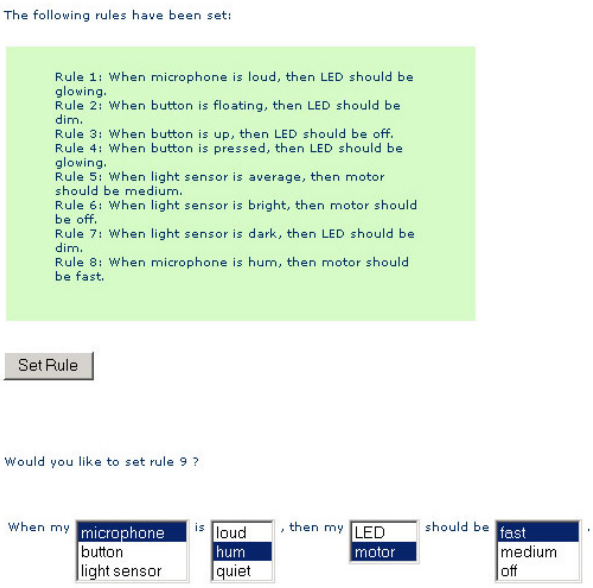
In addition to Crickets, my colleagues Justin Manor, Simon Greenwold, and I created a programming environment and language called Nylon that lets users write the controlling code for programmable devices in addition to developing dynamic graphics. Nylon builds off DBN in that it was designed for artists and visual designers. Nylon, LogoBlocks, and Crickets are each very powerful and are used in a variety of educational settings, however they are based on procedural languages and follow traditional models of computation. For my work, I want to move one step further and develop a system that doesn't rely on procedures and algorithms, but instead upon the structures of natural language and fuzzy logic calculations.

All of the work has provided the framework upon which I am building Oats. Without the pedagogy of such a varied collection of systems, it would be difficult to understand and discuss the intricate relationships and experiences that adults and children have with computational systems. These programs provide my work with a basis to expand upon, both conceptually as well as in execution.

The mathematics of fuzzy logic control the behavior and establish the basis for the non-procedural language used in Oats. Fuzzy logic was designed to model the uncertainties of natural language and thus works well as a methodology upon which to build powerful and intuitive programming tools. Fuzzy logic is a form of boolean logic that is capable of handling partial truths. Instead of assigning completely true or completely false values to elements, they fall in a subset between two absolute ends. A mapping from one set to another is not discrete. Subsets are described by membership functions, which map the degrees to which elements belong to a set.

Users of Oats navigate through a website to create the programs, establishing rules for how their fuzzy device is expected to behave given certain circumstances, or inputs. The collection of rules is mathematically analyzed in order to determine the actual behavior of their device. Once input values are mapped to a membership value, they are processed mathematically and compared against values set by other rules. After each rule has been taken into consideration, a discrete output value is obtained through min-max comparisons and a centroid calculation, the final steps in the fuzzy logic process.

Oats consists of a website, programming process, and software simulation that illustrates how hardware devices behave. The next stage of development following this paper is to go beyond simulation and use set up real devices that run using the fuzzy logic code produced by Oats. Meanwhile, the software simulation is designed so that users can manipulate input values and visualize the resulting behavior of the devices that will eventually run the fuzzy logic program. The simulation was generalized for this project to focus on a scenario where users control a robotic car. The car has a motor with variable speed and headlights with variable brightness. It also has a sense of the surrounding environment because it contains a light sensor, microphone, and slide switch.



In order to test Oats, I met and interviewed two women who have little to no experience building hardware or programming computers. I chose these women in order to gauge how the system was perceived and experienced by non-technical adults. My interview subjects went through identical processes to experience the program. Using the web interface, they first answered a series of questions about their educational background, technological background, interests, exposure to com-

puters, and communication. Then they read through the scenario I set up for them. The scenario described the small design project they were about to undertake, going into details about available materials, such as the controller they were using and the input and output devices they could use. Next, the women stepped through the site and created their fuzzy programs. This involved naming their inputs and outputs and describing the three states each could reach. Next they defined their collection of rules that determines how the car should behave. Finally, they clicked on "Simulate," which began the fuzzy processing simulation and allowed them to see their small car in action. I let them play with the car and interface for as long as they were interested before we moved on to further discussion.

Before starting this paper, I was curious about what the adults already knew about computation, programming, and the internal structure of machines. I was curious about their abilities - whether they'd ever thought about using or building computational devices, and whether they knew the steps involved in constructing projects with computational objects. I spoke with them generally about how they used and perceived computers or computational objects on a day to day basis. Then, after they used Oats, I pondered whether it helped users conceptualize the functionality of hardware devices and programming languages. Did it generate new ways of thinking they hadn't experienced before?

I also questioned whether the fuzzy process follows conventional "accepted" methods of control and design, and what this means for usability and credentials. I wondered how adults felt emotionally when approached with the task of writing a computer program. Does Oats fall short or exceed their expectations in any way? Did it help initiate powerful ideas? I asked myself all these questions and many more.

I wanted to see how adult minds, generally unexposed to programming tasks, interacted with the Fuzzy program, digested the ideas, and executed the goals set out for them. Therefore I chose two women, Heather and Laura, to interview because they have very little experience in the domain of computer engineering. The first of the women, Heather Casey, is a social worker in Boston. She has a bachelors and a masters degrees in social work, and has worked for the last two years at an early intervention program located in one of the poorer neighborhoods of Boston. Her job consists of working with children aged 0 to 3 years who are diagnosed with developmental delays. She individually spends time each week with the children, conducting exercises designed to bring the children up to speed with others their age. Most delays are manifested through language, mobility, cognition, and social behavior. In addition, Heather runs several groups through her agency, where she works with children and family members in a group setting. The groups are designed to create a community and establish trust amongst other people experiencing similar problems and frustrations.

When Heather has the time, she enjoys engaging in projects like sewing and knitting. Computers are not a prominent component in Heather's daily activities, however she owns a desktop computer which resides at home. She uses the computer several times a week, primarily for checking email, shopping, and visiting websites on the internet. Heather used to use her machine frequently while still in school, primarily to write papers and create reports for her classes and research. At Heather's agency she shares a computer with several other colleagues. She usually does not have time to use the

machine at work, unless it involves obtaining paperwork or conducting research for a client in need of housing, schooling, or other social services.

The second interview subject, Laura Davis, is a post-doctoral researcher at a medical research facility in downtown Boston. Laura moved to Boston from England just over a year ago. She was still working on her Ph.D. when she arrived in the United States, and thus she completed the last stages of her thesis and defense work remotely. Last winter she hand delivered a draft of her thesis document to her committee while visiting home for the winter holidays, and this spring she travelled back to England to defend her thesis and finish her degree. When asked if she used computers and email to help exchange revisions of her work with her committee overseas, she surprisingly said no. Before she left England, her committee members supplied her with an outline of what they expected of her work. This outline became her primary guideline while she worked independently.

As a biological researcher, Laura spends a lot of time at her lab growing and working with cells. The lab is equipped with digital cameras designed for use in conjunction with the microscopes and other high-precision instruments. Laura recently bought herself a laptop so that she could complete many of the tasks required of her research, such as writing papers, putting together presentations, and analyzing data. Laura primarily uses her computer for word processing, spreadsheets, checking email, and managing her personal finances. In addition, she uses digital photography to record the developmental stages of her cells, so Laura is familiar with paint and imaging programs, like Adobe Photoshop and the ones included with the cameras.

Computational objects are an integral part of adult lives. Palm pilots, electric toothbrushes, and cellphones are commonplace among professional adults. These objects are driven by small microcontrollers, sensors, and displays, just as bricks and programmable objects used with the educational tools are constructed.

The idea of programming or getting inside electronic devices to learn about and control their behavior makes many adults anxious and nervous. Through my experience teaching workshops on computational design and also showing my own parents and siblings the work I do as a masters student at the Media Laboratory, I have seen that adults can get excited about programming when someone is with them to hold their hand and explain what is going on. However several people have admitted that they would not attempt such projects on their own. They recognize the power of computation and their dependency on computational objects, but the infrastructure of such devices is otherworldly. The "guts" of their machines occupy a space they dare not tread.

Before showing Heather and Laura Oats, I walked each of them through a series of questions to gauge how much or how little they knew about hardware and computational elements. With each question it became clear that neither of the women knew much about this subject, and my progression of questions proceeded to make each one feel a little uncomfortable and slightly inadequate despite being educated adults. I reassured them that they were not expected to know what I was referring to because I was moving into knowledge that is highly specialized and reserved primarily for engineers. Part of my intentions in asking the questions was to gauge which computational concepts were "common knowledge" and which were not. Both women perked up when we started talking about the internet and different ways they use the computer to communicate. Once we entered into a computational realm that was familiar, they regained confidence and became more excited about the discussion.

Ideas about computer literacy have changed, and it is no longer clear what we need to know or should know in order to become masters of our technology (Turkle, 1996). Heather and Laura's perceptions of computers does not include the complex algorithms, logic gates, transistors, or binary machine code that enables the computers to run. They use the machines without understanding how the machines work underneath the surface. Concepts used in programming or hardware design, such as iteration or the execution of algorithms, are foreign to them, as they are to many adults.

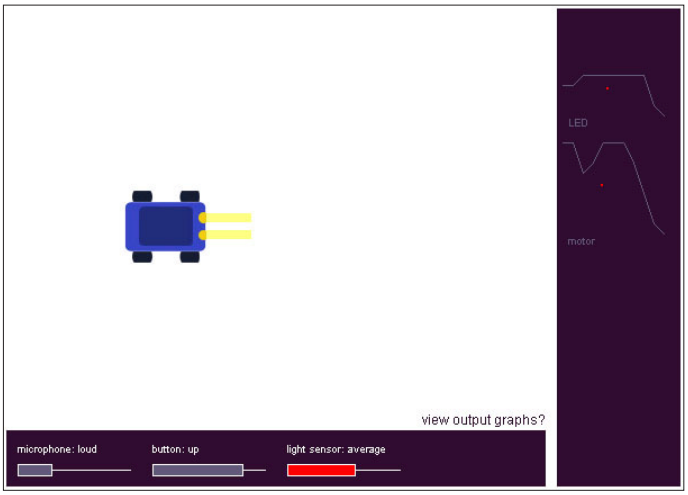
Heather, Laura, and many adults learn to understand their computer through exploration and interaction. Some adults enjoy reading manuals to learn how to use a computer program. Others, like Laura, operate on a need-only basis. Laura learns about the functionality of her machine only as she stumbles upon new tasks she must accomplish with it. Despite using a computer at work or at home, neither Laura nor Heather think about their machine as a calculated and methodical robot. They do

not visualize its behavior in terms of data flow mapped by procedures, algorithms, and variables. They are not exposed to the ordered structure, the synchronized gates, or verbose chunks of code that lie within. It's not that they are incapable of knowing this, they just don't have the need, or else they are scared to discover it for themselves. Heather and Laura each repeatedly told me, "I don't know anything about computers" despite using them frequently and owning personal machines. The question should be asked then, when are such ideas and knowledge about the process of computation necessary and useful? What would entice an adult to learn more about technology or to want to build projects using programmable devices and computational elements? It is not important for them to know how their computers work in order to use them effectively. When would they find it useful to understand computational concepts?

Each of these women had successful academic careers and slightly higher than average mathematics educations. Heather does not use math or science directly in her social work career, but she took college level Calculus and considered herself a strong math student while getting her undergraduate degree. Laura is a science researcher and uses math both directly and indirectly in her work.

Heather took a computer class in 1988 at her middle school in Florida, when she was in the eighth grade. In the class, they were given the task of executing rudimentary programs in what, we think, was the BASIC language, although this is unclear. She remembers the task as "making colored lines go across the screen" and she had difficulty remembering any other details about what they did. When asked about this experience, Heather was uncertain if the class "counted" as programming experience. She was hesitant to bring it up or describe it to me in detail because it occurred so long ago and existed as a vague memory in her mind. She remembered that it was fun, but didn't feel that it changed they way she thinks as an adult or that the work was useful in her current life.

Aside from Heather's middle school computer class, neither of the women had experience programming either software or hardware, nor did they use computers for tasks outside email, commerce, and the others previously listed. Laura's new laptop is still missing a few programs she'd like to use for her work. We started to discuss what other software programs she wanted to have on the computer, what she would use them for, and how she would install the software on her machine. Laura said she had a friend who could get the programs and install them for her. Through her exasperated facial expressions, it was apparent that



A screenshot from the Oats simulation.

installing software was an alien process to Laura, one in which she is very hesitant to get involved in. Laura told me that she thinks installations are probably quite easy once someone knows what they're doing. She admits she could probably install software herself if someone showed her how to do it. But she was also adamant that she didn't want or didn't need to learn. The task required a level of computer literacy that she was not willing to reach. She had friends she trusted to do it for her and do it correctly.

After working with Oats, Heather and Laura were both surprised that they had actually programmed the car on their own. Heather felt that the format was more natural than what she understood programming environments to be like, and so it became easier to comprehend what she was doing. She thought that more people would be able to use it because of the ease. She liked that it didn't crash or take a lot of time to use. Once she worked through some initial confusion, she really started to have fun and wanted to add more components and behaviors to the car, such as turning or reversing directions. Through the simulation and the rules she established, Heather really started to understand how the different components related to the rules she established. She became interested in the types of sensors and inputs available for programmable devices. She asked if there were temperature sensors that would open the car's sunroof or control its air conditioner. She also asked whether it was possible to determine how near other objects were, so the car wouldn't crash. She thought it would be fun to make stuff she could use with the kids at her job, so they could understand cause and effect.

> "I'd like to make it so that when I talk really loud the car moves around, and the kids would know if I'm angry or not," she said. "I think it would be really helpful to teach them about cause and effect. They like lights and noises and it would be cool to make stuff for them to play with and learn."

On the floor in Heather's room, I noticed that she had an array of holiday ornaments that she was sewing as gifts for her colleagues at work. I prodded her to see if she'd be interested in building computational stuff for her colleagues, and she laughed at the idea of making squishy candy cane ornaments that blinked or played carols. She said she would definitely be interested to learn how to use motors and lights because she didn't know how to use them.

Laura, on the other hand, seemed very hesitant to call what she had done "programming," even though she'd set up the inputs and outputs and defined the rules herself. She played with the simulation briefly and saw the relationship between input values and her car as it processed the rules. She was not interested in looking at the output graphs generated by the fuzzy logic calculations, and didn't care to ask questions about what they meant or how they related to her car. The entire system didn't fit with her model of programming as a complicated and impossible task, and so she resisted calling it programming. She was sure of her role as someone who "knows very little about programming" and didn't seem to have a desire to explore outside that box, or to admit that she actually had.

Building computational objects seems to have little relation to the work Laura does at her lab or home. Therefore she didn't care to brainstorm about other uses for the system outside the task I laid out for her. It would be fun to let Laura play with programs like MicroWorlds or StarLogo, programs with more relation to her work as a biologist. Although the physicality of the work I'm doing does not illicit inspiration or evoke new ideas about computation in Laura, my guess is that programs

designed for simulating distributed systems and cellular bodies would be received very differently. One thing she appreciated that the system was in a format that she could access easily. She finds that her favorite websites are organized clearly and contain links that work and follow through the site.

It is clear from the discussions with Heather and Laura about their experience and the things they learned from the program that adults need to see constructive, direct uses for computational systems in their careers and daily activities in order for them to get excited about and interested in using educational tools. Alan Kay has previously expressed these same ideas about children,

> "What really seems to be the case is that children are willing to go to any lengths to learn very difficult things and endure almost an endless succession of "failures" in the process if they have a sense that the activity is an integral part of their culture."

The strength of such an idea should not only be limited to young kids. It resonates throughout my work with Heather, Laura, and I imagine with other adults.

Observing the women as they programmed with Oats uncovered particular behaviors about how they approach problems, explore spaces, and generate solutions. Heather was much more playful than Laura with the system, and she had questions about how the underlying mathematics worked in addition to how the programmable devices behaved. Laura executed the task perfectly and quickly, but was less playful or curious about how it was actually working or how her rules affected the system. Heather found a relationship between the task, its potential uses, and her job as a social worker. Laura, on the other hand, found no direct relation to the program with her current career.

Initially, the program caused anxiety in Heather. Both women were curious and excited before starting the design task, and Heather particularly admitted she was a little nervous. Heather told me that she gets nervous when she tries most technically involved activities. With Oats, however, she quickly figured out how it worked, so it became fun. The consistent format helped her through the process and eliminated much of the confusion she had at the onset. She suggested an example in the beginning would help people see how the interface was set up before they jumped into it. Laura, on the other hand, was initially curious about the project and less nervous about what she was expected to do. However, when she finished she admitted that she felt very anxious about what she was doing throughout the process, even though it stepped her through it in a clear and intuitive manner.

Laura did not watch the output graphs generated by the fuzzy logic math with any interest, rather she seemed to shrug them aside because she didn't understand them. She didn't care to know how the fuzzy logic related to the rules she wrote or the input and outputs on her car.

Heather, on the other hand, wanted to know what the graphs meant and why they were changing. She liked that she could see the heavier weights of the graphs shifting as she changed the values of her inputs. She understood visually how the rules she made changed the behavior, because she could see this in the graphs. This got her exited because she felt she had really affected the system, but it was not too detailed that she was overwhelmed of confused. "I like that you did all the really hard stuff for me."

Heather was interesting to observe while construction her rules because she flew with ease through the first six rules that connected two of the inputs with the two outputs. She initially had a one-to-one mapping between input and output. She paused when she got to the third input because it was no longer clear to her what it should control or how it could fit in to her car's control system. She hesitated before establishing rules that connected the third input to either of the car's outputs. She gave me a quizzical look and "wondered if she could do that." She didn't really know what to expect from her car at this point, and this idea both excited her and scared her. I did not tell her to stop or scold her for trying this, so she assumed it was okay and continued.

Later I asked her whether she would have kept the rules had I not been there while she was exploring. She said that most likely she would have not kept them at first, but once she became comfortable with her car's behavior, she might add them in to see what would happen.

In the brief moment where Heather questioned how her car would react with two different inputs controlled the same thing, Heather learned an important rule about computational processes. She exposed herself to errors and pondered what it would take to make a program "break." The fuzzy logic safeguards certain breaking points from happening because of the nature of fuzzy logic, but it does not mean that users of fuzzy devices will never learn about or understand the limits of a program, as Heather showed me during her Oats experience.

In addition, Heather thought about the car as though it existed in a real space. She described the inputs as "changes in the room." By situating the car in an environment within her mind, she enabled herself to get more emotionally involved with the project than Laura. Many of the sensors she wanted to make for the car involved reading data about the surrounding space or even about the behavior of people. For instance, at one point we were talking about her job, and she asked me how she could make a "bullshit meter" to tell when people weren't giving her the true story. She laughed at this because she thought the idea was silly, but an underlying seriousness resided in her voice.

Laura inquired about more control features for the car, such as steering capabilities, but she did not have a mental relationship between it and any physical environment. She told me before the exercise that computers can communicate using devices like touch screens or the mouse, but she did not connect these capabilities with her own project.

After spending time with Heather and Laura, I was curious about using Oats with adults who have high levels of programming experience and a very fluid and deep understanding of computers, but who do not use hardware or have an equivalent amount of experience working in hardware as they do in software. Therefore I engaged in an informal discussion over coffee with two graduate students who are working on thesis projects that involve many hours of programming and computer time. Stan Port and Jamie Wood studied computer science and electrical engineering as undergraduates at the Massachusetts Institute of Technology. Stan is currently working on his master's degree at the MIT Media Laboratory, and Jamie is working on her M.Eng. degree with professors at the Whitehead institute. Neither Stan nor Jamie feel comfortable using hardware, although they have different opinions of hardware as a medium.

Stan loves programming in software because of the ease of implementation, the relatively free cost, and the speed at which he can develop complex programs. He has forayed into hardware on occasion, but found it extremely frustrating to pour over catalogs, scratch his head about broken components, and repetitively build the same circuit over and over. He had no idea how to go about starting a hardware project without the help of his colleagues. He explained it to me as an "indescribable phobia of hardware."

Jamie, on the other hand, has a desire to use hardware because she finds it aesthetically beautiful and much more powerful than the screen-based programs she used to create in software. She likes the idea of being able to work in a three-dimensional medium rather than building programs that can so easily get ignored or lost. However Jamie hasn't the faintest idea where to start such a project. She explained that, if given a circuit diagram, she can build it perfectly because she knows all the components and how to put them together, but she wouldn't have any notion of what the circuit did on her own. "I can follow the directions if someone gives them to me, but I want to be able to just put things together and know they'll work. I want big, tangible pieces that I can just play with." Jamie expressed a need to separate hardware from the personal computer altogether.

Speaking to these two helped clarify where the next stages of the Oats development needs to focus. Heather and Laura helped me recognize that Oats is fairly successful at making the software and programming component of project development open and accessible to people with no programming background. The fuzzy logic calculations successfully implemented control code that allowed users to program using natural language rather than procedural languages. It created an abstraction that was high enough to be useful to beginners, and transparent enough to encourage questions and curiosity about how things behave. Although Oats is not the best example of a tool that will excite all users, it is a good tool for people like Heather (and hopefully fashion designers) to start thinking about and creating computational projects. The next step is to create open and accessible modules for designers to build hardware circuitry, to explore electrical engineering concepts, and to feel like they can grasp how their projects behave on many levels beneath the surface.

Bibliography

Aesthetics + Computation Group. Design By Numbers website, http://dbn.media.mit.edu/. Massachusetts Institute of Technology, Media Laboratory. Cambridge, MA. 2002.

diSessa, Andrea A., Lay, Edward H., The Berkeley Boxer Project website. http://dewey.soe.berkeley.edu/boxer/. 2002.

Kay, Alan. Powerful Ideas Need Love Too! Written remarks to U.S. Congressional Committees. October 1995.

Keller, Evelyn Fox. Reflections on Gender and Science. Yale University Press. 1985.

Maeda, J., Sakai, Y., Sawada, Y., Komuro, R., "Fuzzy Rules As a Programming Medium for Children." Proceedings of 2nd International Conference on Fuzzy Logic and Neural Networks. pp. 709 - 715.

Mamdami, E. "Application of Fuzzy Algorithms for Control of Simple Dynamic Plant." Proc. IEEE 1974. Vol. 121, No. 12. pp. 1585-1588.

Papert, Seymour. Mindstorms: Children, Computers, and Powerful Ideas. Basic Books. 1980.

Reas, C., Fry, B. Proce55ing website. http://www.proce55ing.net/. 2002.

Resnick, M., Berg, R., and Eisenberg, M. Beyond Black Boxes: Bringing Transparency and Aesthetics Back to Scientific Investigation. Journal of the Learning Sciences, vol. 9, no. 1, pp. 7-30. 2000.

Rosenschein, Stanley. "Artificial Agent Architecture." MITECS: The MIT Encyclopedia of the Cognitive Sciences. http://cognet.mit.edu/MITECS/Entry/rosenschein.

Turkle, Sherry. "Seeing Through Computers: Education in a Culture of Simulation." The American Prospect. 1996.

Squeakland website. http://www.squeakland.org/. 2002.

Interviews

Heather Casey, 70 minutes
Laura Davis, 50 minutes
Stan Port and Jamie Wood, 30 minutes