5-2014

Engaging Alternative High School Students Through the Design, Development, and Crafting of Computationally Enhanced Pets

Maneksha Katrine DuMont
Utah State University

ENGAGING ALTERNATIVE HIGH SCHOOL STUDENTS THROUGH THE DESIGN, DEVELOPMENT, AND CRAFTING OF COMPUTATIONALLY ENHANCED PETS

by

Maneksha Katrine DuMont

A dissertation submitted in partial fulfillment
of the requirements for the degree

of

DOCTOR OF PHILOSOPHY

in

Instructional Technology and Learning Sciences

Approved:

Victor Lee Major Professor	Mimi Recker Committee Member
Brett Shelton Committee Member	Deborah Fields Committee Member
Andrew Walker Committee Member	David Smellie Committee Member

Mark McLellan
Vice President for Research and
Dean of the School of Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah
2013

ABSTRACT

Engaging Alternative High School Students Through the Design, Development, and Crafting of Computationally Enhanced Pets

by

Maneksha Katrine DuMont

Utah State University, 2013

Major Professor: Dr. Victor R. Lee
Department: Instructional Technology and Learning Sciences

Hybrid design technologies, a combination of physical crafting, construction or art, and computing, have the potential to broaden participation in computing by appealing to youth through existing interests and hobbies. Expanding participation in computing is important because computational thinking, for example debugging, is a set of skills fundamental for success in our society. Youth can participate in and gain exposure to multiple disciplines with various hybrid design technologies. Yet alternative high school students, those labeled failing and been moved from the conventional school to a facility that focuses on building adult skills and remediated instruction, are not often the beneficiaries of innovative learning environments. There is reason to believe that these students could benefit from a new way of learning with new hybrid technologies including learning about debugging, art and craft, technology design, and aspects of computer programming.

This dissertation investigates whether a novel hybrid technology can provide alternative high school students with new forms of access to computation and encourage participation in debugging. This dissertation will serve as a multi-faceted report of one cycle of design, implementation, analysis, and refinement of a hybrid technology intervention with a diverse, oft ignored, and challenging population. In this project, students at an alternative high school worked to create interactive pets, similar to some commercially available, popular toys and then shared them with the community. The pets were virtual, existing on the computer screen, and tangible, existing in the physical world. Students worked predominantly by reusing and modifying existing programming code.

In the end, there were a number of encouraging results, such as observed instances of high engagement, success in dealing with programming bugs, and the connections some students made to computing and mistake making. There were also some areas in which the design and implementation could be improved for future iterations, namely through refinement of the activities and technologies to encompass a wider range of student interests, a more concentrated effort to cultivate a nurturing community of designers, and a more consistent fostering of motivation for and understanding of the final product and its intended audience.

(298 pages)

PUBLIC ABSTRACT

Maneksha Katrine DuMont
Ph.D. Instructional Technology and Learning Sciences

Engaging Alternative High School Students Through the Design, Development, and Crafting of Computationally Enhanced Pets

A new kind of technology combines making in the physical world with computing in the virtual world. These technologies are simultaneously physical and virtual, require the design of artifacts like computer programs and the crafting of physical components. Combining approaches encourage youth to participate in computing who might not otherwise be interested by harnessing existing hobbies and interests. Because fluency with computing is important to success in a 21st century society, increasing the ways in which young people can experience and connect to computing is important. Computational thinking, and with it the process of debugging computer code, is an example of the skills young people can learn.

This dissertation is a report of the design, implementation, analysis, and refinement of a hybrid media intervention in an alternative high school. Students designed, crafted, developed, and shared interactive pets. The pets were physical creations that users could interact with via an external microprocessing board and corresponding computer program. The investigation included questions concerning whether students could complete the project, how students engaged with the design tasks, how students participated in addressing programming errors (bugs), and whether students exhibited elements of empowerment. The results suggest youth were engaged, successful in addressing bugs, and some students made personal connections to computing and mistake making.

The empirical work in this dissertation contributes to theory related to design of constructionist learning environments and student collaboration with hybrid media. Limitations of the project and next steps for research are also discussed.

ACKNOWLEDGMENTS

First I would like to thank my committee for continuously pushing me to think more deeply about and attend more thoughtfully to my work and writing. I especially would like to extend sincere gratitude to my chair and advisor, Victor Lee, who took an extraordinary risk by taking me on as his student and then persevered through my cynicism, back talk, and revisions to the end, even when circumstances became, at times, nearly impossible. I will always be endeared to the ways in which you were able to guide me through this process and will continue look to you for much needed advice. Also, I am grateful for all the support and feedback from fellow students at the VITAL Collaborative. To the students at Winder Alternative School, don't ever let anyone tell you that you can't do it. I appreciate your willingness to join me on this crazy project and I learned a tremendous amount from all of you. None of this would have been possible without my friend and colleague at Winder as well as the dedicated and inspirational staff, administrators, and teachers there.

To my family and friends, thank you, thank you for not letting me quit and encouraging me unconditionally despite continuous claims of not knowing what I was doing and why it was taking me so long.

Finally, to Joel and Ewan, with whom everything is possible.

Maneksha DuMont

CONTENTS

Page

ABSTRACT iii

PUBLIC ABSTRACT v

ACKNOWLEDGMENTS vi

LIST OF TABLES xi

LIST OF FIGURES xiii

CHAPTER

1. INTRODUCTION 1

Rationale 6
A Path Forward 11
Research Goals/Questions 14
Outline for the Thesis 16

2. POPULATION, SETTING, AND THEORETICAL PERSPECTIVE 18

Context 18
Theoretical Perspective 24
Constructionism 25
Programming in Constructionist Environments 26
Studies in Constructionist Programming Environments 29
Hybrid Design Technologies 32
Examples of Hybrid Design Technologies 35
What Makes for a Constructionist-inspired Learning Environment? . 41

3. METHODOLOGY AND LEARNING ENVIRONMENT DESIGN 45

A New Kind of Hybrid Design Technology 45
Learning Environment Design 48
Three Principles for Learning Environments with Technology 49
Facilitating Connections 50

Developing a Supportive Culture 51
Promoting Creative Expression 53
Learning Activities 54
Planned Activities 56
Opener..... 56
Workshop..... 57
Round Table..... 59
Design Exhibit Night..... 59
Research Plan 62
Data Collection 62
Author's and Faculty Involvement 65
Sampling and Recruitment 66
Resultant Participant Numbers 69
Data Collection Instruments 70

4. HOW INDIVIDUAL STUDENTS ENGAGED WITH THE PROJECT 73

Narrative Construction 74
Design Versus Realization: Time 76
Tegan 81
Jamal 99
Carlos 115
Hybrid Design Technologies and Limitations on Engagement: Dino 123
Summary 128

5. HOW STUDENTS INTERACTED 129

Why Should We Care About Collaboration and How Was It Encouraged? 130
Operationalizing Collaboration 131
Collaboration Analysis 135
Collaboration Results 140
Accounting for Absence 144
Why Else Student Collaboration Was Limited 145
Expertise And Interest Lead To Divided Roles and Goals 147
The Difficulty Of The Task Prevented Students from Taking Up Collaborative Opportunities 150
Modularity: A Population Characteristic 153
A Proprietary, Wary Population 154
An Emotional Technology 158

Conclusions About Student Interactions 163

6. DEBUGGING 165

Analysis Methods 167
Operationalizing Bugs in Computational Crafts..... 170
Categories of Bugs 172
What Did Debugging Look Like? 184
Accounting for Overall Numbers of Bugs 188
Bug Fates by Group 190
Individual Student Groups & Bug Fates 192
Modes of Debugging: Student Problem Solving Activities 195
Minimal Support Versus Coaching 199
Implemented Direct Solution Idea 204
Tinkered 204
Used Brute Force Repeated Failure 205
No Strategy 205
Deleted Buggy Code 206
Gave Up 206
Post Assessment on Bugs 207
The Debugging Assessment Task Explained 208
Results from the Debugging Task 211
Conclusions About Bugs 216

7. STUDENT IMPRESSIONS OF THE PROJECT 217

Ideas About Making Mistakes 218
Ideas About Computing 225
Overall Impressions of the Project 227
Conclusion 229

8. CONCLUSION 232

Theoretical Contribution …..... 235
Relative Notions of Sharing 235
The Hybridity Continuum 236
Reflections on the Results 240
Limitations and Steps For Improvement 243
Next Steps and Final Thoughts 249

REFERENCES 251

APPENDICES 262

Appendix A: Sample Lesson Plan 263
Appendix B: Pre and Post Survey 266
Appendix C: Field Notes Sample 272
Appendix D: Debug Journals 278
VITA 281

LIST OF TABLES

Table Page

1 Digiblepets Intended Activity Sequence, Weeks 1 and 2 60

2 Digiblepets Intended Activity Sequence, Weeks 3 and 4 61

3 Student Projects 76

4 Digiblepets Realized Activity Sequence. Items in red were omitted from the original planned activity sequence and items in blue were added to the new activity sequence based on shorter time blocks 80

5 Debugging Task Days: Bug tasks students were given with corresponding programming concepts explored during the initial two workshops with a prototype DigiblePet (PicoBoard) and corresponding scratch program 83

6 Student Groups, Projects and Number of Bugs Encountered on the 4 Days Selected For Analysis. The student group for whom the day represented the biggest bug day is highlighted 139

7 Number of Collaborative Episodes, Total Collaborative Exchanges, Length of Workshop, and Percentage of Collaborative Exchanges Versus the Structured Day for Tegan, Rocky and Ted's Group 141

8 Minutes of Collaboration Time for Each Group During Workshop 1 and Then During the Combined Independent Workshops Numbered 4, 9, and 12 142

9 Student Design Teams and Roles 147

10 Tangible Bugs Encountered During Representative Workshop Days 175

11 Virtual Bugs Encountered During Representative Workshop Days 177

12 Summary of the Code Categories, Examples from the Data Corpus and Fixes Implemented 179

13 Debugging Modes, Counts, Frequencies, Student Group Employment of Modes and Percentage of Bugs Where Mode Was Employed By Group 198

14 The Debugging Task Assessment Items, Programming Concepts Covered and Text From the Worksheet 210

15 Students' Debugging Task Assessment Results. Students received two points for correct solutions and solutions with minimal support and one point for solutions with moderate support 212

16 Students' Survey Responses for Statement: "I Am Confident I Can Fix A Bug/Error When My Computer Code Isn't Working Right." 219

17 Students' Survey Responses to the Statement: "I Lean Best When I Have to Figure Out My Mistakes" 220

18 Students' Survey Responses to the Statement: "I Like Figuring Out How to Fix My Mistakes" 221

19 Students' Interview Responses to the Mistake-Related Interview Questions 222

20 Students' Interview Responses to Feelings About Computers 227

21 Students' Interview Responses to How They Felt About the Project Overall 230

LIST OF FIGURES

Figure		Page
1	A Zhu Zhu Pet. (www.zhuniverse.com).	4
2	An online game featuring Zhu Zhu Pets. (www.zhuniverse.com)	4
3	A Webkinz aadvark. At center is a picture of the physical pet, at top is the virtual character, and at bottom is the pet's ant hill thrill pit accessory. (www.webkinz.com).	5
4	A game from Webkinz World. The game features your pet and allows you to earn coin to buy pet accessories for your Webkinz. (www.webkinz.com).	5
5	A partial timeline of microprocessing boards designed for learning.	7
6	Winder Alternative High School's daily bell schedule. Taken from the school's website.	22
7	Scratch and its interface. Scratch has repositories for imported or created sounds and costumes. Users can incorporate many characters, known as sprites and program them via chunks of code found in the menu. The sprites then enact the programming code in the stage area when the program is run.	28
8	Topobo. (http://www.topobo.com). The large blue pieces are motors that can be programmed through motion. The yellow, green and red pieces attach to form a creature's structure.	36
9	A PicoBoard. The board features light, resistance and sound sensors, a button and a slider.	38
10	PicoCricket and an interactive lamp project using PicoCricket where the light changes colors based on sensor input. (http://images.businessweek.com/ss/09/12/1209_25_world_changing_products/19.htm).	39
11	Lilypad Arduino and a turn signal jacket project using Lilypad Arduino. (http://www.arduino.cc/en/Main/ArduinoBoardLilyPad, http://web.media.mit.edu/~leah/LilyPad/build.html)	40

12 How DigiblePets work. Users pet the fluffy pet on left, depressing the button on the embedded PicoBoard. The Scratch program interprets the button press and computer code translates it to cause the sprite on screen to roll over and bark out loud. 48

13 Cujo: A prototype DigiblePet. The pet was made out of fur and pipe cleaners with a cork body to hold its shape. The PicoBoard is embedded beneath..... 68

14 Research Design. 72

15 A reproduction of Tegan's background modification in Scratch. 86

16 Some of Tegan, Rocky, and Ted's monkey costumes. Each one was painted by hand onto the original monkey stock sprite. The costumes all refer to "Cujo" the original prototype pet because students were using the prototype project as a guide..... 89

17 Tegan, Rocky, and Ted's Scratch Program. Their monkey climbed the ladder and jumped on the bed. 90

18 The original prototype walking script. 90

19 Tegan and Rocky working in parallel on different components of the monkey project. Tegan works on the physical monkey and Rocky on the Scratch program. 95

20 Tegan's Monkey. 98

21 Jamal's Wild Thing walking in the woods. 104

22 Jamal's hand-painted tree in progress. 105

23 Jamal's walking code that never worked correctly. 105

24 Jamal using the paint editor to paint Nike 6.0s for his unicorn. 107

25 Jamal zooming in to fix his high top problem. 107

26 Jamal working on his tangible pet design, facing away from all the other students in the class. 108

27 Jamal's finished zebra with eye pointing downward and tail up, the PicoBoard in embedded beneath and not visible. 109

28 Jamal's hand painted zebra sprite. 110

29 Jamal's zebra doing a backflip in the dance club. 112

30 Carlos, Dino, and Maya's alien Scratch program. 117

31 Maya, working on the many-eyed alien, Carlos, touching the computer, and Dino, looking at the Scratch program. 121

32 The prototype Scratch code for debugging task #1 that Tegan and Rocky are engaged in. The collaborative exchange is considered as beginning with contribution #1 (from Rocky). 134

33 Collaboration between groups during the sampled workshops. All groups collaborated with Tegan, Rocky, and Ted's group. 143

34 Steph and Tabitha's final project. The unicorn on the left was Steph's physical pet design whereas the hippo on the right was Tabitha's virtual pet design. The head feathers are one visual aspect the two creatures, that are intended to be representations of the same pet, share. 146

35 Tabitha's two-costume approach to dancing. 151

36 Total bugs by category during representative workshop days. 177

37 Overall bugs per day by workshop group. 185

38 Number of individual bugs by student group per day. 187

39 The fates of all bugs encountered during independent design workshops by percentage. 191

40 Students' bug fates by group (in percentage). 193

41 Frequency of debugging strategies used by students. 197

42 The debugging assessment Scratch project. 209

CHAPTER 1

INTRODUCTION

In the 1960s, a new system emerged to help disconnected, vulnerable youth by providing an alternate education for students at-risk of dropping out of or being expelled from traditional schools (Lange & Sletten, 2002). This system involved the development of "alternative schools," a catch-all term used to describe a wide variety of educational programs separate from the conventional system. It was designed for students who typically have a history of limited academic achievement, disciplinary problems, and lack of engagement in school. Students can struggle in school for myriad reasons including learning disabilities, English language deficiencies, and chronic truancy due to behavioral or psychological disorders, pregnancy or parenting, incarceration, addiction, difficulties at home and/or full time employment responsibilities (Pang & Foley, 2006). To help attending youth satisfy high school graduation requirements, many alternative schools have increased autonomy within districts to focus on more individualized programs that center on discipline, structure, community building, developing trusted adult relationships and remediated general education, and sometimes also including adult or job skill building (Aron, 2003; Pang & Foley, 2006). Alternative schools support disenfranchised students in meeting state and national academic standards, yet most do not have access to essential facilities like science laboratories or computer labs (Pang & Foley, 2006). Proponents find the prospect of an equitable, non-conventional educational program enticing, however alternative schools are often perceived by the public as not much more than a holding place for problem students (Aron, 2003). Limited large-scale studies

provide evidence promoting the effectiveness of alternative schools and programs (Lange & Sletten, 2002).

Alternative schools are not typical spaces for testing radically new educational technologies because the demands placed on new technologies are already considered high, yet these technologies and encompassing design-based learning environments are thought of as powerful because they provide new forms of access and encourage many different ways of knowing (Eisenberg, 2003; Turkle & Papert, 1991). The latter point about new technology-enhanced learning environments is resonant with Constructionism, a theoretical and philosophical perspective that there may be many legitimate epistemologies and ways of tapping into individual's knowledge and engagement in service of learning (Papert, 1980; Turkle & Papert, 1991). In that regard, students who are initially seen as having difficulty can actually be seen as having potential for success so long as they are given the opportunities to engage with "powerful ideas" through expressive media and a supportive learning environment.

This project investigates whether novel hybrid technologies can provide alternative high school students with new forms of access to computation, promote debugging, and encourage different relationships to learning and mistake making, despite difficult circumstances. In this project, students at an alternative high school worked individually or in groups to create a new kind of digital pet, similar to some commercially available, popular toys. This digital pet was virtual, existing on the computer screen, and tangible, existing in the physical world. The pets were meant to piggyback on the popularity of recent toys of similar spirit like ZhuZhu Pets (see Figure 1 and Figure 2)

and Webkinz[1] (see Figure 3 and Figure 4) Those commercial toys combine physical stuffed animals with online personas. In the online world, pet owners can play games and interact with other pets. In the case of Webkinz, children can earn coins online to buy new accessories for their pet's virtual habitats. The pets students made in this project are similar in that they had both a physical body and online character. However, the pets designed in this project were intended to more closely integrate the physical and virtual. Interacting with the pet in the physical world was to cause things to happen in the virtual world[2]. The toy emphasis was intended to engage youth who may not necessarily relate to computation, but could be captivated by the process of designing and crafting a familiar type of computationally-enhanced and engrossing toy (as described by Turkle, 2005). For five weeks, students in this project, all novice programmers, designed, developed, programmed, and crafted their own virtual pets with interactive fluffy and fuzzy physical bodies to share with members of their school and local community, in an open-ended, semi-structured learning environment. Students explored the design cycle and computer programming, namely through modification/reuse of existing code and debugging, using the Scratch media-rich programming language (Maloney et al., 2004) coupled with PicoBoard microprocessing boards (see Rusk, Resnick, Berg & Pezalla-Granlund, 2008), along with supplementary craft, art and found materials.

---

[1] Launched in 2005, Webkinz was estimated to be worth $2 billion and had sold over 2 million toys by 2008. (http://www.wired.com/entertainment/theweb/magazine/16-11/st_webkinz)

[2] The pet's embedded logic board could be thought of as functioning like a multi-sensory mouse where pressing buttons, changing lighting, talking, moving a slider and/or engaging resistance sensors provides the computer program with inputs that can then be programmed to cause certain reactions on screen.

Figure 1. A Zhu Zhu Pet. (www.zhuniverse.com).

Figure 2. An online game featuring Zhu Zhu Pets. (www.zhuniverse.com)

Figure 3. A Webkinz aadvark. At center is a picture of the physical pet, at top is the virtual character, and at bottom is the pet's ant hill thrill pit accessory. (www.webkinz.com).

Figure 4. A game from Webkinz World. The game features your pet and allows you to earn coin to buy pet accessories for your Webkinz. (www.webkinz.com).

Rationale

A project such as this is possible in large part because of the increase in commercial availability of small inexpensive microprocessing boards, including but not limited to Arduino, Programmable Bricks, GoGoBoards etc. (See Figure 5). Microprocessing boards have galvanized the area of inexpensive tangible computation in the spirit of broadening participation in computation through physical design. Principles that foster this aim hinge on a physical construction component accompanied by the metaphors low floor (ease of use), wide walls (flexibility to support varied interests and intuitions), and high ceiling (robustness to allow for intellectual growth) (Maloney et al., 2004; Papert, 1980). Arduino and their more specific counterparts can include various sensors for processing the external world and can sometimes perform simple functions like activating motors and lights. These physical boards are simplified and low cost enough to begin to allow for researchers to further broaden participation in computing by moving from studies in out-of-school environments to studies within traditional classrooms with known constraints like time and resources (Buechley, Eisenberg, & Elumeze, 2007).

Young designers have used microprocessing boards to develop scientific instruments (Sipitakiat, Blikstein, & Cavallo, 2004; Resnick, 1998), themed robotics (Eisenberg, Elumeze, MacFerrin, & Buechley, 2009; Rusk, Resnick, Berg, & Pezalla-Granlund, 2008), e-textiles (Buechley et al., 2008), interactive miniature rooms (Meyers, LaMarche, & Eisenberg, 2010), and interactive paper art (Eisenberg et al., 2009). Early documentation in out-of-school contexts show youth can engage simultaneously in

multiple subject areas with these "hybrid" media technologies including art, craft, circuit design, industrial design, sewing, and computer programming (Rusk, Resnick, Berg, & Pezalla-Granlund, 2008; Kafai, Fields, & Searle, 2012).

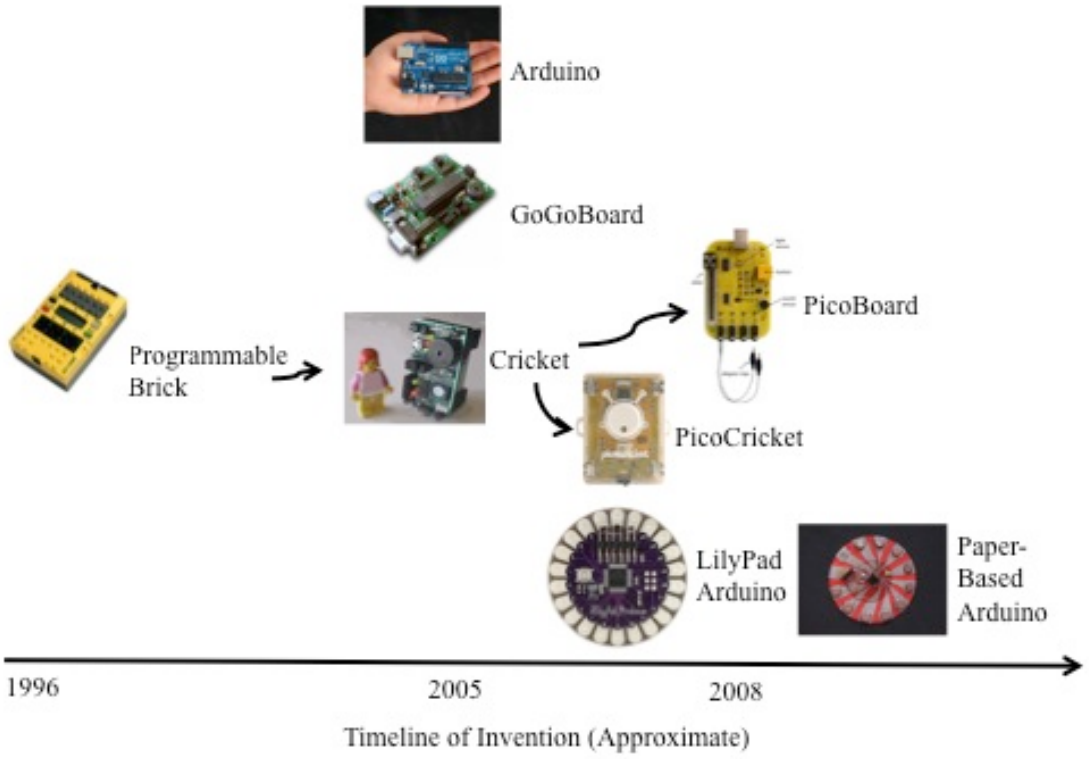

Figure 5. A partial timeline of microprocessing boards designed for learning.

The integration of multiple modalities, tangible and virtual, is designed to harness young people's emotions and desire for expression as a way to promote the development

of cognitive skills and encourage the perseverance needed for computer programming (Eisenberg et al., 2002). Computational crafts are a form of hybrid media technology that integrates hobbies, like painting, sewing or crafting, with computing. They are part of a growing *maker* or DIY movement that endeavors to use hybrid design technologies to capitalize on certain affordances, for instance, connecting to young people's interests, thus presenting young people with alternative ways to relate to computing, especially those who may not otherwise have an interest (DuMont & Fields, 2013; Kafai & Peppler, 2011).

Beyond supporting engagement and emotional connection, the relevance of hybrid technologies also stems from their potential to encourage young people to engage in computational thinking. Computational thinking has been characterized as a set of cognitive resources used to define, approach and solve problems with the aid of computers, but useful and relevant to many domains (Grover & Pea, 2013; Lu & Fletcher, 2009; National Research Council, 2010; Wing, 2006). These cognitive processes, termed *procedural thinking* (Papert, 1980), include activities like planning, modeling, systems thinking, algorithm building, testing, feedback, and debugging (NRC, 2010; Resnick, 1998; Wing, 2006). The advent of computational thinking has drastically altered fields like statistics, biology and the understanding of complex aggregate systems (Wilensky & Reisman, 2006) through computational modeling, simulating, and solving power (Wing, 2006) and can be useful in computer free contexts as varied as the teaching of journalism (NRC, 2010), and even when playing tabletop board games (Berland & Lee, 2011).

Providing ways for more young people to relate to and participate in computing is represented, in principle, in the growing national interest in computational thinking. Several recent prominent reports have made urgent calls toward the importance of developing computational thinking in all students (National Research Council, 2010; Wing, 2006). The National Research Council (2010) stated computational thinking is fundamental for individual success, as an outlet for self-expression and empowerment, and to advancing innovation in a technological society. Indeed, some have likened it to an essential 21st century literacy (diSessa, 2000). The conceptualization of computational thinking as a literacy suggests that the processes involved in computational reasoning are both ubiquitous and elemental. A literacy has the power to expand the ways individuals think and know (diSessa, 2000; Papert, 1980). The process of designing and developing videogames or software in novice programming environments, such as Logo or Scratch, or building robots to perform specific tasks all require computational thinking.

One common form of computational thinking appears in the process of *debugging*. Debugging is the process of locating and fixing errors that cause disparities between a programmer's intent and a program's output (Pea, 1983). Activities involved in debugging are critical for two reasons: 1. They are a fundamental part of the iterative design cycle prompting programmers to creatively solve problems and refine their thinking, and 2. They can allow individuals to develop new relationships to learning through the process of dealing with errors (Papert, 1980).

The positive impacts of technology design projects, including media or software design, robotics or construction, have been documented in a number of case studies (see

Harel & Papert, 1991; Kafai, 1996; Rusk et al., 2008). Yet, efforts to capture specific student outcomes or outcomes at scale have reported mixed findings. For example students learned aspects of circuitry and computer programming through e-textiles classes, but in some early instantiation students' overall programming knowledge decreased (Buechle et al., 2007). In another study, students learned a number of important science concepts in after-school robotics programs (Sullivan, 2008). However, in other studies, learners struggled to develop transferable thinking practices used to approach and solve problems, which is so pertinent to programming and design. For instance, after a full year learning Logo, students showed no gains over their non-programming language learning peers in programmatic thinking skills (Pea, Kurland, & Hawkins, 1985). Similarly, in a large-scale quantitative study, K-12 students did not improve in problem-solving ability after a class in Lego Mindstorms, a robotics kit for students (Hussain, Lindh, & Shukur, 2006). Promisingly however, in other research, students improved their debugging skills through a specific Logo debugging curriculum (Klahr & McCoy Carver, 1988). The studies, in some cases, involved years of supported student interventions and computer savvy and/or academically advanced student populations and schools, which are necessarily difficult to replicate with the broader population. A conclusion from these studies may be that the transfer research paradigm does not fully capture the changes that are involved in being able to better solve problems.

Along with supporting the development of cognitive skills there is a sense that enticing a broader swath of young people into computational domains through designing,

programming, and building artifacts may be also be empowering, in the sense that young people may develop greater understanding about aspects of learning as well as a new interest in computing. Although developing deep personal connections to learning and knowing is part of what designers and learning scientists consider an ideal product of students' participation in hybrid media design projects (Ackerman, 1996; Eisenberg et al., 2009; Papert, 1980; Turkle & Papert, 1991), limited research exists to support the claim that this sense of empowerment develops after participating in a media design project. While we may hope for students to feel confident, interested, and excited, it could be, for example, that the amount of time investment and the difficulties associated with building and programming lead students to feel frustrated and uninterested. With students who are placed in alternative schooling, those latter outcomes could potentially be more likely.

A Path Forward

As stated above, alternative high school students are not typically beneficiaries of educational research studies involving innovative technologies. Yet, hybrid technologies have potential for developing necessary thinking skills, promoting aspects of empowerment and appealing to a diverse population who may not otherwise relate to computing. Therefore, it is reasonable to anticipate that students from an alternative high school may ultimately benefit from participating in a design project with hybrid technologies. Given the lack of empirical work done already with this population, a way to approach this hypothesis is to design and implement a project with a new hybrid technology in an authentic alternative high school. The environment would necessarily be

designed to welcome youth with diverse, non-computing interests, and provide necessary supports. At the same time, it would largely provide young people time and space to develop their own ideas in their own ways. Ideally, it would encourage deep, connected engagement with the design process and programming. The research paradigm known as "design research" can be suited to this purpose, with its focus on iterative implementations of designed interventions in messy, real-life classrooms rife with constraints and independent variables. Design researchers aim to conceptualize new learning designs with an emphasis on "principles derived from prior research" (Collins, Joseph, & Bielaczyc, 2004, p.15) and implement them in an intact learning setting.

The benefits of design research include greater capacity for deep understanding of a particular instance of a designed intervention within an authentic learning setting. However design research is frequently highly variable and intensive for a researcher or research team. It often requires re-design efforts in the midst of implementation, and success hinges on the ability to continuously shift and modify plans according to student needs. Furthermore, researchers who go on this path must be willing to change planned analyses based on unanticipated observations within and outcomes of the study (Brown, 1992). Learning in authentic settings is challenging to study because it is social, cultural, and personal and involves students' educational histories and interactions between students, teachers and established classroom norms. Students can be absent, reticent, refusing, ill-prepared. Teachers and facilitators must be on their toes, know when to intervene, follow-up, provide guidance or leave things be while collecting and immersing themselves in copious video and observational data during the study itself. The researcher

can often be highly involved in the intervention, thus allowing her to have first hand knowledge of all that occurs. However, that involvement can also limit researcher objectivity. Data collected in design research studies are abundant. They can be both qualitative and quantitative, but small participant sizes can limit statistical rigor and generalizable robustness (Brown, 1992). At the same time, attending to so many different issues of implementation can make it hard to do the in-depth work required for rigorous qualitative research. Finally, analytical methods in these settings are necessarily emergent because design researchers never know exactly what will occur beforehand. These challenges all contribute to discussions that have positioned design research as an occasionally contentious research paradigm (Edelson, 2002).

Yet, since this is a new population and setting for hybrid technologies (which are themselves "new"), a project such as this one that seeks to provide access to computation and characterize the interactions of atypical students with new technologies should involve high levels of active researcher support and also flexibility in implementation. In short, design research may not be the only empirical path forward for studying this population, but it can be an apt and flexible one. It is admittedly, interventionist in character, and thus not fully free of bias. However, given the expected challenges, design research provides a set of research methods that can enable success and generate useful knowledge for better supporting implementation efforts in the future.

It is important to note also that design research often relies on cycles of implementation and refinement that occur both during an implementation and between multiple implementations. Although, multiple implementations are currently beyond the

scope of this project, as it involved a tremendous amount of time and resources and had limited personnel, the objective for me was to parallel the characteristics of design research in the development, realization, analysis, and improvement of the project with the intent to iterate in the future based on what was learned through this particular implementation.

Research Goals/Questions

This project draws inspiration from design research, but is knowingly narrowed to a single iteration of a design. Given the constraints and challenged expected with this particular instantiation, I considered there to be many things that could be thoughtfully examined for this project. Specifically, I am attempting to investigate the following questions:

1. Can these students successfully complete a hybrid technology design project, given the constraints and challenges associated with their experiences, histories, and school? Potentially, students at an alternative high school may not be equipped, willing or capable of completing a complex academic task even with the designed supports and given the aforementioned challenges that are associated with alternative education.

2. How do alternative high school students engage in the designed learning environment? What is the nature of their participation? Potentially, student engagement could be really positive and students could participate in multiple

disciplinary areas including computer programming. This could radically shift students' relationships to learning and computing or, also not unlikely, the project could be seen as unrelatable and irrelevant, thereby generating resistance.

3. Assuming students will participate in debugging by virtue of the task and technologies used, in what ways do students engage with bugs? What is the nature of students' debugging strategies? Finding and fixing bugs is an elemental type of computational thinking, pertinent to problem solving skills and fundamental to design. It is conceivable students may face considerable adversity in the face of debugging which could have profound implications for students' subsequent reactions and activities.

4. Do students exhibit indications of empowerment as a result of participating in this project? Namely, do students reflect in a productive and positive way about their experience, especially with respect to their ideas of making mistakes, their feelings toward participating in computing in general, and their enjoyment of the experience? The learning intervention will be a drastic departure from what these students are used to. Thus it could be really engaging, having the potential to change how students' view mistake making in relation to learning and computing in general, or it could be insurmountably frustrating and counter to what they expect and desire.

Outline for the Thesis

This dissertation will serve as a multi-faceted report of one cycle of design, implementation, analysis, and refinement of a hybrid technology intervention with a diverse, oft ignored, and in some respects challenging population. It documents efforts to get students using a new hybrid technology for the purposes of completing an independent design project. In the end, there were a number of encouraging results, such as observed instances of high engagement, the success students had in dealing with programming bugs and the connections some students made to computing and mistake making. There were also some areas in which the design and implementation could be improved for future iterations, namely through refinement of the activities and technologies to encompass a wider range of student interests, a more concentrated effort to cultivate a nurturing community of designers, and a more consistent fostering of motivation for and understanding of the final product (i.e. the designed pet) and its intended audience.

In the following chapter, I will discuss the population of students and one of the central theoretical perspectives grounding this work, Constructionism (Papert, 1980). I also discuss and provide examples of related youth programming and hybrid media technologies. In Chapter 3, I discuss additional literatures related to the design of open-ended technology environments, how those literatures informed the design of the project and outline the research strategy and data sources. In Chapter 4, I provide narrative summaries of the actual experience of four students throughout the five weeks of the project. These summaries are intended to both familiarize the reader with the individuals

who figure prominently in subsequent chapters and also give a sense for how the project played out from start to finish for a subset of alternative school students. In Chapter 5, I examine group interactions in which the students were involved, focusing specifically on cooperative design. In this chapter, I discuss when collaboration happened most and least often, and offer some explanations for how the activity design and the mix of media appeared to influence how and when collaboration took place. Chapter 6 is about students engaging with debugging. I analyze debugging in two ways. One is an approach where I analyzed observed bugs as they happened during the implementation with groups of students. The other is an assessment of students' debugging performance after the project was complete. In Chapter 7, I begin to explore the question of student empowerment, namely through post-project responses related to how students perceived of the overall experience. I report and discuss students' comments about mistake making, their feelings about computing, and their overall impressions of the project. Finally, in Chapter 8, I discuss what this project implies for the design of hybrid technologies and their potential for supporting learning in alternative high schools with a diverse student population. In that final section, I will discuss what I learned as a researcher and consider some specific steps for how this project could be better reiterated in the future.

CHAPTER 2

POPULATION, SETTING, AND THEORETICAL PERSPECTIVE

In the introductory chapter, I discussed some of the intentions and challenges associated with alternative schooling. For example, struggling students often have difficulty engaging in school (Finn & Rock, 1997) and alternative schools have limited access to laboratories and computing technologies (Pang & Foley, 2006). Alternative schools are not an environment highly represented in the literature on educational technology. Furthermore, the context, professionals, and students mattered greatly both with respect to the ability to pursue this project and with respect to how the project ultimately unfolded. Therefore, in this chapter I will dedicate several pages describing the partnering alternative high school site and the kinds of students who are enrolled there. Then I will discuss my theoretical orientation. Specifically, I situate this project as building upon key ideas related to Constructionism. I also focus on one new kind of technology that lends itself to some of the central commitments of Consructionism: hybrid media.

Context

Winder Alternative High School,[3] the partnering alternative school site for this project, is located 30 miles east of a major metropolitan area in a mixed rural/professional county in the Mountain West. This area is one of rapid growth and new development over

[3] All names used are pseudonyms.

what was previously open space and ranchland. To the west of the school is a view of a major fast food restaurant and beyond that the mountains. To the north is a view of a long county highway dotted with ranches, horses and a new condominium complex. According to census records, in the past 10 years, the county population has grown by over 50% with a large influx of immigrant populations and professionals. The encompassing school district covers 1,200 square miles and has approximately 5,000 students. Winder is a mile away from the lone conventional high school. According to administrators, Winder's student population is made up of 92% of students on free or reduced lunch, 10% of students parenting or pregnant and a 30% Latino population. In one-on-one conversations with teachers at the school, I learned that to be considered for the school, students must be in grades 10-12 and so far behind on course credits that they cannot feasibly graduate from the conventional high school. The enrollment at Winder has doubled since 2011. As a former teacher in the district, I was acquainted with some individuals who worked at Winder who helped me to negotiate access to the school.

In the district, students are considered failing if they have received an F, or multiple Fs, in one or more courses or have not received credit for courses due to unexcused absence. Current district policy, as presented to me by a Winder teacher, states that absences can be made up either after school or in Saturday school. However participation in Saturday school begins at 6 am and costs the students $3 per hour. An example student at Winder was a senior who had completed seven total academic credits. He needed to earn 21 credits in one academic year in order to graduate with a district

high school diploma.[4] A teacher told me of another student who arrived at Winder this fall as a junior with zero credits; she had been working on her grandparents' dairy farm instead of attending school since third grade.

Students are identified for Winder based on individual meetings with school counselors, parents and administrators, but according to teachers, some students simply drop out of the conventional high school. As she shared with me in a conversation, the principal of Winder prided herself on "finding" a student. This involved tracking a dropout student down and arriving at the students' house to convince them to come to Winder so they could still graduate. This involved communicating to students that the school was an intervention aimed at remediating and supporting struggling students in earning a high school diploma, meeting minimum state and national academic standards, and learning relevant life skills. The school website highlights Winder's mission to help every student develop fundamental skills necessary to a productive life. Approximately 66% of Winder students successfully earn a district high school diploma. That number does not tell the entire story. In keeping with district policy, after the academic year in which a student turns 18, that student is dismissed from Winder and referred to adult education, where he or she can earn a GED. Those students' outcomes are not fully known.

At Winder, as is the case for other alternative high schools (Pang & Foley, 2006), students have innumerable reasons for struggling in conventional high school. These include teenaged pregnancy or parenting responsibilities, trouble with the law, drug

---

[4] A typical year of coursework is normally 6-8 credits.

addiction, chronic illness, working to provide for their families, and persistent truancy for a variety of reasons, including absence due to lack of interest in school activities. According to teacher reports, their perception is that about 80-90% of students have used drugs, 25% have addiction issues, and 20% are on the regular juvenile or adult court docket. Teachers say it is not uncommon for the school day to be interrupted by the local police barging in to handcuff a student and take them to jail.[5] Students are often unable to stay after the school day to finish needed work because of mandated counseling sessions or work commitments.

There is one bathroom at Winder, in plain sight of the main large classroom presumably so teachers can monitor students going in and out. As an acknowledgement of the teen parents who attend the school, it is equipped with an infant changing station. Various baby paraphernalia litters one corner of the main classroom as teachers are continuously collecting used car seats, strollers and other necessities for their teenaged parent students. Students choose to attend the alternative school and continue to pursue high school diplomas despite all of the challenges mentioned. The expectation and pattern for others who have attended Winder is that the high school diploma will most likely be a terminal degree.

Because of the nature of the student population, students attend school for either the morning or afternoon session, which abbreviates the length of the school day (see Figure 6 for Winder's bell schedule). Students may also attend some classes at the traditional high school at the same time, although they are required to find their own

---

[5] In fact this happened during the project to one of the student participants and will be described later.

transportation between campuses. In general, the school day is broken into fairly languid segments and the teaching team has the autonomy to modify the day and curriculum as needed. For example, the school oftentimes has speakers come in - such as when a representative from the local credit union came in to talk to the students about basic finance. Five full-time teachers, including one special education specialist, work individually with students to make up needed credits. Making up credits largely consists of completing lengthy packets of workbook materials pertaining to a certain subject until the teacher determines the student has demonstrated knowledge sufficient to warrant course credit. For some courses, like geometry, which are required for graduation and have been flunked repeatedly by many alternative school students, the teachers hold a more traditional remedial class. School policy dictates students at Winder refer to teachers by first name and choose which courses they wish to work on in order to encourage positive relationships and autonomy.

Figure 6. Winder Alternative High School's daily bell schedule. Taken from the school's website.

All students at Winder attend a class entitled “Adult Roles” (see Figure 6) which teaches preparedness for life after high school. On the school's website, the essential learning for adult roles is based on money management, occupational skills, parenting skills, how to maintain healthy relationships, productive citizenship, and awareness of addiction. These goals tie back to Winder's mission to focus on fundamental life skills. The school also incorporates art, and one of the full-time teachers is credentialed as an art teacher. From conversations with members of the school staff, I determined that many Winder educators are proud of this opportunity for arts education because they believe it engages many students who might otherwise not participate in school.

The majority of Winder students have no experience computer programming. Winder does not have a course offering for students to learn programming. Programming is a technical elective class offered through the state's Career Technical Education program but not offered in this district.[6] Winder does have a "computer lab," which occupies one of three total classrooms at the school. The "lab" has approximately 15 desktop machines from the late 1990s placed in a long row. When asked about using the "lab" in our initial meeting, the teachers chuckled at me, probably because the lab is rarely used except to hold excess school supplies because the machines were so outdated. The teachers asked if I would be able to bring my own machines for the project because they felt their computers would be woefully inadequate.

---

[6] The district does offer some of the other Information Technology courses like a course on Digital-Media or Network Administration.

Winder is also a place where the staff aims to make students feel more in control of their own learning by giving students choices and treating each student as an individual. As a result, many of the school's structures provide affordances for doing this sort of project. Even though there is a schedule, teachers do not adhere to strictly to it nor to a specified curriculum. This made it possible to partner with the school and have students participate in the project. Students speak freely and voice their opinions openly, meaning many of the traditional classroom norms and power relationships that can hinder open-ended learning with independent projects may not be as prevalent. The staff was invested in the students and amenable to this research project. Indeed teachers dropped by every day to monitor student's projects, offer praise and came to see first hand what their students were accomplishing. One teacher, particularly proud and enthusiastic about my involvement with Winder, wrote an article about the project that was later published in the local newspaper.

Theoretical Perspective

Winder's values, a combination of helping young people develop the skills to lead a successful life and engaging young people in learning through art education, overlaps to some degree with Constructionism (Papert, 1980), the theoretical perspective that greatly influenced the conceptualization of this project. Constructionism can be seen as both a theory of learning and also an educational philosophy. At its core, Constructionism is premised on the idea that people learn deeply when they are working to create an artifact they care deeply about and can share with others. Constructionism often involves design,

independent projects, and open-ended learning along with a bit of whimsy related to the artifacts that are created.

Constructionism

Constructionism was developed by Seymour Papert to highlight and explore the potential of computers to revolutionize youth's thinking and learning (Papert, 1980), but has gone on to be applied more broadly. Based on the theory of constructivism, Constructionism is a reference to Piaget's theorizing that knowledge is actively constructed coupled with the idea that designing and constructing personally meaningful projects to share provides a rich environment for learning (Bruckman, 1998; Kafai & Resnick, 1996; Papert, 1991). Constructionism crosses traditional academic domains and aims to leverage affect, curiosity, deep thinking, and play (Resnick, 2006). It is believed working on personally meaningful artifacts can provide youth ways to relate to *powerful ideas*, like planning and carrying out a complex project or debugging (Papert, 1980).

In addition, there are implications about epistemology nestled within Constructionism. A key part of Constructionism is a focus on different ways of learning and knowing. Learners have the freedom to work in individual ways, for instance very relational and organic, bottom-up approaches to programming, called "bricolage," or more logical top-down methodologies, called "planning" (Turkle & Papert, 1991). The opportunity to work in an individual way can contribute to youth gaining more pluralistic beliefs such as the belief that there are multiple valid ways to approach and solve problems (Resnick, 1998). An extension of Constructionist learning environments, described in one of Papert's later books, is that they have an atmosphere where young

people can take breaks, watch what peers are doing, move around and daydream as well as work directly on their projects, which is not commonplace in "instructionist" classrooms (Papert, 1991). This connects to the idea of a Samba school where people of differing levels of expertise and interests learn together in preparation of a creative production for Carnivale (Papert, 1980). This open form of interaction and cross-talk among novices and experts represented by Samba schools is often an important model for constructionist learning environments (e.g., Bruckman, 1998). Computers are appropriate for Constructionist learning environments because computers have tremendous expressive potential. Given the right software environment, children can learn and make a whole range of interactive software tools, animations, and displays. However, Constructionism is not limited to computers, having expanded to include many learning environments with other designable media including crafts (Eisenberg, 2003) and the study of knots (Strohecker, 1991).

Programming in Constructionist Environments

Logo, a programming language for novices, is the typical tool associated with Constructionism. Logo involved an on-screen turtle with drawing capabilities (Papert, 1980). The premise was young people, often thought of as too young to learn the concepts of programming, could leverage "body syntonicity," a basis for knowing how one's own body moves in the world and mapping the desired movements to a Logo turtle (Papert, 1980). Students interacted with the Logo turtle by typing individual commands and eventually building entire routines and computer programs. In the process of using Logo, young people also could learn about debugging methods, design cycles, and how

computers think. Logo inspired a multitude of similar programming and development environments for young people with the same basic underlying assumptions. Some examples include: Boxer (diSessa, 2000), MOOSECrossing (Bruckman, 1998), StarLogo (Resnick, 1996), and NetLogo (Wilensky & Reisman, 2006).

One of the most widely used and current instantiations of a Logo-based environment is Scratch (Maloney et al., 2004). Scratch is a multimedia programming environment used to develop video games, music videos, interactive stories and art. Scratch was developed based on Logo, specifically with an eye toward youth who attended and were part of Computer Clubhouses. Computer clubhouses were an effort to design spaces out-of-school spaces for youth in inner cities where youth could engage in "Samba school" like interactions involving computer programming. Scratch was intended to be a media-rich programming environment that was culturally resonant, easy to begin using, complex enough to continue learning, and supportive of many different kinds of media projects (Maloney et al., 2004).

One distinguishing feature of Scratch relative to Logo is that it involves a visual programming interface rather than a command-line one. In Scratch computer programming code blocks are dragged to a scripting area and fit together like puzzle pieces, eliminating many syntax errors, such as forgetting a comma. Scratch also exposes programmers to sophisticated and parallel programming concepts seen in professional programming languages including constructs like statements, Boolean logic, loops, variables, conditionals, threads and events (see Figure 7).

Scratch is media-designed intentionally to be media-rich. This means users can incorporate and customize imported sounds and pictures into a project. The design is intended to support youth as young as eight in creating multimedia artifacts (e.g., music videos) or software (e.g., computer games). Scratch also has a large online community, complete with a storehouse of open-source projects shared and commented on by members (Monroy-Hernandez & Resnick, 2008).

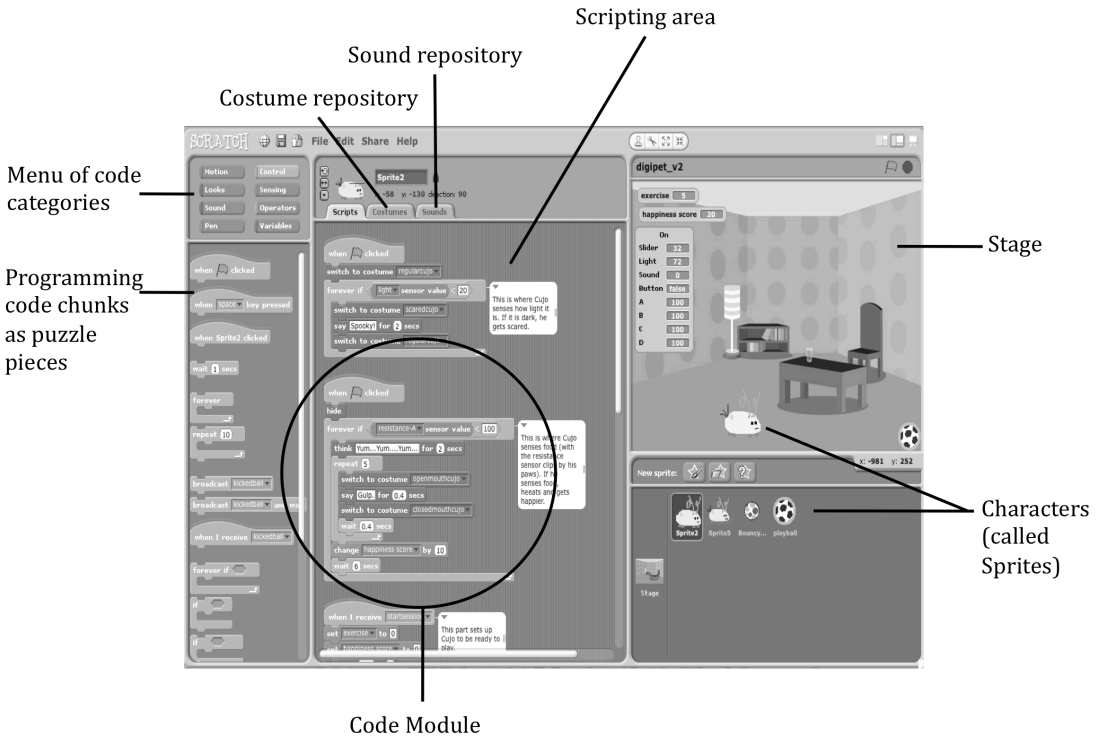

Figure 7. Scratch and its interface. Scratch has repositories for imported or created sounds and costumes. Users can incorporate many characters, known as sprites and program them via chunks of code found in the menu. The sprites then enact the programming code in the stage area when the program is run.

Given its simplified syntactic structure and emphasis on creation of multimedia animations, Scratch bears some similarity to other visual programming environments like Alice (Cooper, Dann, & Pausch, 2000) and ToonTalk (Kahn, 2004). These visual programming environments follow three recent trends in novice and youth-oriented programming languages: they more closely resemble spoken language, have the power to allow users to produce relevant, meaningful content, and can provide instant feedback (Guzdial, 2003). At the same time, the Scratch programming environment familiarizes users with the structure of more expert languages (such as Java). Scratch stands out from other novice languages in that it has grown and sustained a tremendous user. Scratch users shared over 250,000 projects online in the 18 months after the launch of the Scratch website (Resnick et al., 2009). Although intended for a young audience interested in making video games, interactive virtual art or stories, there are even some studies that show Scratch can even help introduce college students to more formal languages like Java (e.g., Malan & Leitner, 2007).

Studies in Constructionist Programming Environments

Early studies showed children were able to successfully design and program projects including software (Harel & Papert, 1991), video games (Kafai, 1996), and music videos or interactive art (Kafai et al., 2009) using Logo and Logo-variants. In Logo-based programming environments children connect deeply with important complex concepts (diSessa, 2000; Papert, 1980; Wilensky & Reisman, 2006), can effectively design and program independent projects (Harel & Papert, 1991; Kafai, 1996), and become thoughtful about their own learning (Evard, 1996; Papert, 1980). Case studies

that highlight accomplishments and struggles of a select few learners have been the predominant evidence used to support these claims.

Indeed, some modestly larger studies have shown that students in Constructionist-inspired programming environments seem to do well. For instance, Harel and Papert's (1991) mixed-methods study is a primary example of student achievements. The study showed students who developed interactive fractions software projects in Logo as a way to learn fractions and computer programming outperformed control groups, those who just learned fractions and programming in a separate and more traditional way, in debugging ability, programming knowledge, and fractions concepts. Analysis of observational and assessment data showed the students learning fractions and Logo concurrently were more reflective, showed greater metacognitive skills, and had deeper understanding. The researchers attributed learning gains to the flexibility and support of the complete learning environment combined with the deep relationships to learning Constructionism fosters. In a similar, later study, fourth graders successfully designed video games in a six-month long Logo intervention (Kafai, 1996). Results from this investigation showed that young people were capable of carrying out sophisticated software design projects in a Constructionist learning environment and that the learning environment for this type of Constructionist activity must be suitably flexible to accommodate a wide range of possible learning styles.

However there has been some uncertainty about how consistently Constructionist inspired programming environments can easily support youth in learning major programming concepts. It has been difficult to isolate and replicate some of the computer

programming gains shown in the previous studies. Some more recent studies indicate that the process of learning to program remains difficult even with technologies designed to be novice and Constructionism friendly, like Logo-like programming languages or hybrid craft technologies (Buechley, Eisenberg, Catchen, & Crockett; 2008; Kafai et al., 2009). In some studies, youth found it interesting enough to make projects using tools offered within Scratch, like the paint editor and ability to compose different media, without having to program (Kafai et al., 2009). In these cases, young people did not participate in aspects of programming critical to learning programming concepts.

Similarly, creating an independent meaningful project in a Constructionist environment has been shown to be difficult. For instance, when youth were engaged in developing video games, a tension was observed between students programming abilities and their ideas about the video games they wanted to produce (Kafai, 1996). In many cases, students' capabilities as programmers and debuggers were not robust enough for them to develop the sophisticated games they loved to play at home. When design ideas became too cumbersome or increasingly when students struggled with programming errors, they often compromised ideas and settled for easier alternatives.

Beyond conceptual difficulties involved with writing or placing code, another challenge associated with computer programming in Constructionist learning environments is whether young people can gain broadly applicable thinking skills. Recent interest in computational thinking (National Research Council, 2010; Wing, 2006) deems these types of thinking skills important. Evidence suggests youth can struggle to transfer important programmatic thinking concepts, like planning and algorithm building, in both

open-ended Constructionist environments and more structured learning environments (Pea et al., 1985). Some research suggests there may be a connection between the amount and type of support students receive when learning to program and gains in thinking skills (Littlefield, Delclos, Bransford, Clayton, & Franks, 1989). In particular, researchers caution that students do not spontaneously learn important powerful ideas like debugging, an important skill involved in computational thinking (Grover & Pea, 2013; Wing, 2006), when learning to program in any environment. Regardless of setting, novice programmers find debugging difficult (Pea, 1983). Taken together, these studies suggest there is much promise in Constructionism but that critical aspects of developing and applying programming and thinking skills associated with computation continue to be challenging for young people to learn. The field asserts that research that aims to increase our understanding of young people in relation to learning computer programming concepts and skills, especially debugging, with Constructionist-inspired technologies are paramount (Grover & Pea, 2013).

Hybrid Design Technologies

The kinds of tools used in line with the Constructionist philosophy and are still inherently computational are not all screen-based. This is exciting because there are new playful ways for young people to access computing who may not naturally relate to the domain (Rusk et al., 2008). In recent years, new sorts of media have been designed to broaden participation in computing and design through computationally enhanced materials (Eisenberg et al., 2002). Computationally enhanced materials are a type of "hybrid" technology that combines aspects of programming with the design of physical

objects, thereby combining tangible and virtual media. Hybrid media technologies exist on a continuum of programmability and designability in both physical and virtual worlds. For instance, with Topobo, children too young to read can construct creatures out of connectable plastic pieces and program them to move using motor pieces (Raffle, 2006). The programming aspect of Topobo is not transparent however, as programming occurs through physical manipulation. This means that although young children could use Topobo to learn aspects of designing physical artifacts and kinesthetics, Topobo cannot be used to learn formal, language-based computer programming. In another example, e-textiles allow youth to sew interactive clothing, learning about electronics, circuitry, fashion design, sewing, and aspects of computer programming (Buechley et al., 2008). Similarly, young people can design and build themed devices using various external microprocessing boards or Arduino (Eisenberg et al., 2009; Rusk et al., 2008). In a final example, young people use Craftopolis, miniature rooms crafted from clay, LED lights, sensors, and speakers, to create computationally enhanced roomscapes and share virtual versions with friends online (Meyers et al., 2010). Unlike the other examples, with Craftopolis, young people are designing both a physical artifact, a miniature clay room with sensors and lights, and a corresponding virtual environment, the online representation of the room with the ability to control the real world sensors and lights. Because of their newness, hybrid design technologies are just beginning to be the subject of rigorous empirical research beyond that of user studies.

Hybrid design technologies are exciting because they can simultaneously build upon elements of affect and cognition; they connect to youth culture, are motivated by

youth's existing interests and the artifacts created can often be held, traded, collected, and sometimes even "loved" (Eisenberg, 2003). For instance, after crafting electronic textiles in workshops, students can take their working creations home to keep and wear (Buechley et al., 2008). Similarly, in another example, a young person developed a hybrid media project to address needs she saw in her real life, inventing a prototype for a self-rocking carriage for her sister that was set in motion when the baby cried (Blikstein, 2011). Also, hybrid technology designers promote the idea of learning through play, meaning that learning should be fun, engrossing, intrinsically motivated, and have some whimsy (Resnick, 2006).

Designers of hybrid technologies espouse the principle of multiplicity, meaning that the technologies should make new ways of thinking prominent and also foster the development of deep connections to knowledge through interactions with the world (Resnick, 1996, 1998; Resnick, Bruckman & Martin, 1996). In doing so, the idea of multiple ways of connecting, for instance through one's body or through one's emotional attachment with an object, to computing is resonant. Recently, the potential of hybrid computational environments to foster engagement and learning has just begun to be explored because of their motivational, emotional and collaborative affordances (Raffle, 2006; Zuckerman, Arida, & Resnick, 2005). Examples of this work are described in the next section.

In making multiple ways of approaching computing salient and by potentially promoting perseverance, hybrid design technologies may have potential to support various types of learning in all types of young people (Alper, Hourcade, & Gilutz, 2012;

Falcao & Price, 2010). If tangible computational objects can be less intimidating, more broadly appealing to a wide audience and allow youth access to ideas in a developmentally appropriate way, then these environments may be able to foster exploring ideas about thinking and learning while further downplaying memorization of programming syntax and structure.

Examples of Hybrid Design Technologies

Hybrid design technologies combine design in the form of art, construction and/or crafting in the physical world with computation. The focus of this section is to describe some of these new technologies that support playful, independent learning through the design and development of physical artifacts, keeping in mind that there are many other tangible technologies and virtual design technologies not included. I will describe three types of these technologies, construction toys, computational construction kits, and computational crafts[7].

One example of a toy-like hybrid design technology is Topobo (see Figure 8) the aforementioned kinetic construction toy that is programmed through physical manipulation of the pieces (Raffle, Parkes, & Ishii, 2004). Children can build creatures by connecting various plastic pieces that resemble bones. Then the child can attach motor pieces to parts of their creation and program the motors to move in sync or separately. In this way young children can learn about kinetics and development of interactive toys by

---

[7] I have conscientiously omitted robots, like Lego Mindstorms, from hybrid design technologies because robotics generally focuses on the tasks the objects can perform, for example getting balls into a goal, rather than the aesthetic and functional appeal of the object.

thinking through processes of how bodies move and how to plan and create a figure that will also move. User-studies with Topobo showed youth as young as kindergarten can effectively make moving creatures and found the construction kit motivating (Raffle, 2006). These user-studies lend credence to the idea that hybrid computational environments can build on children's natural play and design instincts, fostering exploration of ideas (DuMont & Lee, 2012).

Figure 8. Topobo. (http://www.topobo.com). The large blue pieces are motors that can be programmed through motion. The yellow, green and red pieces attach to form a creature's structure.

Computational crafts and construction kits are two other varieties of hybrid design technology. For this project, external microprocessing boards, commonly available as

Arduino,[8] were used to couple craft or construction activities with electronics to make something interactive and aesthetically pleasing, much like an interactive art project. Historically speaking, there have been and continue to be many types of electronic logic boards available for purchase. For example, MIT's Lifelong Kindergarten group developed and supports PicoBoards, which is a replacement of sorts of PicoCrickets. When commercially available, PicoCrickets could be purchased bundled with many sensors and craft supplies. On the other hand, PicoBoards interface directly with Scratch, which has a substantial user base and is accessible for novices, but the boards have more limited sensor capabilities (see Figure 9) (Rusk et al., 2008). Other examples include LilyPad Arduino, a thin, flexible Arduino invented specifically for creating e-textiles (Buechley et al., 2008). GoGo Boards were designed to provide a low cost alternative to fancier proprietary boards for use in low-income communities, and thus the assembly instructions are freely available online (Sipitakiat et al., 2004). However, they still require specific skill and tools to build.

Every one of the above-mentioned boards is programmed through a computer and allows youth to build physical objects that interact with the environment. In some cases the boards have been altered to accommodate a particular craft or hobby, for example e-textiles (Buechley et al., 2008) or interactive paper art (Eisenberg et al., 2009). In other cases, the boards are used to facilitate the constructing of a device, like the self-rocking baby carriage mentioned earlier or a lamp whose light changes color based on sensor input (see Figure 9) (Rusk et al., 2008). The ability to incorporate these boards in existing

---

[8] www.arduino.cc

hobbies or in the creation of a custom object is motivated by the belief that combining computation with hands-on craft or construction can provide youth an opportunity to learn complicated conceptual ideas in math and engineering in natural, intrinsically motivating ways (Resnick, 2006), thus making programming more manageable and accessible (Eisenberg et al., 2009).

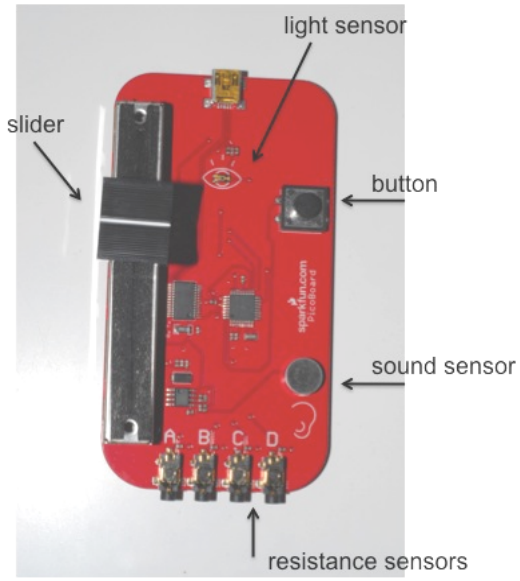

Figure 9. A PicoBoard. The board features light, resistance and sound sensors, a button and a slider.

In exploratory case studies, youth used Crickets (see Figure 10), and even earlier predecessors Programmable Bricks, to create a wide range of interactive creatures,

autonomous atmospheres, and scientific inventions (Resnick, 2006; Resnick et al., 1996). Youth became deeply involved with their physical designs and naturally followed an iterative design cycle by trying new ideas, testing and debugging with tangible computational construction technologies (Resnick et al., 1996). Researchers espoused the potential of hybrid media for learning provided that youth are given multiple culturally relevant and personally meaningful ways to connect with interactive tangible construction projects (Rusk et al., 2008).

Figure 10. PicoCricket and an interactive lamp project using PicoCricket where the light changes colors based on sensor input. (http://images.businessweek.com/ss/09/12/1209_25_world_changing_products/19.htm).

Youth, and in several cases, girls who may not have otherwise been interested in computing, have been highly motivated to create interactive e-textiles with LilyPad

Arduino (see Figure 11) (Buechley et al., 2008; Fields, Kafai, & Searle, 2012). By the same token, youth who were initially interested in computing and used e-textiles also broadened their interests and knowledge to circuitry and sewing (Fields, Searle, et al., 2012). By making the artistic, inventive, imaginative component of learning even more salient through familiar physical materials and activities, tangible computational environments is hypothesized to promote deep engagement, and can be leveraged to promote learning in multiple disciplines like programming, electronic circuitry and sewing (Fields, Kafai, et al., 2012).

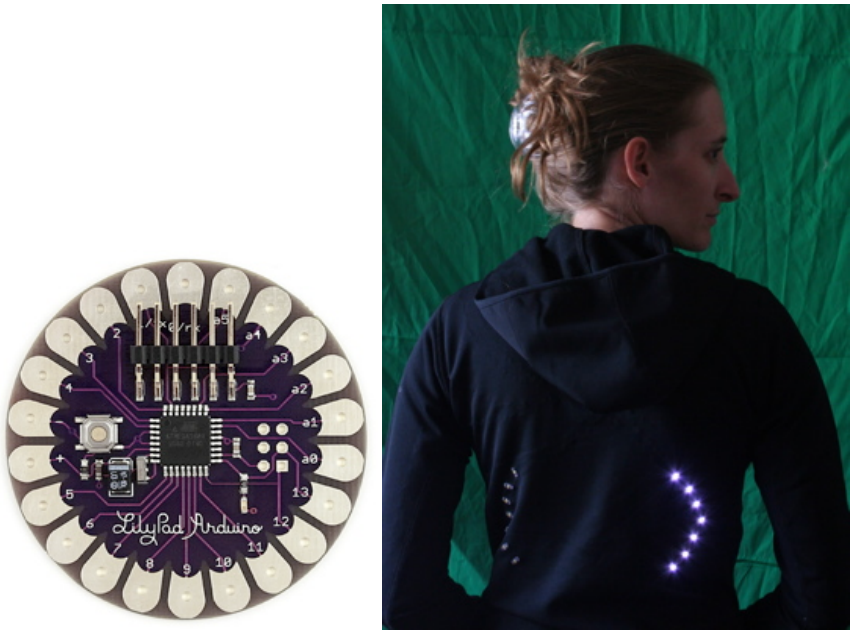

Figure 11. Lilypad Arduino and a turn signal jacket project using Lilypad Arduino. (http://www.arduino.cc/en/Main/ArduinoBoardLilyPad, http://web.media.mit.edu/~leah/LilyPad/build.html)

In some cases, artistry and computing compete for young designers' attention. For instance, without guidance, youth in an e-textile design program tended to avoid programming in their projects in favor of focusing more on artistic quality and embellishment (Buechley et al., 2008). The aesthetics of personally relevant craft projects can drive computing, but it can also interfere with completing the computing functionality of a project or realizing the design as desired (Fields, Searle, et al., 2012). Supporting young people in integrating both the artistic and computational components of projects is paramount to helping youth explore programming concepts and thinking processes.

To summarize, there are a number of current instantiations of hybrid media technologies that link some kind of external microprocessor in some fashion to physical crafting or construction. Recently, microprocessors have become less expensive and more straight forward to use. Combined with the rise of the DIY (Do-It-Yourself) movement (Kafai & Peppler, 2011), microprocessors are promising to open computing to even broader audiences by integrating programming with crafting or constructing a physical object out of known materials.

What Makes for a Constructionist-inspired Learning Environment?

As described earlier in this chapter, Constructionism was founded on the model of young people actively building knowledge through designing artifacts in an open-ended environment. A Constructionist-inspired learning environment should pay homage to the spirit of Constructionism by remaining true to the principles put forth by Papert (1980,

1991) in the areas of pedagogy, epistemology, technology, and affect. Technology, the first principle, was a core inspiration behind Constructionism (Papert, 1980). Papert's ideas about the revolutionary capacity of computer programming to change how young people think and learn and to expand what youth were capable of accomplishing helped drive the first articulations of Constructionism. However, many young people today do not relate to computing or computing culture (Buechley et al., 2008; Rusk et al., 2008). As mentioned earlier, some young people, like the students at Winder, do not have access to computer programming courses and would not at the current school district even if they were interested. A prominent report states that youth will be much more likely to participate in computing when it is not only available, but also relevant to them and within a discipline of interest (AAUW, 2000). Hybrid design technologies are capable of this for two reasons, because they have a hands-on, physical component and because they can combine elements of art, craft, construction, and computer programming, normally disparate domains. The inclusion of art, craft and physical materials with computing has the potential to attract young people who may not otherwise have been interested in participating in computer programming and ultimately makes computing more approachable (Buechley et al., 2008; Eisenberg et al., 2009).

The pedagogical principle of Constructionism is founded on the ideas that learning should be youth-directed, based on youth's individual interests and motivated by youth's individual projects (Piaget, 1980; Piaget, 1991). Likewise, a personally meaningful project can provide young people with intrinsic motivation, the impetus for the affective principle of Constructionist-inspired learning environments. Ideally,

working on projects permits young people to form a deep and personal relationship with complex concepts, like debugging and planning, not usually encountered in school learning. The students at Winder were enrolled in a remedial curriculum and had not necessarily been exposed to many complex ideas that are critical to adult life in a technological society. These students had not participated in learning through developing and completing projects, but instead completed worksheets to accrue course credit for failed classes. Students at-risk for academic failure struggle specifically to engage with school and traditional formats for learning (Fredricks, Blumenfeld, & Paris, 2004). As a result, having the opportunity to complete a personally meaningful project could give these alternative school students a new reason to invest in learning and might even have profound implications for these students.

Finally, the epistemological principle of Constructionism asserts that young people can explore more accepting ideas about what it means to know and what kinds of knowing are valid through computer programming independent projects in an open-ended environment (Papert, 1980). For example, in programming there are multiple suitable styles and variable ways to approach and solve problems (Turkle & Papert, 1991). Changing youth's relationships to knowing is important because, according to Papert (1980), students who struggle with the traditional teaching of math and science *deserve* to find productive ways to interact with concepts like building algorithms, planning, testing, and debugging. Ensuring that youth have the opportunity to approach and engage in computer programming in natural, rather than mandated, ways is fundamental to Constructionism. The facilitator in a Constructionist-inspired learning environment

should then pay particular attention to embracing different ways of approaching computer programming and in giving students the freedom to pursue their projects in individual ways.

Given the description of Winder Alternative High School at the beginning of this chapter and the description of some of the central ideas associated with both Constructionism and hybrid technologies, I hope to have conveyed the opportunity and potential match between the research site, theoretical perspective, and intervention technologies. However, I have not yet addressed the resulting design of instruction. To do that properly requires consideration of more literatures beyond those associated with Constructionism. The Constructionist perspective is notable in that it establishes a vision and set of ideals for what a learning environment could look like. What it does not feature as prominently are the specific steps, activities, and trade-offs that must be considered and organized. In the following chapter, I discuss the literature that informed my thinking and the unit I subsequently conceptualized and designed for implementation at Winder. I will also begin to describe the sources of data I collected so that I could appraise the design both during and after the unit was completed. The combination of these two upcoming sections will help establish what was a baseline for the design and why and how the resultant unit had to be adapted in the actual course of implementation.

CHAPTER 3

METHODOLOGY AND LEARNING ENVIRONMENT DESIGN

A New Kind of Hybrid Design Technology

As mentioned in Chapter 2, hybrid design media that include computer programming involve physical objects – often microprocessing boards or Arduino– that can be controlled through some separately prepared programming code. At the time of this project, PicoBoards and Scratch were available, affordable, and compatible with one another. Recall that Scratch is an environment that enables low threshold creation of animations, games, and media compositions (refer to Figure 7 from Chapter 2). PicoBoards enable the development of interactive physical artifacts. When coupled these two technologies could be used to support young people designing, developing and crafting their own physical objects that interact with both the user and with virtually programmed components. One opportunity for hybrid computing to add something special resides in the potential for emotional connection and connections to youth's existing interests. This could, in principle, further expand the types of young people who engage in computing and also encourage young people to participate in activities that span multiple disciplines. This new type of hybrid design technology combines the facets of rich media design, through Scratch, with the promising aspects of interactive, craft-oriented physical materials.

Also, in the previous chapter I described the setting, population and theoretical background for the research project. In this chapter I outline the designed intervention and describe the learning activities I developed. Then I describe the overarching research plan, which includes how the study was designed, what data were collected, and how participants were recruited.

I organized the project's instructional unit around both Scratch and PicoBoards as tools from which students could design and build their own "DigiblePet." DigiblePets are both Digital and Tangible. The pet component was important to encourage and support whimsy and emotional connection. Interactive pets draw from the idea that an object could be both a companion and a toy or both fun and in need of nurturing. In other studies of hybrid design media, youth have chosen to develop independent hybrid projects that included interactive animals, like light-sensing paper caterpillars and felt meowing cats (Rusk et al., 2008). Recently designers have even created a prototype software system, PlushBot, to help young people use LilyPad Arduino to design and develop interactive plush toys (Huang & Eisenberg, 2011). These ideas suggest that designing interactive pets may be motivating and interesting to young people and worth investigating. DigiblePets build on this existing documented interest in making pets in that they are computationally enhanced pet toys comprised of known technologies combined in a new way. The DigiblePets for this project were made from an embedded PicoBoard, a pre-formatted microprocessor (shown Figure 9 from Chapter 2) that interfaces with the Scratch programming environment through USB (as shown in Figure 12)

I have already identified how the crafting aspect of these technologies is critical. I conceptualized DigiblePets as a toy that could be designed and crafted by users out of materials such as wood, fur, feathers, googly eyes, fabric, pie pans, cork and other found supplies. The designed pet would have a physical body and a virtual persona linked to that body via the PicoBoard and USB cable. The embedded PicoBoard would let users interact with the pet. Sensors, buttons, and sliders, accessible through the pets' bodies, could be programmed to allow the user to interact with the virtual version of the pet. For instance, a furry dog might be designed and built so that when petted in the physical world, the motion depresses a button, and the dog then rolls over and barks in her virtual room (see Figure 12). DigiblePets are different from recent commercially-produced popular interactive pet toys. For example Webkins are a plush toy with no interactive capabilities that link to an online world where the owner can play games and earn items for the virtual version of their pet. ZhuZhu pets can move and react to users in the physical world and have a virtual version users can play with (see Figure 1 and Figure 3 in Chapter 1). However, DigiblePets have both an interactive physical component and an interactive, modifiable, and programmable virtual component that are connected together. Therefore interacting with the physical pet causes reactions on the screen. This enables DigiblePets to be a more robust and integrated user experience than the commercial pets.

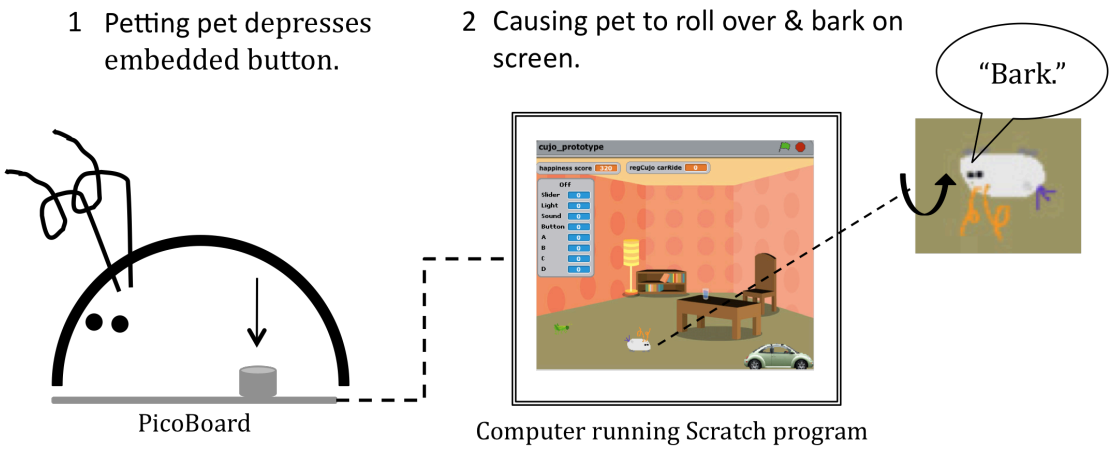

Figure 12. How DigiblePets work. Users pet the fluffy pet on left, depressing the button on the embedded PicoBoard. The Scratch program interprets the button press and computer code translates it to cause the sprite on screen to roll over and bark out loud.

Learning Environment Design

In keeping with design research tradition (Brown, 1992; Collins, 1990), the process of designing the learning environment was itself intended to be a key contribution of this project. Constructionism allows for young people to deeply engage in learning by following their interests through completing open-ended projects. Hybrid media technologies broaden the types of young people who may be interested and willing to participate in computing by providing access to computing through crafts and art. By combining a new way of accessing computing with a unique kind of personal learning, I hoped to encourage struggling high school students to complete a complex design project. While there were many important ways in which Constructionist ideas were well

suited for this environment, there were also some important recommendations from other bodies of research related to technology-based learning environments (e.g. Bielaczyc, 2006; e.g. Bruckman, 1998). Thus I designed the learning environment around the tenets of Constructionism with core commitments garnered from research on how to effectively design learning environments with technology. I will use the following section to describe the core commitments, informed by contemporary literature related to technology-enhanced learning environment design that informed my conceptualization of the broader DigiblePets activity that was to be introduced at Winder. Following that, I describe some of the recurring activities that were planned.

Three Principles for Learning Environments with Technology

Drawing from existing literature, I developed a set of core commitments to impel the learning environment design, influenced by Constructionism but with specific intent to consciously facilitate connections, develop a culture of support, and promote creative expression. These promises stem from a call to increase consideration of socio-cultural aspects of technology rich learning environments (e.g., Bielaczyc, 2006; Edelson, Pea, & Gomez, 1996), creativity in education (e.g., Sawyer, 2012) and emotional engagement, especially for at-risk students, in school (e.g., Fredricks et al., 2004). The idea was to preserve Papert's vision of a Piagetian-inspired, intrinsically motivated learning environment, to do so in a way that emphasized play and deep commitment, as Papert outlined, while simultaneously increasing initial guidance and working to develop a supportive design community. In the subsequent sections I expand on the core commitments and describe their foundations in the literature.

Facilitating connections. The idea of facilitating connections, borrowed from Resnick, and colleagues (1996), highlights that affect and intellect are intimately connected (Minsky, 1985). Independent projects should connect to youth's interests, giving young people a way to express themselves and also allowing youth the tie previous experiences and knowledge to new computing domains (Resnick et al., 1996). By facilitating connections, I meant to infuse the design of the learning environment with as many different ways to capture and promote students' engagement as possible. The learning technology itself represented this commitment because it was both tangible and virtual and connected to students' presumed inherent interests in aspects of art and craft. Recall that teachers at Winder felt their art program was effective in promoting some students to engage in school at all (discussed in Chapter 2). Finding ways to encourage at-risk students to engage in learning is paramount to these students' academic success (Fredricks et al., 2004). Combining art and craft with the potential to integrate projects with culturally relevant tokens from outside of school, such as popular music and decorative accessories, may help promote these students to participate and engage in the task. Furthermore, this project was designed to be known to students, meaning that students would have first-hand expertise with the kind of toy we were creating. Lastly, the project was intended to foster an increased sentimental component as an attempt to harness the idea that pets are companions. Animal companions are known for fostering an emotional connection. For example the use of pet training and care is an increasingly popular therapy method for developing empathy, care giving skills, responsibility, and compassion in inmates in the United States (Furst, 2006).

In addition to the project students embarked on, the designed activities for the unit promoted connections for a number of reasons. These reasons included providing copious craft and found materials[9] for building pets that students could discover and use as they wished. Also, I intended to demonstrate a variety of open-source model Scratch projects to students hoping to incite curiosity and give students a sense of what was possible with the programming environment. I planned also to invite expert designers and software developers to answer student questions and give students a professional perspective. Finally, I promoted the integration of media from other software already loaded on each machine, when students showed interest in these types of tools. The project was also designed to give students an opportunity to show off their creations to friends, family and teachers. Finally, I intended to communicate to students, as both a researcher and the primary facilitator, that it was acceptable and sometimes desirable to scrap designs or parts of designs when students felt their inspiration heading in a different direction. In all the ways mentioned, I attempted to promoted deep commitment to the design process in ways that suited each student's ways of learning and working.

Developing a supportive culture. Constructionism largely assumes a productive, supportive learning community. An effective learning culture is not inherent to learning environments but must be actively nurtured through sharing, collaboration, discussion and reflection (Brown & Campione, 1996; Edelson et al., 1996). A culture of support

---

[9] Found materials were important as this project was done independent of any funding source. However, purchasing or finding objects for use in schools is standard practice for professional educators.

must be cultivated through peer interactions (Bruckman, 1998). This may be particularly true of students at Winder who do not normally learn together. To help develop a supportive culture, I designed the learning unit to include structured times for sharing, reflection, discussion and brainstorming ideas, namely through round table discussions and openers. Knowing that these kinds of discussion can be challenging to manage for many teachers, and that the technologies and computer programming were unfamiliar to the classroom teacher, I decided to act as the primary facilitator during the project as I had prior experience as a teacher, prior knowledge of the tools and resources available, and a sense for the kinds of interactions that I expected to see.[10] As the facilitator, I wanted to foster peer interaction by encouraging students to ask one another for help, to seek each other for ideas, to show off new designs to others and to work in small groups. Finally, the project was not graded or assessed; there was no competition structure. As agreed upon by the classroom teacher, Anna, and me, all students who made a concerted design effort during the project workshops were entitled to course credit. The intention was to create a studio environment characterized by students working on a large authentic problem with constraints, with fading support from a facilitator, much like problem-based learning (Barrows, 1996), and for students to engage in sharing and reflection while also balancing the work of learning aspects of programming with desires to pursue creative expression (Sawyer, 2012).

---

[10] The researcher acting as primary teacher and/or facilitator has precedent in educational research. See for example the work of Deborah Ball, Magdalene Lampert, Richard Lehrer, or Tobin White.

Promoting creative expression. The third commitment concerned the character of the task students' pursued, creative design. Creativity is considered by some to be a fundamental skill associated with the future success of society (National Advisory Committee on Creative and Cultural Education, 1999; Sawyer, 2012). Creativity involves flexible thinking (McCrae, 1987) and the ability to develop novel artifacts or ideas (Csikszentmihalyi & Wolfe, 2001). The combination of creativity with design intentionally emphasized the importance of a fun, challenging, and engaging environment modeled after children's play where students were free to explore unique ideas. Learning through play is active, intrinsically motivated and demanding, but the term "play" admittedly does not necessarily adequately portray the intellectual seriousness of the endeavor (Resnick, 2006).[11]

Design, which is interdisciplinary by nature, often combines aspects of art, science, engineering, and technology with problem solving, and thrives on elastic thinking, improvisation, and original ideas (Sawyer, 2006). With the possible exception of art class, Winder students were not typically learning in a multidisciplinary way nor by working on independent projects. Both of these, it could be argued, actually more closely resemble skills fundamental to a productive life than remediated drill learning. By designing artifacts using technology, the students in the project would need to be actively involved in the iterative design process and engaged in aspects of complex thinking including defining problems, developing solutions, testing, and debugging. The commitment to creativity, realized through design projects, capitalized on the importance

---

[11] This is changing slowly. For example, see a recent volume entitled *Design, Make, Play* edited by Margaret Honey and David Kanter.

of fostering creativity in students to help them develop as elastic, diverse thinkers and the value of play-like learning for motivation, extended engagement and cognitive growth. For the students at Winder, this would be a drastic departure from their normal school routine.

Learning Activities

As described in Chapter 2, the values of Winder and the core ideas behind Constructionism shared some compatibilities. Winder went by different rules than a conventional school; it had an open-ended school day structure and informal setting with learning goals developed to meet individual students' needs. These features are part of what made Winder unique and suited to this design and research project, but also made Winder a challenging setting for the project. Activities planned as part of the research project needed to be fairly open-ended to be flexible enough to accommodate teachers' and students' needs, which matches Constructionist ideals. On the whole, I planned the learning activities, classroom participation structures, and facilitator interactions to support students in a complex design task without taking away from the independence aspects integral to both Constructionism and Winder. The activities and supports were a way to attempt to cultivate the norms and customs of a learning community in a school where students were most often engaged in teacher-approved, rote work. For an overall picture of the designed learning intervention see Table 1 and Table 2. Detailed daily lesson plans can be found in Appendix A.

The designed intervention had two components. First, students were to work in self-selected groups, if elected, over the course of one week to find and fix a series of

pre-programmed bugs in prototype tangible digital pets that I designed. Students would be given PicoBoards, Macintosh Powerbooks or MacBooks loaded with the prototype Scratch project and USB cords to connect the boards to the computers. The idea was that students would tinker, a relevant playful way of learning by mucking around and trying new things (Resnick & Silverman, 2005), with the prototype pet and corresponding programming code to learn how the pet's worked and what they could do. Then students would be given a list of seven pre-programmed bugs in the program code (see chapter 4 for more information about the bug tasks) and asked to find and fix each bug in the programming code. This strategy was based on *debuggem's*, used successfully for exposing students to effective programming design examples and critical programming concepts (Griffen, Kaplan, & Burke, 2012) with Scratch, suggesting its potential usefulness as an instructional technique for computational crafts as well.

Second, for the remaining four weeks, the students would design, develop and craft their own interactive pets in open-ended workshop sessions where I planned to circulate around the room answering questions and checking in on students. Students were encouraged to use and modify code from the prototype pet, reducing the need for robust programming knowledge. This would be an explicit opportunity to reuse and remix existing code (Kafai & Peppler, 2011). Additionally, I had planned for specified sharing and discussion times so that discoveries and challenges could be made public and be informative across groups. At the end of the project, students exhibited their designs in front of invited guests and community members in an evening design show.

Planned activities. The DigiblePets project was comprised of a collection of activities developed to foster the core commitments of the previous section. Daily project work was designed to follow a planned pattern including a short opener, a workshop session and a concluding round table discussion. As I discuss in the next chapter, the planned unit was altered because of last minute constraints to the amount of uninterrupted time the students were able to spend on the project. As a result, sessions were split across two days of shorter duration rather than longer, less frequent sessions, often with the opener and workshop occurring on one day and the conclusion of the workshop time and round table discussion on the following day (see Table 1 and Table 2 for details). True to the sharing component of Constructionism, the project concluded with a culminating design exhibit with invited guests that took place after classes at the school, a location that the students chose instead of a more public arena.

Opener. I intended to begin each session with a brief *opener* to provide students with a way to switch gears from the routines of the alternative school (e.g., filling out worksheets) to the design workshop. These sessions were to consist of one of the following: brainstormed sessions on design and programming, invited expert designer software engineer Q & A, or an infusion of potentially new ideas through Scratch program models or other sources. I did not plan to lecture on programming concepts; instead I planned to share prototype and online projects with students as models and examples from which code or ideas could be borrowed. I intended to invite expert programmers and designers, from the community, to share ideas and insights about design over Skype to provide guidance from outside experts as outlined by the

educational frameworks of Edelson et al. (1996) and Bielaczyc (2006). I planned to maintain an explicit focus on creating a collaborative learning community through encouraging peers to share, problem solve together, and support one another's ideas through full group discussion.

Workshop. Free, open design time called workshops of 35-50 minutes in length were planned to comprise the basic structure for the 5-week program (see Tables 1 and 2), as described in Harel and Papert (1991). By making the bulk of the time unstructured, the intent was that each student would be able to find ways that allowed him or her to be most productive. I wanted the atmosphere to be less like traditional school and instead share some similarities with Computer Clubhouses, where young people are welcome to work on the computers or glue pieces of fabric, talk, watch or seek feedback and advice from other members or the present mentor (Kafai, Peppler & Chapman, 2009).

The first two workshop sessions were planned to allow time for students to engage in the debugging protocols. These workshops intended to introduce students to debugging and programming through the process of modifying the code given to make the pet interact differently than programmed. I knew before the project commenced that students had never seen the Scratch programming environment or PicoBoards previously. However, there was no formal programming instruction planned. Instead, the students were to be allowed to play freely with their prototype pets to figure out how the pets functioned. By having a working prototype, I envisioned students experimenting with and exploring the ways in which the pet interacted as well as the virtual environment in which the pet lived by importing new media, changing values and moving code blocks. I

planned also to encourage students to borrow, modify and reuse code from these prototypes because I believed seeing and interpreting code that was already structured was perhaps more important than learning to create code from a blank slate. Code reuse and modification were the main avenues for creating programming projects and are considered a preeminent component of computational literacy (National Research Council, 1999) and a culturally relevant exercise (Kafai & Peppler, 2011). Code reuse and modification means that students would explore existing code to figure out how the programming code worked and then change parts of working code to achieve different and desired results. Combined with targeted debugging activities, code reuse and modification also provide students with a model for how to create working code and can help introduce concepts that are not always naturally explored by novices, like variables (Griffen et al., 2012). Modifying code, sometimes called remixing, a strategy often used by professional programmers and those learning a new programming language is just beginning to be studied as a valid programming practice (Shelton et al., 2010), but is not often encouraged in school settings (Kafai & Peppler, 2011).

In the activity design, once the students worked through the series of debugging protocols given, the remaining workshops were to be dedicated to students' individual projects with regular opportunities for sharing progress and opportunities to look at more examples available online.[12] Again, student teams were to be encouraged to borrow code from the prototypes and work on ideas together.

---

[12] Going online was actually a bit troublesome because at Winder you need special permission to use the Internet, which could be granted when students expressed interest, but added a level of friction to the process.

Round table. Full group sharing, critical discussion and reflection, called *round table*, was designed to briefly follow each workshop session. The class was to physically come together to share ideas, receive feedback from me and each other, work through specific problems, or contribute to a question posed by the facilitator. I intended to encourage every member of the community to participate, but participation was not mandatory. I allotted 10 minutes for round table discussions. The intention of the round table was to create a safe place to vent frustrations, ask for advice from the full group, share ideas and show off new ideas. Some goals were to work toward being able to provide targeted constructive feedback, reflect on learning and gain the confidence to share ideas. Some round table discussions were designed for general sharing, others had more specific aims for the community, for instance we attempted to define bugs, elucidate our individual ways of engaging in the debugging process, and understand what exactly design is.[13]

Design exhibit night. I designed the culmination of the project to be a Design Exhibit. One of Constructionism's main ideas is to create an entity to be publicly shared, whether that is through display within a classroom for peers or demonstration to the intended audience (Papert, 1991). Similarly, Brown and Campione (1996) have argued for an authentic *consequential task* to provide the impetus for a learning unit. In this project I planned for students to have designed and developed a tangible/digital pet for children and then to demonstrate the designs in a design exhibit. The intent was for the

---

[13] As will be discussed more in Chapter 3, some round table discussions were not enacted due to time constraints. Students often felt rushed and as a result were sometimes reticent to abandoning their projects to discuss and share ideas.

Table 1

DigiblePets Intended Activity Sequence, Weeks 1 and 2

<table>
<tr><th colspan="5">DigiblePets Intended Activity Sequence</th></tr>
<tr><th></th><th colspan="2">Week 1</th><th colspan="2">Week 2</th></tr>
<tr><th>Day/Time</th><th>Tuesday</th><th>Thursday</th><th>Tuesday</th><th>Thursday</th></tr>
<tr><td>10</td><td>opener</td><td>Opener</td><td>Opener</td><td>opener</td></tr>
<tr><td></td><td rowspan="5">(+ survey)
(+ programming intro)</td><td>(+ programming intro ~ pb&j bugs)</td><td>(what do we know about digital pets?)</td><td>(+ expert design perspective (skype))</td></tr>
<tr><td>10.15</td><td>Workshop</td><td>workshop</td><td>workshop</td></tr>
<tr><td></td><td rowspan="4">(@ debugging prototypes)</td><td rowspan="4">(@ digiblepet projects)</td><td rowspan="4">(@ digiblepet projects)</td></tr>
<tr><td>10.30</td></tr>
<tr><td></td></tr>
<tr><td>10.45</td><td>workshop</td></tr>
<tr><td></td><td rowspan="3">(@ debugging prototypes)</td><td></td><td rowspan="3">(* debug journals)
(* discussion: what is a bug?)</td></tr>
<tr><td>11</td><td>round table</td></tr>
<tr><td></td><td></td></tr>
<tr><td>11.15</td><td>round table</td><td>round table</td><td>round table</td></tr>
<tr><td></td><td>(* debug journals)</td><td>(* debug journals) (share 2)</td><td>(* debug journals) (share 2)</td></tr>
</table>

Table 2

DigiblePets Intended Activity Sequence, Weeks 3 and 4

<table>
<tr><th colspan="5">DigiblePets Intended Activity Sequence</th></tr>
<tr><td></td><td colspan="2">Week 3</td><td colspan="2">Week 4</td></tr>
<tr><td>Day/Time</td><td>Tuesday</td><td>Thursday</td><td>Tuesday</td><td>Thursday</td></tr>
<tr><td>10</td><td>opener</td><td>opener</td><td>Opener</td><td>Opener</td></tr>
<tr><td></td><td>(+ design exhibit planning)</td><td>(+ expert design perspective (skype))</td><td>(+ scratch models from online)</td><td>(+ survey)</td></tr>
<tr><td>10.15</td><td>workshop</td><td>workshop</td><td>workshop</td><td>Workshop</td></tr>
<tr><td></td><td rowspan="5">(@ digiblepet projects)</td><td rowspan="6">(@ digiblepet projects (deadline tangible bodies))</td><td rowspan="5">(@ digiblepet projects)</td><td rowspan="5">(@ digiblepet projecst (prepare for design exhibits))</td></tr>
<tr><td>10.30</td></tr>
<tr><td></td></tr>
<tr><td>10.45</td></tr>
<tr><td></td></tr>
<tr><td>11</td><td>round table</td><td>round table</td><td>round table</td></tr>
<tr><td></td><td></td><td rowspan="3">(* debug journals) (* discussion: how to debug?)</td><td rowspan="3">(* debug journals) (* discussion: what was this project like for you?)</td></tr>
<tr><td>11.15</td><td>round table</td><td>round table</td></tr>
<tr><td></td><td>(* debug journals) (discussion: what makes a good digible pet?)</td><td>(* debug journals) (discussion: what's your style?)</td></tr>
</table>

the class to invite parents, friends, influential local officials and other members of the community at large to attend the Exhibit. During the exhibit, each student team was to

showcase their DigiblePet. Modeled after a professional art exhibit, the assembled guests would then be given a chance to wander to the projects and interact with pets and design teams. Guests would also be encouraged to fill out feedback cards, developed by the class as a rubric for what we discovered makes for good design, on each design they have visited. I intended to serve refreshments and appetizers, believing that a less formal atmosphere would be more familiar and comfortable for the student participants, but had planned to formally discuss and co-plan the event with the students during the unit.

Research Plan

Beyond trying to design and implement a new unit for the sake of introducing students to a new technology, this project was also driven by research questions. In pursuing this project, I had questions about how successful students would be, how students would engage in the design project and debugging and how, by the end of the project, students would talk about making mistakes, computing and the overall experience after everything was finished.

Data Collection

Given my interest in understanding if this project could be successful, seeing how students engaged with the technologies and one another, and the likelihood of students needing to do debugging throughout the unit, videorecording each day of the project

seemed appropriate.[14] Three video cameras (2 High Definition cameras and one older Mini-DV camera) on tripods arranged behind the groups documented what the groups of students and teacher were doing. PZM Microphones were initially placed on each table, but students moved about so much that the microphones often got knocked around or unplugged. I eventually decided to capture student talk using the in-camera microphones. I recorded video of both the students' computer screens and the students themselves so that I could get a clear record of what they were working on and how they engaged with one another. At the end of each session, student teams were asked to complete a short, written debugging journal; most often they did so. I photographed students' DigiblePets and collected their computer programs at the end of the unit. Design feedback cards, developed by the students, were distributed to observers at the design exhibit. The attendees of the exhibit filled out the cards and gave them to me and I kept these as an additional data source.

In the early stages of this project, I had intended to identify and collect data from students who exhibited a range of proficiency. To this end, I also planned to use the surveys coupled with observations of the first day to select a low technological proficiency pair, a high technological proficiency pair and one pair that fell somewhere in between for in-depth study. This strategy would give me a subset of six students selected to represent three different levels of initial proficiency. However, after looking at the surveys of the prospective participants, I found that none of the students had any

---

[14] Guidelines for video research, as outlined by (Derry et al., 2010), such as camera positioning, clip segmenting, and recording with the intent to produce narratives or moment-by-moment analyses of conversation, was followed.

experience of any kind with computer programming. Therefore, choosing a subset of students based on proficiencies none of them had became irrelevant. Also, absenteeism was rampant. For practical reasons, I subsequently chose six students from four groups to participate in more in-depth research based pragmatically on attendance. All four groups were a focus for group level analyses, but I made more concerted efforts to obtain and keep video and field note records of six of the students represented in those groups. The six students chosen were present for at least three of the first four sessions of the project.[15] This meant these students participated in most of the preliminary activities that were intended to get students acquainted with computional logic, debugging and the DigiblePets technologies. In addition to the data collected from all students, these six students participated in a post-project interview, which included a debugging task assessment on the computer. In this case, the idea of *debuggem's*, formerly used in the project as a way to engage students for instructional purposes, was used as an assessment tool (e.g., Fields et al., 2012).

Student participants completed a survey regarding prior experience with programming and attitudes toward technology, design, and mistake making at the beginning and end of the project (see Appendix B). Students self-selected pairs for the duration of the project. Students were encouraged to work in pairs because they have

---

[15] A subset of students was chosen for two reasons. Firstly, being absent for 50% or more of the initial sessions might prevent students from fully engaging in the project and its components. Therefore, the more prevalently absent students were not asked to demonstrate their debugging skills or speak frankly with me about the project. Secondly, paring down to six students was necessary for feasibility. I was the only researcher on site.

been shown to be a fruitful structure for encouraging motivation and peer learning in studies of learning computer programming (Webb, Ender, & Lewis, 1986). Also, I was largely interested in the interactions between students and the technology and expecting student discourse to be a valuable source of data. Because of the nature of the student population it seemed most appropriate to let the students self-select pairings. This also led to two groups of three students because some students were absent the first day, when we formed teams and wanted to join a group in progress. One student, Jamal, did not want to work with any other students in the class and elected to work alone.

Students worked on their designs for 2-3 hours per week and received one quarter's high school elective credit for successful completion of the project, which meant they made a pet and either participated in the design exhibit or wrote an essay about their experience for the classroom teacher. These terms were negotiated and agreed upon with the faculty at Winder.

Author's and Faculty Involvement

In keeping with design research, I was very involved with the planning as well as day-to-day implementation of the project. As a former professional software designer and mathematics teacher at more conventional high schools,[16] I worked closely with the classroom math/science teacher to implement a five-week (the designed intervention was four weeks, but the project ran a full five weeks due to shifts in scheduling and student time) long project on designing digital pets. I encouraged the math/science teacher at

---

[16] One of my teaching placements was at the conventional high school in Winder's district.

Winder to participate in the project by making her own DigiblePet alongside the students, helping to foster the idea that teachers are learners too and that knowing is an ongoing process regardless of age or status (Koschmann, Myers, Feltovich, & Barrows, 1993; Papert, 1980). Other teachers, students and administrators were very interested in seeing what the students were doing and frequently visited the classroom. Students often interrupted their work to show off their designs to visitors, which I believed to be valuable in keeping with the spirit of publicly sharing the creations. The art teacher was particularly popular in this regard.

To help foster a supportive learning culture, Bruckman (2000) called for *ubiquitous support*, where support is available at any time and for any reason when asked for. Assistance was available from me, and I deliberately focused on providing coaching and then fading my support in order to encourage peer collaboration and student autonomy. I attempted to refrain from dictating steps or telling students the answer to questions for which I saw a viable path for students to generate those answers on their own with support. I was always present throughout the workshop days and times, ready to answer questions, help guide students and work through ideas. However, students were encouraged by me throughout the unit to ask one another for help and try to work through their bugs independently.

Sampling and Recruitment

Students were selected for the DigiblePets Project from the entire population of alternative high school students based on interest, but participation was subject to the discretion of the teacher and principal. I had an initial meeting with staff at Winder

during the preceding summer break where I demonstrated the technology tools and discussed my vision for the project. The staff expressed excitement about the opportunity to bring innovative technologies and learning environments into their school. However, they identified some initial caveats. Because the purpose of the school was to provide support for students who were not succeeding, Winder teachers would not have complete rosters until well into the school year, as students from the traditional school would be identified as "failing" and then subsequently processed through the system. As a result, the DigiblePets project needed to wait nearly three months to commence. Similarly, because students have generally struggled with school for a long period of time, I was warned that they often drop out of the alternative high school program. Recruiting and maintaining students in a 5-week long course would be a primary challenge according to the teachers.

The teachers also expressed to me that they wished to encourage students to participate in the project that they believed would be enthusiastic, receptive to an unfamiliar learning environment and goals, and had shown some success within the Winder setting. For instance, they did not want students to participate that were resistant to learning or had very poor attendance. However, when the time arrived to invite students to participate two months later, the staff shifted slightly in their position and felt that it would be better for the students to not limit access to the learning experience, believing the uniqueness of the project might be similarly or even more beneficial for even the most difficult students. The staff invited me to speak to the entire group of

morning students about the project and encouraged me to give participation materials to any student who had interest.

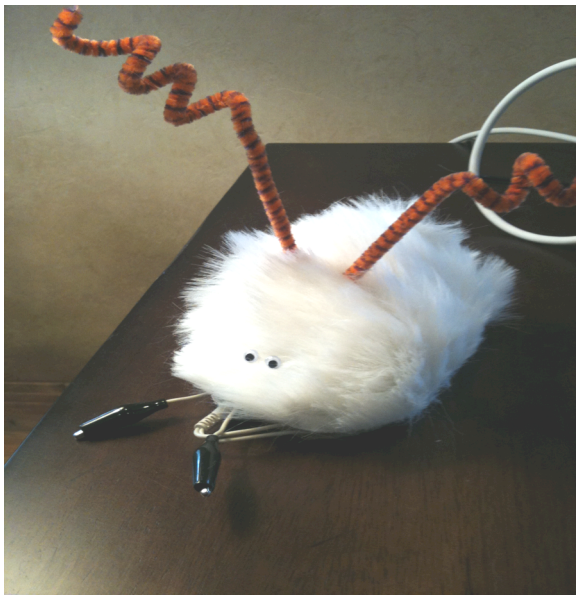

Figure 13. Cujo: A prototype DigiblePet. The pet was made out of fur and pipe cleaners with a cork body to hold its shape. The PicoBoard is embedded beneath.

When introducing the project, I presented a prototype DigiblePet (see Figure 13) that I made, described how the pet functioned and explained that the students would be responsible for designing, developing and crafting their own pet using art materials, found materials and computer programming. I also explained that the time required for

the project would replace class time during which the students could be making progress towards their graduation requirements. Approximately 25 students were present in the morning group. After my presentation, I asked students to raise their hands if they might be interested in participating in the project. It took several seconds before students volunteered, but to my surprise every student did. I distributed participation materials to all the students and informed them about their rights as prospective participants. The teachers asked me to leave a few extra forms for absent students and absent-minded students who may lose their forms in the coming week.

Resultant Participant Numbers

Eleven students (grades 11 and 12) participated in the project. Two students dropped out after the first few sessions for reasons of too much other academic work that they needed to complete, or too little interest in continuing, or a combination of the two. The remaining students were 4 females and 5 males (n=9) arranged into groups as follows: group 1 - 2 males, 1 female; group 2 - 2 males, 1 female; group 3 - 2 females; group 4 - 1 male. Student groups worked on large tables with Macintosh MacBook or PowerBook laptops that I obtained from various individuals for temporary use during this project, a PicoBoard, a set of alligator clips, a USB cord, and various craft materials. Tables were arranged in two rows in a V formation such that all student chairs faced towards the center of the room. The craft materials, glue, tape, scissors etc., not introduced until later in the project, were placed on their own table so students would have to move about the room to retrieve needed items and thus encounter or see other students and their projects. I moved the tables and chairs to the more communal

arrangement at the beginning of each session and returned them to their original places after each session. The project took place in 13 sessions, 12 workshops and one introductory session, over the course of 5 weeks and culminated in an after-school design exhibit. Each session was between 35 and 70 minutes long (the original plan was to have 90 minute sessions) depending entirely on what other activities were planned that day by the Winder staff.[17]

Data Collection Instruments

The data collected during and immediately following the project were intended to capture, as much as possible, how students approached and engaged with the design task, how students dealt with errors, and how students felt about their experiences. For an overview of the research design, see Figure 14. The corpus of data include:

- Daily video recordings of design teams as they engaged in activity,
- Post-project interviews,
- The students' designs and programs,
- Daily debug journals,
- Pre and post surveys,
- Design exhibit feedback cards,
- My field notes.

I expected important things to take place through interactions with the technologies and tasks, so I captured video and then selected portions of the video corpus

[17] Recall in Chapter 2 that Winder was flexible with their schedule and would often make impromptu changes to accommodate guests or special activities that were considered beneficial for the students.

to analyze in greater detail. These data facilitate analysis of students' conceptual processes as they worked together to solve problems and learn about hybrid design media and computer programming. All together, I collected approximately the equivalent of 136 hours of recordings, including nearly 12 hours of observation time. A brief written survey I had prepared was given to each student so that I could get more information about students' ideas about making mistakes in relation to learning and feelings about computing and the project experience. The interviews were conducted after the project, when trust with the facilitator had been established, and were intended to give students an opportunity to talk about their experiences. Because of small participant numbers and the formative nature of the work, these surveys and interviews were considered as being useful for providing more details about each individual student, rather than as data from which I would be determining statistically significant differences associated with the project. The remainder of the data were intended to triangulate what happened during the project and why.

I kept a journal of field notes and personal reflections for each session in the spirit of Lampert's (2011) work using her own mathematics classroom as an object of study. The journal provided was a tool for me to reflect upon and consider how to immediately revise and modify activities and support structures as needed depending on what I perceived as student needs and qualities of student interactions. My field notes also served as a modest design record (Cobb, Confrey, diSessa, Lehrer, & Schauble, 2003) as the project unfolded (see Appendix C for a sample page of field notes).

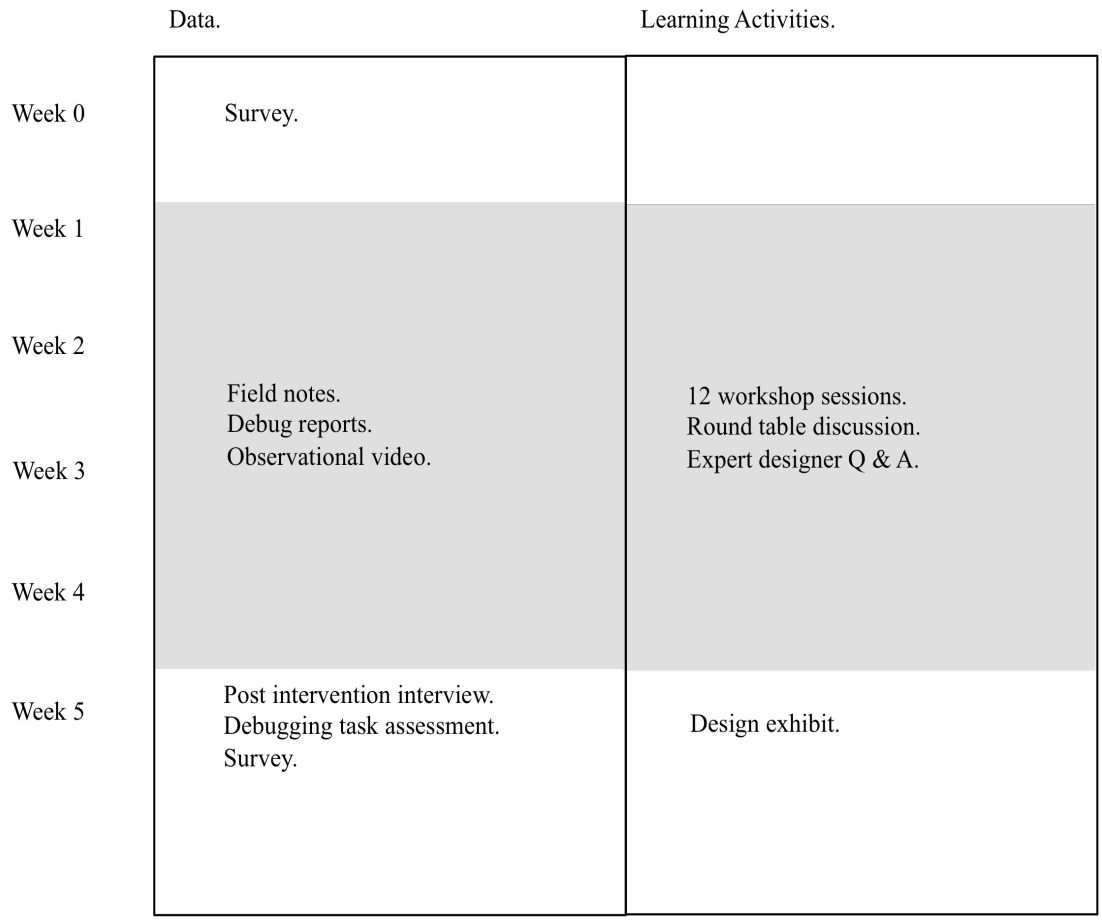

Figure 14. Research design.

CHAPTER 4

HOW INDIVIDUAL STUDENTS ENGAGED WITH THE PROJECT

Despite a number of challenges, all student groups successfully designed and developed a tangible/digital pet with a crafted physical body and corresponding virtual pet with at least some functionality that differed in ways from the given prototype project (see Table 3 for student projects). Students were engaged in aspects of the project, connected to their pets, and showed pride in their work. There were, as expected, several times in which students were challenged or frustrated with the project and periods when the students were not interacting with each other collaboratively. Additionally, students did not uniformly engage in all the facets of media design. Still, on the whole, the project showed promise using computational crafts with an academically struggling population.

My goal for this chapter is to provide some more detailed images of what individual participation looked like. What Eisenberg (2003) and others have described as a path to promote learning in computational domains through an artistic, hands-on, crafting domain, was indeed apparent for some students in this project. There were also some clear derivations from this. For example, one student took on the role of a programmer who worked only on programming the virtual pet and another who took to programming right away but then dropped programming to devote all her time to crafting the physical pet. The latter student was drawn away from computation because of an unequivocal attachment to the ensuing physical pet design.

Prior to providing the illustrations of how individual students engaged with the project and the technologies, it is necessary for me to first discuss in more detail one important factor that led to modification of the unit. Specifically, the amount of available workshop time for the project changed how the workshops needed to be run and consequently the ways in which students engaged with the task. Following a discussion of the role that available time played, I will provide four narratives of student experiences with the project based on my records and observations. Using their own words and descriptions of their activities throughout the design life cycle, I highlight and reflect upon how different students' approaches to and work throughout the project shifted my ideas about ways participating. The final portion of the chapter will speak to the recognition that a hybrid design technology and design activity that has the potential to broaden the ways in which students can engage with computation can still have features that can also limit or narrow individual participation.

Narrative Construction

I developed the student narratives below based on the various data collected (see Chapter 3 for details). There were nine students involved throughout the project. Six were the primary focus for intensive data collection. Between the combination of field notes, observations, and video review, four students were selected for further development as descriptive narratives because of the contrasts they represented from one another and the ways they participated in different facets of the project. For instance, one student, Tegan, shifted her focus during the project. This was something I noted in my field notes and

wished to trace over time. Another student, Jamal, focused almost entirely on the more artistic portions of his project and only programmed his project on the final day of the workshop. Despite this, he had a very successful project. I wished to better understand how his project developed over time. Another student, Carlos, was focused solely on programming his pet's functionality and did not craft the on or off screen pet. I wished to understand how Carlos engaged in the project. A final student, Dino, had difficulty throughout the research project and struggled to reconcile his hatred for computers with the pet his group was making. Thus, I selected him as what may be considered a negative example.

The process of narrative preparation involved an initial review of my own field notes, observations, and interviews for contrasting cases. The four above were identified for the qualities described, based on the records I used in that initial pass. Following their identification, I reviewed the entire video corpus and proceeded to identify and transcribe portions of video in which these individuals figured prominently in an interaction with another student or with the technology. I subsequently annotated these transcripts and began to create a timeline of events for these students. Using these, I began to posit what factors may have influenced the ways in which students had engaged. For example, as I will discuss later, affective connection to a pet become a potential candidate for influencing one student's engagement. When I had identified that, I re-reviewed video and transcripts to identify specific word usage (such as referring to the pet with a personal pronoun rather than 'it') and to transactions between the case student and their peers that related to the candidate influence (such as speaking possessively about the pet).

The result was then crafted into and iteratively refined as the narratives that appear below. While this was not formally part of the analysis process, some of these were submitted for peer review and presentation at conferences for feedback (e.g., DuMont & Lee, 2012) and were revisited and refined further based on suggestions from scholars in the field.

Table 3

Student Projects

Students in Group	DigiblePet Project
Tegan, Rocky & Ted	Monkey
Steph & Tabitha	Feathered Unicorn/Hippo
Dino, Carlos & Maya	Alien
Jamal	Zebra

Design Versus Realization: Time

The constraints of real classrooms caused me to have to modify the activity design, as outlined in the previous chapter, during the implementation of the project at the alternative high school. I conceptualized the project, based on initial conversations with

the faculty at Winder, into 90-minute blocks with students twice per week. However, due to unforeseen scheduling constraints, which are minimal at Winder but still existent, the available time with students was altered just prior to implementation to be three to four 35-50 minute sessions per week, essentially splitting the original sessions across two days but resulting in about 10 hours of time versus the planned 12 hours. Additionally, days when students fell behind on "packets," the main way of making up course credits by completing large workbooks of practice problems, were often days students would choose to stay or were asked to stay by another teacher at Winder in the main room and miss the project day entirely. To deal with this change in schedule, I made sure to set up each laptop and pet in progress before students arrived and broke down the room after students left to capitalize on the brief stints of time students had.

With shorter sessions, the intent to give students have ample time to wrestle with ideas and make progress was threatened. This meant that there was less time for round table discussions, important for promoting a collaborative culture, and less time for openers, a way to introduce new ideas. It was too difficult to try to fit everything in because students were enthusiastic about starting their projects right away when entering the room, sometimes arriving nearly 10 minutes early to get started (so long as they weren't behind on packets), getting their in-production pets out of plastic bags and starting up their Scratch programs, so I improvised and decided to keep the main activity structure as before, but opted to in certain cases distribute daily activities over two days (see Table 4). Therefore, I endeavored to stick to planned openers and round table

discussions but sometimes pushed them or, on a few days, omitted them entirely to make room for students' open-ended development time.

These changes alone created challenged as the continuity that could be established with one and a half hour long sessions was disrupted. Although some students often arrived early each day, others would amble in 10 minutes late. These students often showed trouble recalling the ideas and tasks they had left hanging from the last workshop. Students complained to me regularly about not having enough time. Once students began work on their pets, there was student reluctance to switch from their pets, which further discouraged completion of some roundtables.[18]

One example of the imposition of time was during workshop 6, the day that my expert software designer Skyped in to give his perspective on design and answer student questions. Students wanted to continue their discussion past the allotted 15 minutes. I wanted to allow the students to continue talking with the professional programmer, who did not go to college and instead taught himself programming and eventually worked his way through a company to a lucrative programming job at a stylish local company, because the students found his insights to be relevant and useful. However, letting the conversation continue meant that our workshop session for that day was all but eliminated. Making tradeoffs like this one were a continuous part of the project where development time was precious but so were opportunities to share and discuss.

---

[18] In my lesson plan notes for lesson 7, next to a round table discussion plan to revisit the idea of what students do in the face of bugs, I have the representative comment, "Students are so far behind on their projects that I let them work rather than revisit this. Will touch on it in interviews" (M. DuMont, field notes, November 21, 2011).

When asked in interviews how to improve the project multiple students interviewed mentioned increasing time specifically. Tabitha, who struggled with confidence as well as computer programming throughout the project talked about the role of time.

> Tabitha I think just more time. I think that's pretty much what I would want...I really wanted to get it. And I thought I would. It just wasn't enough time for me. It wasn't enough time for me.

Jamal reiterated Tabitha's idea that time was an issue for him with the following suggestion on how to improve the project.

> Jamal And maybe make the time we do it, we only did it for what like an hour or 45 minutes or something like that, maybe extend that...Yeah. Cause we would set up and just start doing stuff and getting ideas and then we'd have to go or whatever.

Students' complaints centered on having a difficult time transitioning into the project and then feeling the session would end abruptly just as they were getting into the project mindset or making progress on a problem.

While it is true that increased time dedicated to the project would have allowed more design time and better adherence to planned activities, feeling time constrained is a catchall for student struggles and does not provide insight into how to manage additional or existing time in more productive ways. I would have wanted more opportunities for students to participate in dedicated sharing, discussion and idea generation, more

Table 4

Digiblepets Realized Activity Sequence. Items in Red Were Omitted from the Original Planned Activity Sequence and Items in Blue Were Added to the New Activity Sequence Based on Shorter Time Blocks

Day/ Time	***10.26***	***10.28***	***11.02***	***11.08***	***11.11***	***11.16***	***11.18***
9:10	**opener** Survey	**opener** Pb & j bugs	**opener** Demo cujo	**opener** ~~Intro craft materials~~ Design perspective - D. S.	**opener** ~~What makes a good digital pet?~~ Plan for design exhibit	**opener** Demo Scratch projects	**opener** Post survey
9:20	**workshop** Peanut butter & jelly intro	**workshop** Debug Prototype	**workshop** DigiblePet projects	**workshop** DigiblePet projects	**workshop** DigiblePet projects	**workshop** DigiblePet projects	**workshop** DigiblePet projects
9:30							
9:40							
9:50	**round table** What did you learn about instructions?	**round table** What is a bug?	**round table** Share	**round table** Share			

	10.27	***11.01***	***11.03***	***11.09***	***11.15***	***11.17***	***11.22***
9:10	**opener** Intro cujo	**opener** Digital pet brainstorm	**workshop** DigiblePet projects	**opener** Design perspective - N. B.	**workshop** DigiblePet projects	**opener** Demo Scratch projects	**workshop** DigiblePet projects
9:20	**workshop** Play with Cujo	**workshop** 1. Debugging prototypes				**workshop** DigiblePet projects	
9:30		2. Begin own pet		**workshop** DigiblePet projects			
9:40							**round table** Advice for next year's students.
9:50	**round table** Share	**round table** What is a bug?	**round table** Share	**round table** Share	**round table** What makes a good digital pet?	**round table** How do you debug?	*design exhibit*

exposure to Scratch projects and existing digital pets. However, regardless of the time allotted, the fact that students seemed rushed and somewhat reluctant to the structure of the sessions also speaks to the complexity of the independent project task for these students. Restrictions on time were an unintended challenge and non-negotiable. Despite the constant pressure of time within a real school setting, students were able to make connections to the project and successfully design and build their DigiblePets. To get a picture of how the project unfolded for students, I provide a description of the project through brief narrative accounts of four student's experiences below.

Tegan

Tegan was a junior transfer student who chose to work with two senior boys, Rocky and Ted. She was a confident and charismatic girl with blond hair that changed artificial hues almost daily. Tegan talked easily with her group mates mostly about relationships, movies and their network of acquaintances. Unlike her schoolmates, Tegan claimed to enjoy math, saying she was good at it. She was always smiling and often toted a large frozen coffee concoction from the nearby fast food restaurant to class. Tegan moved to the district the previous year from Florida with her Mom. Tegan described an aspect of her relationship with her Mom on one occasion during class. She told another student across the room that her Mom had asked to borrow enough money from her daughter to buy cigarettes that morning. In her reporting of that exchange, Tegan was dismayed by the request and told the student she did not comply. The student she was speaking to was incredulous about a Mom acting this way. It was clear that the family

relationship was strained and that Tegan's mother was not always as responsible as her daughter wished.

Why Tegan moved was not clear, but after the move, Tegan struggled at the traditional high school for her sophomore year. She earned virtually no credits, failing nearly every class. The math-science teacher at the alternative school expressed to me her confidence in Tegan's abilities and believed she would do well in the Winder environment. Tegan only missed one of the 12 workshop sessions, an unusually high attendance record for the project, and she was one of the few students who voluntarily attended the design exhibit at the end of the project.

In keeping with her outward confidence, Tegan did much of the initial programming for her team in the workshops prior to the craft materials arriving. Tegan controlled the keyboard for the majority of the initial debugging task workshop (recall the structured debugging task took place over two workshop sessions) and with her partner Rocky's input, they were able to solve six of seven bugs with minimal facilitator support, even though neither of them had any previous programming experience (see Table 5 for debugging tasks). Noticing their bug-fixing prowess, two other teams asked Tegan and Rocky for advice during the initial structured debugging task workshop. No other teams were solicited for advice in this way.

Tegan and Rocky also made three independent design changes to the prototype code of their own volition on the initial structured debugging task day, workshop 1. In fact, Tegan was ready and excited to implement aesthetic changes to the prototype project in the very first minutes of seeing the Scratch program.

Table 5

Debugging Task Days: Bug Tasks Students Were Given with Corresponding Programming Concepts Explored During the Initial Two Workshops with a Prototype Digiblepet (Picoboard) and Corresponding Scratch Program

Bug	**Programming Concepts Covered**	**Bug Text**
1	Events; Basic Programming Statements; Multi-media: Speaking Bubbles	The grasshopper is annoying. When Cujo bumps Mr. Jumps he should say "pesky bug" not "hello."
2	Input: Button Pressed; Multi-media: Sound Editor	Cujo barks. All wrong. He is really supposed to be a cat. Make him meow instead.
3	Threads; Stage as Coordinate Plane; Programming Modules	The car (a VW Bug) always starts in the same spot. It's the wrong spot. Make it so the car always starts at the very bottom left of the screen.
4	Conditionals; Input: Resistance Sensor	Cujo can eat and eat with no effect. Lame. Make it so when Cujo eats he grows **bigger**.
5	Threads; Variables: Initializing, Posting	Cujo has a happiness score. But it goes up and up and up. Figure out how to make sure the score is zero when you start a new session.
6	Multi-media: Paint Editor; Input: Light Sensor; Costumes; Wait	When it's too dark, Cujo gets scared. When this happens, he is supposed to turn green. Make that happen. (Don't forget to turn him white again when he isn't scared anymore)
7	Variables; Conditionals; Event Handling Across Sprites	Cujo can ride in the bug. Then the car self-destructs. Figure out how he can ride in the bug 2 times (but no more than that).

In the following episode, Tegan wanted to begin the debugging task with a design idea for the prototype pet. Students were supposed to be working on fixing given bugs,

but instead, Tegan was interested in making the pet's environment look different. The that time in the unit, other teams were still trying to figure out what was being asked of them and what Scratch and PicoBoards were.

Tegan	Let's change his *(the pet's)* house.
Rocky	How do we change his house?
Tegan	I don't know.
Rocky	Can we figure it out?
Tegan	MmmHmm.
Tegan	Ok. Let's change his house.
	(she moves the computer towards her)
Rocky	Well we've gotta wait for instructions first.
Tegan	No we don't.[19]

Tegan's first utterance once the Scratch program was opened on the structured workshop was "Let's change his house." Not only was she already developing her own ideas, she felt confident that they could realize the ideas. Tegan then reiterated interest in implementing her idea. She was motivated to work with the technologies and took the computer from Rocky to do so. Tegan also did not think the group needed to wait to begin their design, even though making aesthetic changes had not been specifically offered by the facilitator as a thing to do at this point. After this episode, Tegan

[19] The language of all interview quotes has been recorded verbatim to retain the authenticity/ originality/ spontaneity of the text.

proceeded to scroll through the computer code, which she had never encountered before, clicked on the stage icon, clicked on the backgrounds tab, clicked on import and found a beach background, which she chose (see Figure 15 for reproduction). She then imported a woods background instead. This was representative of what Tegan would do. She would develop an idea about some aspect of the pet development and figure out how to accomplish it.

This episode suggests two important things to me: one Tegan seemed excited about the project and eager to begin implementing design ideas, and two, Tegan, as the programmer, seemed able to understand aspects of Scratch and the ensuing programming code even though she had no previous programming experience. Tegan and Rocky together, with Tegan at the helm, were the most successful group at finding and fixing the given bugs during workshop 1, solving the given bugs more quickly and with less facilitator support than any of the other groups, and also were a creative group during that day, going above and beyond the requirements to make unique changes to the project. No other groups made so many changes to the prototype pet, Jamal made one and the other groups none, during the structured debugging task. After a fixing a particularly complex bug #6 (see Table 5 for the debugging tasks), near the end of the workshop 1, Tegan and Rocky decided to change their cat sound to the *shutup* sound they recorded themselves using the microphone built into their laptop. This was not part of the debugging tasks, but something the pair had developed on their own, to make the prototype pet say, "shut up" out loud when the button was pressed.

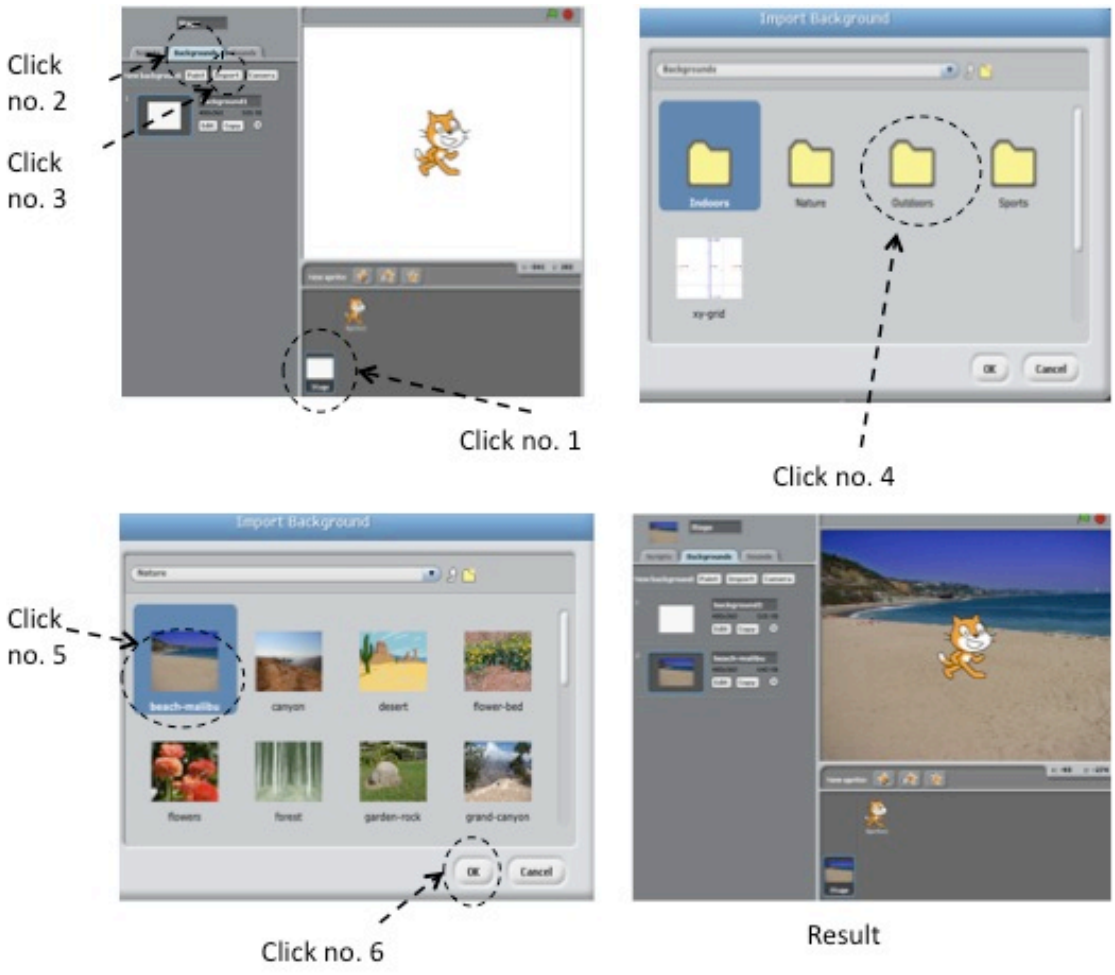

Figure 15. A reproduction of Tegan's background modification in Scratch.

In the episode, Tegan changed the code to enable the *shutup* to play and Rocky pressed the button on the PicoBoard to cause the announcement.

Tegan	Ok. Wait. Where is it again? (*changes play sound cat to shutup)*

Rocky	Go push start.
Tegan	(*clicks start*)
Rocky	(*presses button on PicoBoard)*
	("shutup" plays on speaker)
Tegan	Yay! We are so smart Ours is the coolest one!

It seemed Tegan was excited about the new functionality the pair developed for the prototype pet when she said, "Yay! We are so smart! Ours is the coolest one!" In this episode, Tegan added further evidence that she found the project fun and exciting even when only the computer programming portion of the design project was available for use.

The following workshop day, Tegan worked alone in Rocky's absence to solve the final and most intricate bug (bug #7), which involved understanding variables, altering mathematical conditional statements and event handling between different sprites, challenging programming concepts (see Table 5). Tegan was the only student to successfully and completely solve this final bug. Tegan worked 49 minutes to fix bug seven, but even then she was not completely satisfied and expressed a desire to make the car disappear after having fixed the car-related bug. The instructions for the bug did not include this final aesthetic fix, but Tegan was determined to work on it even though it was not required.

The following several workshop days, Tegan and Rocky, and eventually Ted who joined them after a brief stint of mandated time away from school,[20] worked to make their own virtual pet and corresponding functionality. Tegan controlled the computer and Rocky supported her by adding ideas and sometimes directing her in what to try. Together they created a monkey with many costumes (see Figure 16) that walked, danced, ate, "partied" and climbed a ladder to get onto the bed in his room (see Figure 17). The monkey spoke, listened to music and had an elusive bunch of bananas to chase. Tegan led the functionality changes the group made, remixing the prototype code. During workshop 5, Ted's first day, Tegan was absent, and Rocky showed off the pet to their new partner. Ted was impressed with the project, he asked, "How'd you design all this?" Rocky replied, "We know what's up." That Ted was impressed with the group's work adds to the idea that in two days of independent project work, Rocky and Tegan created an interesting virtual pet that required what appeared to another student to be a remarkable amount of programming. Rocky's response implied he was confident about his and Tegan's abilities as programmers and designers.

Tegan alone developed much of the team's resulting functionality. One of the most complicated pieces of functionality she developed was to remix the pet's walking code. The pet walked across the screen based on input from the slider that was translated via mathematical expression into coordinates (see Figure 18 for the original prototype walking script). No other student attempted to understand the mathematical code but Tegan was unsatisfied with the way her monkey moved and determined to fix the

---

[20] Ted was physically taken from the school on the first workshop day by the police in handcuffs and because he was 18, put in jail.

Figure 16. Some of Tegan, Rocky, and Ted's monkey costumes. Each one was painted by hand onto the original monkey stock sprite. The costumes all refer to "Cujo" the original prototype pet because students were using the prototype project as a guide.

Figure 17. Tegan, Rocky, and Ted's Scratch Program. Their monkey climbed the ladder and jumped on the bed.

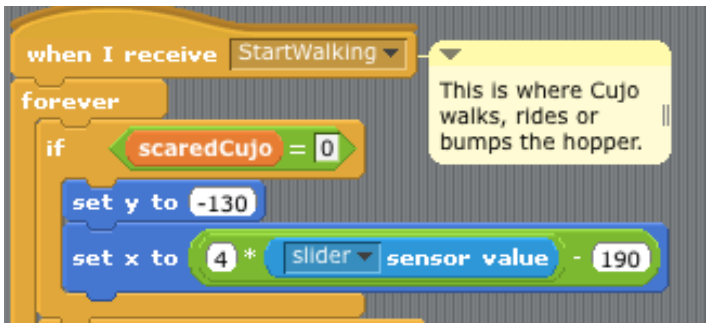

Figure 18. The original prototype walking script.

problem. In the excerpt below Tegan tinkered with the mathematics in the code until she achieved the desired result.

Tegan	Ok. I'll figure it out then.
Facilitator	Sure you will.
Tegan	Ok. y. (*changes y = -130 to y = -150)* (*tests*) (*changes back to y = -130)* *(tests)* *(changes y = -110)* *(tests)* *(changes x = "sensor value *4 - 200")* *(tests)* Oh cool, I did it!
Steph	What are you doing Tegan?
Tegan	Just messin with stuff.

This episode illustrated how Tegan tinkered with code in a playful but sophisticated way. During the short excerpt (less than 60 seconds), Tegan changed small pieces of code and tested the results four times. She made small modifications and based on the results ascertained whether she should make a further change. At the end of the excerpt, Tegan exhibited pride by saying "Oh cool, I did it!" Then Steph asked Tegan what she was doing. Tegan's response was "Just messin with stuff." Her "messin" changed how Tegan's pet walked in its virtual space in a way that was pleasing to Tegan. Again, Tegan determined an idea then set out to figure out how to accomplish it independently.

Another unique idea that Tegan decided to embark on occurred during workshop 4. However, in this instance, Tegan's previous exhibition of enthusiasm and perseverance in programming were subverted. Tegan, alone again, decided, with some idea brainstorming with the facilitator, to make a new piece of functionality where the monkey would climb a ladder to get up onto the bed in his room. She got very excited about implementing the idea saying, "Oh my God. That's cool. I wanna do that!" The functionality required several pieces of development. First Tegan had to draw a realistic ladder by hand and position it to look authentic. Second, Tegan had to establish that when the monkey touched the ladder, he would climb. Third, Tegan had to program the monkey to climb. Tegan worked hard on the first piece of the design. At the end of the workshop, Anna, the classroom teacher, came over to check out what Tegan had been working on. After seeing the ladder idea, the classroom teacher exclaimed "Oh that's cool!" and "That's a good idea!" Tegan immediately rebuffed, "Thanks. Don't take it!"

That response showed Tegan's personal investment in the ladder climbing functionality. She wanted others to see her work but not copy her ideas, "Don't take it!" The classroom teacher's response confirmed to Tegan that her idea was unique and interesting. Tegan displayed feelings of ownership and interest in the ladder climbing idea. However, the idea was unrealized until Rocky and Ted took it up. After workshop 4, Tegan ceased to touch the computer neither implementing nor providing insight into any other aspects of the virtual design. Why Tegan immediately and completely stopped her involvement with computer programming and an idea she showed interest in was observed to be due to the introduction of the craft materials during workshop 5.

Throughout the project, Tegan showed a personal connection to the group's DigiblePet initially through her interest and enthusiasm for programming aspects of the pet and subsequently, in her dedication to creating the physical pet. However, when the physical development of the pet commenced, Tegan devoted all of the remaining workshops to creating a physical version of the monkey, letting her partners handle the virtual design, even leaving the climbing on the bed idea, that she was very excited about initially, unfinished (see Figure 19) In an interview, Tegan described the group's working style and her own role within the design.

Tegan	Well, at the beginning when Ted wasn't here you know, I just did all of that (*programming*) and when he got back and I was working on the monkey I just let them do whatever they wanted to add to it.
Facilitator	So before Ted came you think you did more of the programming?
Tegan	Well yeah. Because that was all we were doing. Then when we started making it (*the physical pet*), I just did it.
Facilitator	So did you and Rocky work on the programming stuff before Ted came? Or was it mostly you or mostly him or?
Tegan	I like did it. He just told me if he wanted to add something, I'd do it. I guess. Cause he doesn't really know how to do that good.
Facilitator	Ok. So you think.
Tegan	Well, maybe he does cause then him and Ted made him like party and stuff. Cause when I wasn't here, they were working on it. So.
Facilitator	Yeah.

Tegan	Yeah, they figured their own stuff out.
Facilitator	Yeah. Why do you think your team broke up the responsibilities like that? Why were you the only one that did the crafty part and.
Tegan	Well, cause they're not really crafty and also they'd just mess it up. Cause I had an idea in my head.
Facilitator	Ok. And why do you think you didn't do as much programming once Ted came?
Tegan	Just cause I was working on the thing (*physical pet*) and I wasn't going to do both at once. You know?

Tegan talked about how she had done most of the programming before Ted came back from jail and then, when he appeared, she was already working on the physical pet and so she let them "do whatever they wanted" to the virtual Scratch program. When asked why they split up the responsibility like that, Tegan explained that the boys were not adept at programming, but then she retracted her statement admitting that maybe Rocky and Ted were good at programming since they had made the monkey "party and stuff". "They figured their own stuff out," she said. Tegan chose to work solely on the physical pet because the boys were "Not crafty" and "They'd just mess it up. Cause I had an idea in my head." Tegan seemed to have a feeling of responsibility to do the crafting for the team because she deemed her group mates to be not crafty. But importantly too, she added that she had an idea that she wanted the opportunity to realize, she did not want the boys to mess it up. The physical monkey design was Tegan's alone.

Figure 19. Tegan and Rocky working in parallel on different components of the monkey project. Tegan works on the physical monkey and Rocky on the Scratch program.

After conceptualizing the physical pet prior to beginning the crafting of the monkey, Tegan grew very attached to the physical monkey during the design process. She showed her pet off to other students, teachers and the principal multiple times during the project. In an interview, Rocky talked about how the group had broken up responsibility by giving Tegan sole ownership over making the physical pet. He admitted that this arrangement made the group get along better because Tegan would "Get mad at us if we tried touching her monkey." For example, in the following excerpt, Rocky tried to assist Tegan with attaching a felt face piece to the already crafted and furred monkey

head, during workshop 8. Tegan did not allow him to do so. She followed him around the room and forcefully took the pet back.

Tegan	Lemme see this (*the monkey)*.
Rocky	(*grabs the head of the monkey)* Is this good?
Tegan	(*reaching for monkey)* No I have to fix it.
Rocky	This? (*grabs felt for monkey face that Tegan has been cutting off of the desk)* No you don't. (*grabs stapler off desk. aims to staple felt face to fur head)*
Tegan	Wait. No. (*gets up from her seat)*
Rocky	(*walks toward front of room with monkey and stapler)*
Tegan	It's not perfect! (*chasing Rocky and monkey around room)* Gimme it! I'll make a new piece. Gimme it. (*catching Rocky)* You stapled it already! (*returns to her seat)* You know I'll just pull it off. I can't believe you're doing this.

In this episode, Tegan said things like "It's not perfect!" and "You know I'll just pull it off" about Rocky's attempted contribution to the physical pet. She insisted on affixing the felt face herself, even though she had already designed and cut the felt. The

episode suggests that Tegan was not willing to share the responsibilities of physical pet creation with her partners.

Tegan's monkey was very professional looking, almost like a commercially available stuffed animal (see Figure 20) However, Tegan spent so much time attending to the monkey's appearance that on the final day she was forced to glue the PicoBoard onto its back in plain sight because she didn't have time figure out how to embed it. The other students in the project took note of Tegan's work. During workshop 11, Ted commented, "You're so talented Tegan. You just whipped that thing up like it was a birthday cake." Tabitha, in an interview, talked specifically about Tegan's monkey and how successful Tegan was at the project. When the project was over, Tegan asked to keep her monkey, even without its interactive components, as the PicoBoard had to be returned.

Attachment and connection to a physical design is precisely what I hoped would occur with DigiblePets, drawing young people into computing through emotional connections to known interests and hobbies (Eisenberg et al., 2007; Resnick et al., 1996). Tegan shifted her time from virtual pet development to physical pet creation once the craft materials arrived. Tegan showed off her monkey to others, was reluctant to let her partners contribute to the physical design, and wanted to keep her physical pet. This suggests Tegan was emotionally and personally connected to the creation of the physical pet.

For Tegan, her relationship to the physical design component was so powerful that she ignored computing after having been quite interested and successful at it initially.

Contrary to expectation, Tegan's affiliation with the project did not lead her to continue to explore programming concepts but instead effectively derailed her computational learning by shifting her attention away from computing entirely. The tangible craft held too much attachment for Tegan, preventing her from participating in both areas of design.

Figure 20. Tegan's monkey.

Also, Tegan was not able to work collaboratively with the physical design because of her inability to relinquish control over the product. In an interview, Tegan

talked about how making a digital/tangible pet influenced her decision to join the project because she thought programming would be boring but making an interactive physical pet sounded interesting. Therefore the tangible craft provided a way for Tegan to connect to programming and was effective at getting Tegan interested and engaged in computing. But then the tangible pet became a powerful draw, luring Tegan away from the programming she was interested and proficient in. What had been expected to be a way into computing for young people who are interested in the craft components was actually a way out of computing in this case.

Jamal

Jamal was a senior student who worked alone on his project, saying on the first day of the project that he did not need anyone else. Jamal was tall, lanky, and reserved. He had a nearly shaved head and a gold chain around his neck. Jamal dressed in baggy shorts and oversized single-colored t-shirts with new looking athletic sneakers. From the beginning, Jamal always seemed very occupied with whatever he was working on and not easily distracted. While the other students seemed to constantly be interacting with one another, Jamal kept to himself often with large headphones hanging around his neck pumping gritty rap music towards his ears. Jamal listened to the gossiping of other students, but rarely joined in. However, by the final workshop (12), Jamal joined in several bantering sessions with the full class including talking about a cartoon rooster whose hair looked like a student he knew, hypothesizing about making a zipline from the high school to the alternative school and even engaging the principal in conversation

about her choice of clothing for the day. He poked fun of her by saying, "You're, dude, you're like a whole color outfit? Like one shade? That's cool. That's cool. Even your shoes, look. Pretty nice."

Jamal came into this his senior year with seven total high school credits, essentially one year's worth, in various subjects. This meant Jamal was trying to make up 14 year-long classes in addition to needing to still complete the seven classes required of seniors, in one year. As Jamal had already turned 18 when the project commenced, district policy dictated that this was his final year in high school (he would be moved to the adult education program after the spring). In workshop 5, Jamal came slouching into the classroom late, unusual for him, chattering about having to take a photo outside for getting a certificate for passing all his alternative school courses in the fall quarter. He said to me that he had never gotten an academic accolade in his life before this occasion and now he was going to be in the newspaper with his new certificate. From my field notes of our talk, I wrote, "All he (Jamal) really wants to do is get off probation. He says he's been on probation since he was 13. He still needs to keep up with the counseling but he feels like he is making progress. I asked him about college and he said he wasn't sure" (M. DuMont, field notes, November 8, 2011).

Jamal's family life was troublesome. His mother and father were both, at the time of this project, serving time in prison and his Grandmother was raising him in a remote location within the large rural district. Jamal had trouble getting to and from school saying that the bus ride took over an hour each way. Jamal spoke a little about his past, talking to me in interviews about several "mistakes" he had made and the consequences

including mandated weekly counselor and parole officer meetings. In class Jamal was quiet, bounced to his music and always greeted my warmly when he saw me.

Initially, Jamal solved six of the seven bugs on the two debugging task days. He abandoned the final bug without concern for leaving it unfinished. Jamal rarely asked for help, but when I would walk by his machine and prompt him about his work, he often needed assistance, but had not wanted to ask for it directly. Jamal was not afraid to change things as he went and to tinker with the programming code. For instance, he made some unique alterations to the prototype code on that first structured debugging task day (workshop 1) to make the car change colors and to change the backgrounds of the prototype Scratch project. In the following episode, Jamal worked during workshop 1 to fix two bugs (#3 and #4) and also changed the car to be yellow. During the process, Jamal talked to himself. He made only one comment to another team, "Whoa, my car changes colors."

Jamal	Nnnn, Nnn, Nnnn. *(makes car background yellow, clicks ok)* Ahhh. Alright look. I'm gonna put. Where's the negative sign? Ok. So. *(makes car go around)* Alright. (*fixed bug #3)* Whoa. My car changes colors. *(scrolling code, makes change to code)* *(tests eating, connects clips)* Boom! Right there! That's how you do it. (*fixed bug #4)*

When Jamal got a bug fix to work, he expressed his pride by saying "Alright." and "Boom! Right there! That's how you do it." He said these things to himself, as if he could not help but make the comments. He only told his neighbors that his car "changes colors" because it seemed he wanted them to know that he had made a unique code change.

In the three days of independent project work prior to the craft materials arriving, Jamal decided to begin with a blank project, instead of using the prototype code as a starting point like all the other students.[21] He imported and modified a "Wild Thing" creature from the Scratch library (see Figure 21) reminiscent of the characters from the popular children's book, and tried to program it to walk to the end of his outdoor forest scene, turn around and come back. He had significant trouble making his idea happen, but instead of exhibiting frustration, Jamal had a working pattern that included tinkering with the walking code, running into a bug, trying to tinker around the bug and then, when not successful, moving to another aspect of the project. Jamal worked for a time on walking. Then, stuck, he tried to figure out how to import a picture from the internet, which was not allowed because internet access was prohibited by the district. He worked on walking some more. Then he worked on some sounds. Finally he freehand painted a tree (see Figure 22) a meticulous and precise process including lots of erasing and revision, to use as a stimulus to cause the Wild Thing to turn. Many of Jamal's ideas went unfinished. With Jamal's haphazard approach (see Figure 23 for Jamal's walking code), it

[21] One other student group ended up using a blank project without prototype code, but this did not happen until later in the project.

seemed he was not fully connecting to the project. Jamal's code did not work for a number of reasons that could have been mitigated had he used some prototype walking code, which he chose not to. For instance, the "wait until" command has no qualifying statement, so it would not work. Also, the motion commands said "walk 20 steps" and "go to x= 52 y=-68", meaning the sprite would walk a short ways and then appear at the coordinates given. This was not the walk down the stage, turn, and walk back that Jamal had mentioned wanting to achieve. During the rest of the 40 minute workshop, Jamal added a wait command and then removed some of the initializing blocks of his code. When compiled, the code never did anything at all.

At the beginning of workshop 5, Jamal said out loud, "I don't really like my dude." For the first six minutes of class, he proceeded to work on and show off his tree to another student, saying "Unn, check out my tree! Yeah Sonny!" Then during minute seven, Jamal deleted his Wild Thing sprite, along with all of the code he had generated. He did not make any verbal remarks when doing so, just began to look for a new sprite. When asked in an interview about why he deleted his entire project he said, "I guess I just lost interest." At this point in the project, Jamal stopped creating programming code and focused solely on the aesthetics of his design, taking time to make sure the new virtual pet looked the way he wanted.

During the next three workshops (5 through 7), Jamal worked diligently on parts of the project that mattered to him personally. He was not satisfied with his original character or project. So, Jamal imported a new sprite, a unicorn, and spent a long amount of time, over 30 minutes over workshops 6 and 7, painting sunglasses that he referred to

as "Stella Shades" and sneakers, referred to as "Nike 6 point 0s" (see Figure 24) in Scratch. These two accessories seemed to carry personal importance for Jamal. Jamal wore athletic sneakers similar to the sneakers he was painting in Scratch many days to class.

Figure 21. Jamal's Wild Thing walking in the woods.

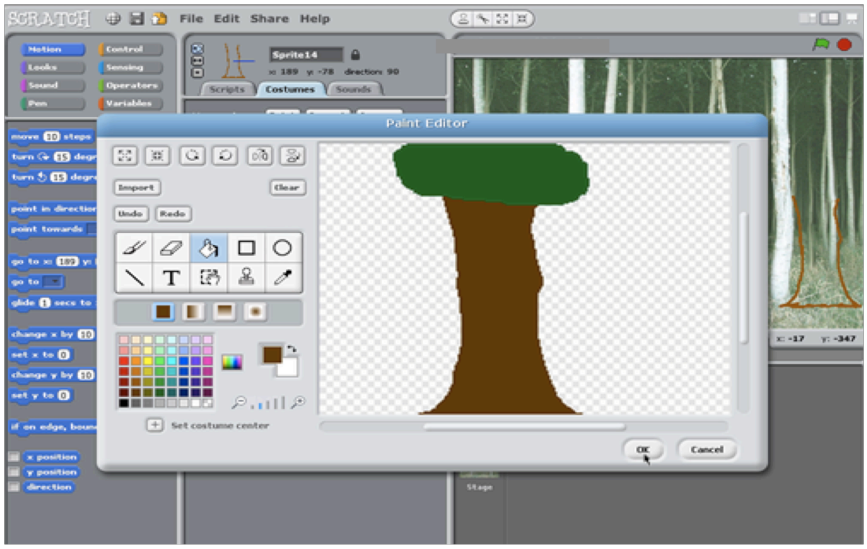

Figure 22. Jamal's hand-painted tree in progress.

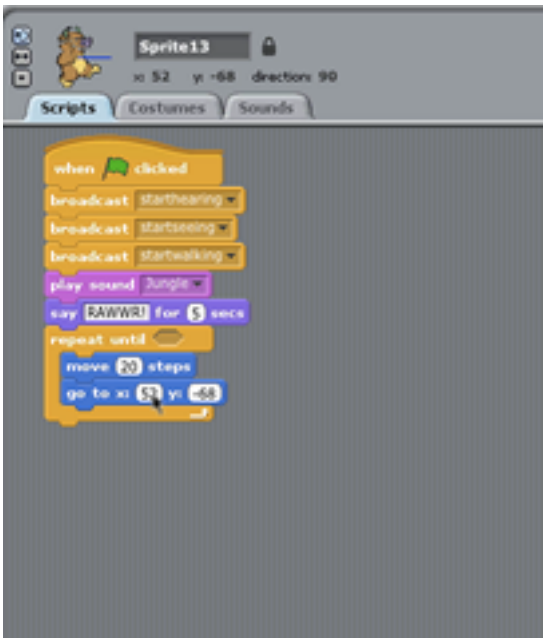

Figure 23. Jamal's walking code that never worked correctly.

Jamal worked without talking much to others and without breaks, sometimes quoting the music in his ears out loud. At one point he got very frustrated that one of the sneakers he had been working on looked like a high top, when it was not supposed to be a high top shoe. In the following episode, after over 20 minutes of creating the shoes, Jamal believed his last sneaker looked too high, but he was not sure how to fix the problem without erasing part of the zebra's leg as well. He figured out that he needed to zoom in and recreate the zebra's leg at a more pixilated level (see Figure 25).

Jamal	Shit. That sucks dude. Hey if I put eraser on the zebra, it'll erase him right? Oh yup. Dang it dude. I just hafta like erase the black, cause they're too high. They can't be high tops.

For Jamal, it was important that the sneakers "can't be high tops." When I asked him that day about his project he declared that he had no scripts (programming code) "Mostly because it took so much longer to make it (the accessories) look awesome on the computer screen". He wanted the relevant pieces of his project to "look awesome" and was willing to put in the time and effort to make that happen. He then declared that he would not be finished with his pet by the end of the project.

Jamal's transformation from a programmer who seemed satisfied to place code without fully understanding how it would work, to a dedicated designer continued throughout the next several workshops. For workshops (8-10) Jamal designed his physical pet, a purple felt creature with big eyes and zebra skin stripes. He built the pet

around a curved pie pan, found by Anna in one of recycling bins in the teacher's kitchen, with seriousness of purpose.

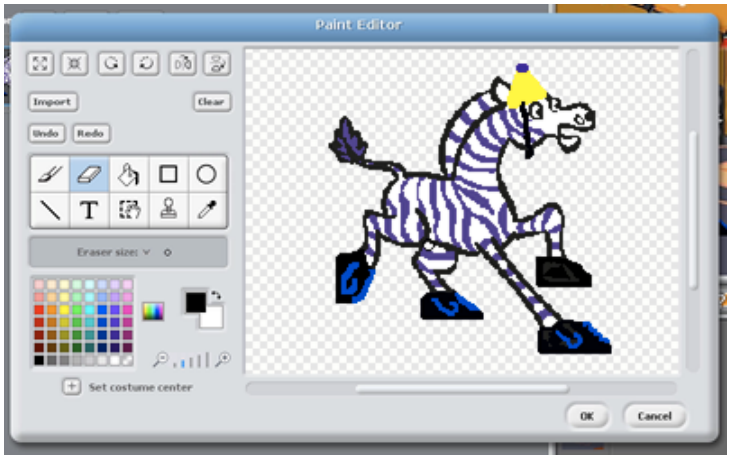

Figure 24. Jamal using the paint editor to paint Nike 6.0s for his unicorn.

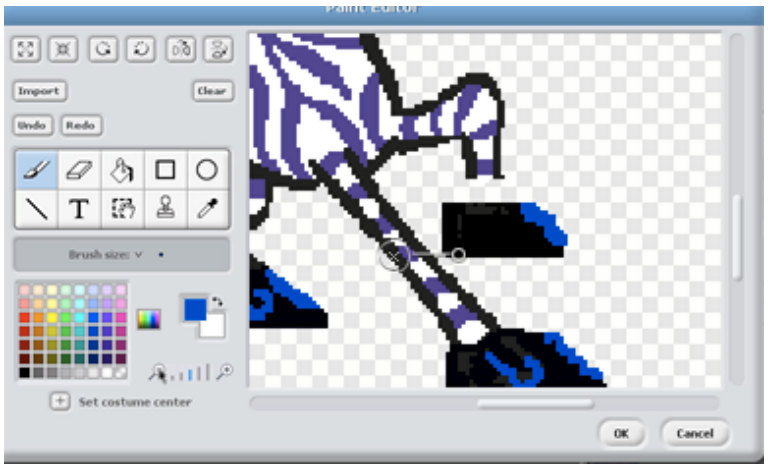

Figure 25. Jamal zooming in to fix his high top problem.

Jamal hunched over his project, and rarely gossiped with the others (see Figure 26). Notice Jamal chose not sit facing the other students in the class. Instead, Jamal situated himself at the end of the table, looking towards the wall. Although the original intention was to create the unicorn with the incredibly detailed Nikes from the Scratch project, Jamal's physical creature took on a persona of its own and became a zebra.

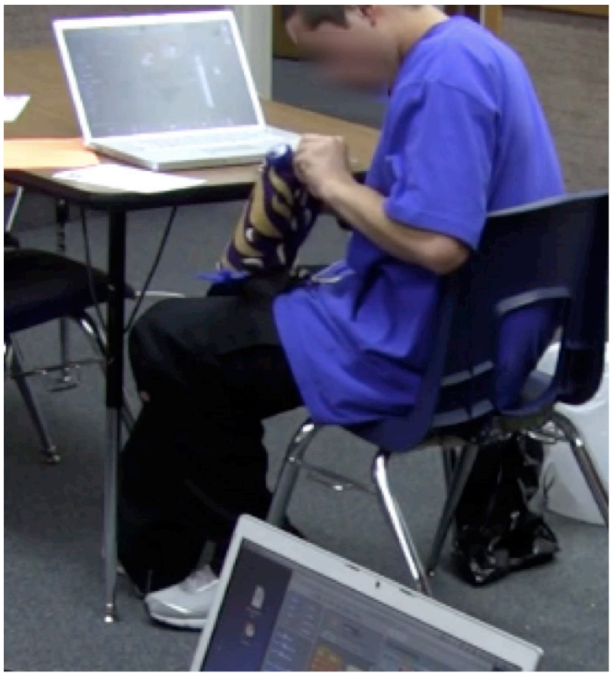

Figure 26. Jamal working on his tangible pet design, facing away from all the other students in the class.

Jamal's physical pet was very deliberately constructed (see Figure 27). He spent three days of concerted effort crafting the pet and devised a way to embed the PicoBoard to allow users to interact with the buttons and sensors without altering the pet's appearance. Some other groups, like Tegan's did not embed the board at all and others like Carlos' had trouble embedding their board and required continued support to get past bugs and design flaws. At the end of the tangible design phase Jamal declared, ""Yeah! I got my little guy! Unnn. Done. Little man."

Figure 27. Jamal's finished zebra with eye pointing downward and tail up, the PicoBoard in embedded beneath and not visible.

As a result of the physical pet becoming something unintended, Jamal had a mismatch between his Scratch sprite, the unicorn with the sneakers and sunglasses, and his tangible pet, a zebra. After realizing the issue, Jamal spent a considerable time during workshop 11 painting a new on-screen sprite that matched the physical zebra exactly (see Figure 28). When finished painting his new sprite, Jamal said to himself, "Almost total likeness. Yeah. Yeah. He's pretty tight." No other group paid so much attention to the exact replication of their two designs. In fact Tegan's group decided that having monkeys that looked different did not really matter and Tabitha's group, save for a couple head feathers, had a completely different creature in the real world versus on screen. Instead of deleting the other character that Jamal spent so much time making aesthetically relevant, Jamal integrated the unicorn into the background of his dancing scene to make the scene more authentic.

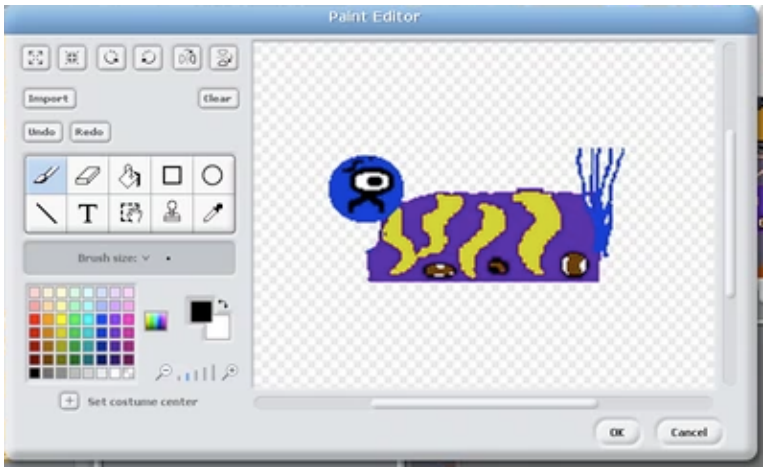

Figure 28. Jamal's hand painted zebra sprite.

On the final workshop day, Jamal began programming again. I had not forced him to program earlier, although I reminded him during several workshops that he needed to think about coding at some point, because Jamal appeared very invested in his physical pet development and I had not wanted to divert him. No other group spent so long without programming any code for their pet. Recall that Jamal played around with some characters and functionality and then deleted it all because he did not really care for the character he had made. This time, Jamal again used a blank project and not the prototype project I provided. It seemed important to him that everything was his own idea and own implementation. At the end of class, Jamal was not satisfied and told the classroom teacher, "I'm not even done, sorry, I've gotta stay here." He stayed for an additional 30 minutes after class was over to complete his program. No other students ever stayed late.

During the final workshop, Jamal ran into four bugs and resolved them by tinkering his way through the problems and asking the facilitator for coaching. He was the only student to resolve every bug he encountered on his independent project; he never ignored, worked around or left a bug unsolved. Recall Jamal did leave bug #7 from the structured debugging task unsolved without any consternation, however he did not do this on his own project. On average, groups tinkered, meaning played with changing bits of code to fix a bug, 14% of the time whereas Jamal tinkered 57% of the time when he encountered a bug. This difference is explained by Jamal's playful approach to his work and his dedication to the parts of the project that were personally meaningful. For example, Jamal, after asking about who might appear at the design exhibit and hearing teachers and administrators as the response, spent upwards of 20 minutes taking samples

of different music from his iPod to find the most pleasing song fragment with the least offensive language. Jamal did not want to have any offensive language that his classroom teacher had already disapproved of. The functionality Jamal implemented included the zebra dancing to music using different costumes repeated in succession, responding to Jamal's voice, doing backflips when the slider was moved a certain way (see Figure 29), and speaking when the button was pressed. Using the slider to control the pet's back flipping in succession was a unique bit of functionality. No other group explored any way of using the slider besides walking.

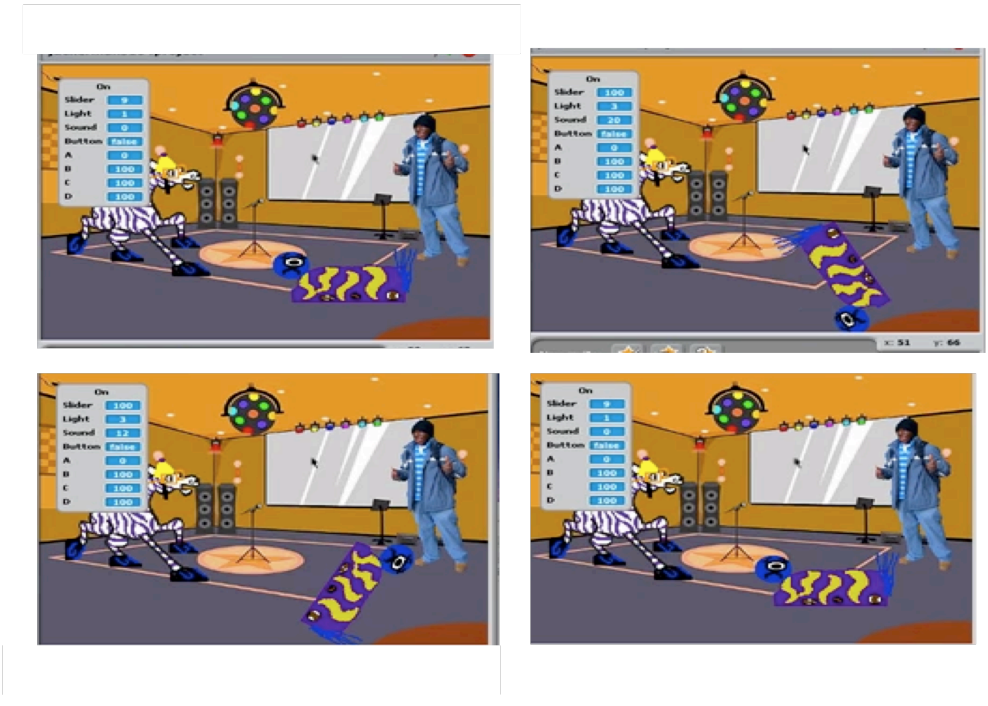

Figure 29. Jamal's zebra doing a backflip in the dance club.

Jamal then showed off his pet to an outside student who came to visit the class, explaining all the parts proudly. As Jamal was cleaning up to leave that final day, I asked him "How'd it turn out? Do you like it?" He responded, "Yeah, it's alright." This from Jamal was a very positive reaction.

Jamal had a shoddy academic history and was very reserved in the beginning of the project. However, Jamal came to the design exhibit despite living so far away that he was not sure how he would get home. Only four of the nine total students attended the exhibit. He was the only student to bring a guest to the event, a friend of his. This show of dedication was unusual for Jamal and speaks to his connection to the project. The following excerpt from an interview describes how Jamal felt about the project as a whole.

Facilitator	How did this project compare to what you normally do in school.
Jamal	It was pretty tight. Normally I do, normally school's like hella lame. But this was pretty fun.
Facilitator	What's lame about regular school?
Jamal	Like everything. (*laughs*) What do you mean what's lame about regular school?
Facilitator	Like what kinds of things do you do? What do they make you do?
Jamal	Like all book work and stuff. It's like DT (*juvenile detention*) kind of. Be all quiet. Can't talk. I'm surprised they don't make you walk down the middle of the hall with your hands behind your back.

Facilitator	That's not a very glowing review of regular school. And then what was different about doing this?
Jamal	I don't know it was fun. We just got to take it and go with it. It was kind of like a project. It was hands on. And we got to make things.
- - -	
Facilitator	What was it like working on the project?
Jamal	It was fun. Like when I first started making my own I didn't really know where to start so that's why I kind of didn't do anything for a while. But then once I figured out what to do and everything, it came together.

Jamal claimed regular school was stifling, boring and most often students had to "be all quiet". In contrast he found the project to be fun and intellectually interesting. He said, "We got to take it and go with it". He enjoyed being able to decide how and what to build and create. For Jamal, the beginning of the project was somewhat difficult to relate to. He said, "I didn't really know where to start so that's why I didn't really do anything for a while". He was referring to the period where he deleted all his code. But it seemed Jamal was able to make a personal, culturally resonant connection to his pet through painting accessories for his sprite and creating the tangible pet that grew into a zebra. Other studies show young people making culturally resonant connections to computing through developing multi-media designs, like youths' music video creations and 'low rida' interactive art projects in Scratch (Peppler & Kafai, 2001).

After Jamal developed these artistic parts of his project, the rest of the project took off as well. In Jamal's case, the physical pet creation combined with being able to customize his project to reflect the things he liked, the shoes and shades, in real life

seemed to allow him to discover something relevant and personally meaningful in programming and design. The tangible aspects of the project were important; in an interview, Jamal said he signed up because "It was more hands on and I'm into hands on." However, it may have been even more important to have the freedom to pursue interests, how and when he wanted to. Jamal appeared to use time and freedom to learn and explore to connect to the project in a way that engaged him profoundly, but once he discovered that connection, he was dedicated, effective and successful. Personal meaning realized in a combination of both tangible and virtual media appeared to provide Jamal a way to connect to the project, whereas just one medium alone may not have.

Carlos

Carlos was a Hispanic student with a heavy accent when he spoke English. He was a junior with a car. Carlos never stayed in town long, going on extended trips, sometimes a month long, to visit his girlfriend in California, where he claimed he would go to college and live off his parents' money. Carlos had dropped out of the traditional high school, for reasons no one explained to me, and then left town. He was one of the students the principal of Winder was particularly excited about because she heard about him having left the high school and personally tracked him down at his house to convince him to attend the alternative school. She referred to him as "so smart". Carlos declared to me one workshop that he ate two egg sandwiches chased by a Monster caffeinated energy drink from the local gas station every morning, saying they were delicious and necessary to survive school.

Carlos's story provides insight into a different way of connecting with the DigiblePets project. Carlos worked with Dino and Maya, the other Hispanic students participating in the project. The threesome seemed to be friends before the project began. They spoke and joked a lot with one another, much of the time giving Maya grief about her boyfriend, another friend of theirs, talking about her pregnancy, or talking about electronics. Carlos began the project wholly interested in programming and ended the project with expertise in only that discipline. In an interview, Carlos claimed his interest in the project stemmed from an interest in fixing computers for his friends and family. He was the only student to mention programming as the sole reason for participating. According to his survey, Carlos signed up for the project because he felt confident and capable with computers even though he had no programming experience. He declared about computers, "It always comes easy to me."

Despite being absent for the first introductory workshop (workshop 0), Carlos jumped into the computational aspects of the project right away. He instantly took over control of the computer from Dino (Maya never had control of the computer, touching only on two occasions, when Anna asked her to run the group's program because none of the other group members were in the room at the time, which she could not do, and when Carlos told her to paint a new version of their sprite alien with many more eyes) and the direction of the group's pet development. At first Dino assisted with programming problems, but after the third workshop, Dino lost interest and stopped offering advice. Carlos's pet, for the functionality was all Carlos' doing, had many intricate features and lots of programming complexity. At the end of the project, the alien could do the

following: make alien noises, put on sunglasses, wait for permission to ride a magic carpet, ride the carpet, get off the carpet, jump on a trampoline to a different world, differentiate between being fed "food" or a person's hand in the physical world, and walk around, all based on interactions with the PicoBoard (see Figure 30) Carlos worked to create all the different functions for his pet and refused to dismiss any of his ideas, even when his partner Dino told him it would be easier to do so during workshop 9. Carlos' project included myriad programmatic changes to the prototype code. Upon getting a piece of his magic carpet idea to work, Carlos said, "Alright, I figured it out." He seemed to take pride in his ability to create complex code and functionality.

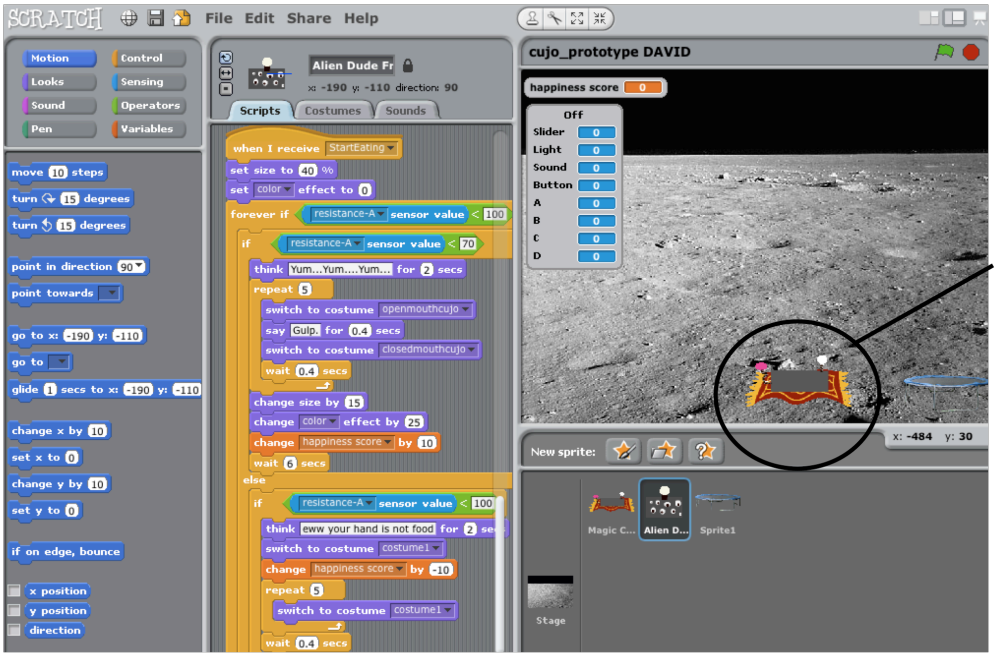

Figure 30. Carlos, Dino, and Maya's alien Scratch program.

When it came time to create the physical pet, Carlos told Maya and Dino that they were "the art people" and dictated they do the work. He rarely touched the physical pet or the rendition of the pet in the virtual space (see Figure 31 for the group's working style). During workshops 8 and 9, Carlos declared himself finished with the programming part of the project and proceeded to watch Dino and Maya as they struggled to finish the physical design. Dino and Maya also worked on user interactions, embedding the PicoBoard into the pet's body and frequently had to ask the facilitator for help. Carlos spent the first half of workshop 9 fixing a phone that Maya brought to class, claiming she found it in the street. When Anna, the classroom teacher came by to see why the group did not appear to be working on the project, Carlos stated that they were finished, even though the PicoBoard was not yet installed in the pet. Anna asked Carlos if he was interested in going to the other room to do his coursework considering he was finished with his pet. Then Carlos insisted they were not finished at all. This episode is representative of Carlos' way of being authoritative, by declaring something that may not have been true but quickly recanting it, if that was in his best interest.

Instead of helping his group Carlos reported that he "did all the programming" as if fairness dictated the other group members should be responsible for something. Carlos all but refused to be physically involved with the crafty and art-like parts of the project, only using the scissors to help them in the last few minutes of class. Instead of being physically involved, Carlos preferred to sit back and bark out commands to the others, sometimes making disparaging remarks, and acting as spokesman whenever an adult came by to ask how the group was doing. For example, during workshop 9 Dino and

Maya physically manipulated the craft materials to embed the PicoBoard within their cardboard box alien, a task that was more difficult than expected because of the multiple items, sensors, buttons, and sliders that needed to be accessible simultaneously. Carlos spent the time making a video of the others working on the phone he had just fixed for Maya. He also offered the group unsolicited advice but let Dino do all the construction work.

Dino	It works, see? It's not like they'll be checking the bottom. You know?
Maya	Maybe they will.
Dino	Says who?
Carlos	You gotta glue that (the paperclip mechanism they created to access the slider) to the side so it stops moving.
Dino	MmmHmmm. Gotta tape it to the side now. Somehow. Let's stick the tape through here and stick it to the side right here.
Carlos	Yeah. That's what I said.
Dino	(*only one touching pet*)
Dino	Duct tape fixes everything.
Carlos	That's not duct tape though.
Maya	It's ok. It's pretty tape.
Carlos	For anyone who's watching, that's not duct tape cause this kid is stupid.

Dino	(*pushes Carlos*)
Carlos	And she's like confused. (*points camera towards Maya)* She doesn't know what's going on.

Carlos was vocal about his perception of his artistic abilities, saying, "I suck at art" and "It's just that I'm not good at art and they are." Yet, when faced with the other two doing the work, Carlos's banter is pejorative and authoritative. He told Dino, "You gotta glue that to the side" and then when Dino explained how he might accomplish that task while inspecting the alien, Carlos responded, "Yeah, that's what I said." When Carlos did not say anything about how to "glue that to the side". Following that, Carlos made several comments like "That's not duct tape though", "This kid is stupid" and "She doesn't know what's going on", that are neither helpful nor nice. This sequence is representative of Carlos' interactions with his group mates. He often displayed feelings of superiority, commanded the others to do certain things, and made remarks that reflected poorly on the others.

Carlos was a confident programmer. He referred to his pet as "the best one" and to his own programming skill as "the most advanced". However, when faced with bugs, his group encountered the most bugs of any group (50% more than the group with the next most bugs) [22] due primarily to the complexity of their project as a whole, Carlos never tinkered, he either implemented a direct solution idea or required assistance or coaching

[22] 39 in total versus, 26, 22 or 14.

to fix bugs.[23] When talking in an interview Carlos said about his personality, "If I get something wrong I want to know why so I can get it right the next time." This sentiment was reflective of his debugging style where he normally asked the facilitator; he wanted not just to fix the bug by trial and error but really understand what went wrong. On day 4, a particularly busy programming day where Carlos was implementing trampoline functionality, Carlos worked through 10 bugs total, a large number for student groups during this project.

Figure 31. Maya, working on the many-eyed alien, Carlos, touching the computer, and Dino, looking at the Scratch program.

[23] See Chapter 6 for more details on bugs.

On average student groups asked for help from the facilitator as 42% of all debugging strategies. In contrast, Carlos asked for help 93% of the time (see Chapter 6 for more details on student debugging). Additionally, Carlos received coaching, a step-by-step method of providing assistance with bug fixes in 50% of cases whereas on average, students received coaching in 17% of all debugging strategies. In an interview, Carlos reported he was happy with the project overall and how his group's pet turned out. About the project as a whole, he stated in an interview, "It was fun" because "I learned how to program it and I got to mess around with the computer".

For some designers of computational crafts, getting youth to participate in computing is the goal, meaning a student who comes into the project with an interest in programming is already on the hoped for path. The notion that the hybrid design technology did not hinder Carlos' ability or interest in pursuing programming should perhaps be heralded. However, for other computational craft designers (DuMont & Fields, 2013; Fields et al., 2012), computational crafts should not just provide individuals with an interest in crafts with experience in programming but vice versa as well. One goal of the DigiblePets project was to broaden student participation in ways of thinking and design in both physical and virtual media and the interplay between them, reflective of the latter notion. Therefore the fact that Carlos, quite successful as a programmer, did not participate in multiple aspects of design is not entirely desirable. He began the project believing he was not good at art and finished the project with renewed faith that he was good with computers but without any increased exposure to art. To be more aligned with the project's goals, Carlos's interest in programming would have translated to a

willingness to engage more fully with the more artistic parts of the project. Although Carlos' case sets parameters for greater success, he was deeply engaged in the project and gained experience in programmatic thinking. Other students struggled to engage with the project in any discipline.

Hybrid Design Technologies and Limitations on Engagement: Dino

For some, hybrid media did not provide access to either artistic or programmatic design. For a small subset of students, lured by one or the other discipline or simply willing to give the project a try because of the promise of elective course credits and a seemingly new way of learning, struggled throughout the project to find a meaningful way of connecting. For instance, Dino, who reported in an interview that he joined the project because he liked art and wanted to build things, also said he could not get past his dislike and distrust of computers to make a meaningful contribution to any aspect of the project save some work embedding the PicoBoard into the physical pet. The stigma Dino attached to computers ran deep. For instance, whenever anything went amiss, Dino immediately provided commentary like "I hate computers" and "you can't tell them what to do". He used his views of computers as an excuse for not participating more fully in the project.

Dino was a small Hispanic junior with a moderate accent when he spoke English. Dino would often speak to Maya in Spanish and refer to Carlos with Spanish expletives. The teachers I spoke to about Dino were concerned about his lack of engagement in any aspect of school and his tendency toward insubordination. Dino's academic past and

personal history were elusive, but he spoke occasionally to his group mates about making money by "finding" electronic equipment (phones, Ipods, game players) in various places like inside other people's backpacks, as he stated during workshop 9 to his group, and fixing them up to sell to others. He was often sullen and not very responsive to me during workshops. When we talked during one workshop about what some of the students might be interested in pursuing after they left Winder, Maya said she would like to become a pediatrician, Rocky a diesel mechanic, and Steph a record store owner. Dino said he would like to move to California to open a pharmacy, the kind that could sell marijuana legally, and make a lot of money.

In our interview, Dino said he considered himself to be an art person. He enjoyed art at Winder and claimed his favorite class was an ACAD, a software program for technical drawing, design class at the traditional high school. Dino was very forthright in our interview and gave me great insight into his thinking about school and life. I appreciated his unconstrained opinions and willingness to share them. Along with his hatred of computers, Dino was vocal about hating math. In particular, Dino described having to do math problems exactly the way teachers told you to even when you got the right answer in a different way as stifling and bad for humankind. In the following excerpt from our interview, Dino discusses in depth how being told exactly what to do in math threatens human creative thinking.

> "Because it's like. I hate it how you, it's always like the same thing. Teachers teach in a way and supposedly it has to be done that way. A lot of students think it's got to be done that way. And they figure it out and they all do it that way. Cause one time I got this same answer in a different way. And the teacher said it was wrong because it was in a different way. And I was like "Why if it's the same answers?" And he's like "But it's this

> rule in math". And then I'm like "But why do we have to do that rule in math?" You know? And he said he doesn't, he didn't even answer. He's like "I don't know, that's just how math is." So I thought that was kind of gay because it was the same answer in a different way. And he said it was wrong. So like I hate it how everybody, if they're taught in a way they all try to figure out in that same way and they always follow the same order, always, always, always. Well, if you tried out something different then a lot of things would be different. For example if we all figured out math the same way we would all think the same, wouldn't we? So if we all try different ways then we would all have different ways to do things. Like if we all thought the same way in drawing we would all have like the same ideas, the same drawings, the same paintings. But if we had different ways to think about it we would think about different ideas to draw, different paintings and the world would be different and with more variety and more things to choose from and not just the same thing."

Dino's insight into the importance of creative mathematical thinking and its parallels to the importance of artistic expression is poignant, especially for a student who was widely seen as unsuccessful. This excerpt opened my eyes to Dino's way of viewing the world and made me understand why Dino was having so much trouble with the traditional school system. He wanted an opportunity to realize his expression and was not given ways to do that productively and was able to articulate that need fairly sophisticatedly. This interview made me realize that Dino was struggling with authority in situations where authority made no sense to him and that he was indeed a thoughtful person.

During the project, Dino was most of the time very subdued and pessimistic. When I came by to help the group, Dino often made an excuse to visit the bathroom or leave the area for other purposes. For instance, when Carlos summoned the facilitator on one occasion during workshop 4, Dino got up and wandered around the room until both the facilitator and Anna, who came by to see what was going on, had left. Then the next

time the facilitator came to help, Dino declared, "I'm going to throw this away" and left the room again. When a bug occurred, during that same workshop he said, "See you can't work with computers. They don't do what you want them to." Once the craft materials arrived, I thought Dino would become more involved with the project. However, save for workshop 9, where Dino helped Maya embed the PicoBoard such that all the buttons, sliders and sensors were accessible beneath the alien, Dino did not participate in the creation of the alien. When I asked him during workshop 5 why he was not interested in developing the physical pet, his reason was because, "It's weird". Carlos insisted, "You're going to make it." To which Dino responded, "No. I'm done."

On the debugging task assessment at the end of the project, Dino refused to complete the debugging questions after spending a minute or two on the first one. He said, "I don't know. I'm just not even going to try. I hate computer shit." During the same interview he declared there were too many numbers in Scratch, too much complexity in programming and that programming was boring. These comments were reflective of my observations of Dino throughout the project. He thought the project would be about pressing buttons to make simple changes to the pet but not about programming. Similarly, Dino chose not to participate in crafting the physical pet because he "thought it would be different". He believed the physical pet construction would be more like industrial design and less like crafting or art, which he said he liked earlier in the conversation:

> "That was I think because all we had was like a box. But like, I like to, I thought we were going to build things because when you said we were going to build our own thing and then we were going to build it like, because I like using, I like building things. I like working with my hands

> you know? Like you, I thought we had to build like an actual Cujo, but not build it with like tape and glue and like that. I thought we had to build it using like scissors and wire cutters and wires and sticks, stuff like that."

It seemed like aspects of the programming had gotten Dino so disenchanted with the project that he was then disinclined to give the more artistic physical pet design a chance. He could easily have used scissors and wire and sticks to make the physical pet. Other students used pie plates, balloons, cardboard boxes, pipe cleaners, cork and other found objects. There were no limitations to the materials or creativity of the construction. Clearly, for Dino, the project was not successful in encouraging him to participate in design or development in the way I would have wanted. Dino's frustration with and the reinforcement of his prior perceptions of computers seemed to influence his thinking in negative ways, manifesting itself in avoidance of nearly all aspects of the design project. Hybrid design technologies did not provide an effective way for Dino to engage in design, but also worked to negatively manipulate his already tumultuous relationship with computational technology.

Although I observed that the hybrid design technology increased the possible ways young people could engage in computing and artistic design, many of whom would not otherwise have participated in a programming project, the hybrid design technology did not necessarily provide a way for everyone. For instance, Dino thought he would be interested in some aspect of the design project, but then was not. Some of this could have been mitigated in part by the design of the project as a whole, but the observation begs a reevaluation of how to envision, design and use computational craft technologies to best broaden participation in computational disciplines and artistic design. The simple link

between art and technology in the design of a relevant artifact is not always enough to promote young people to participate in and engage meaningfully with programming and physical design.

Summary

In total, four student groups and one classroom teacher designed five DigiblePet projects. Carlos, Dino, and Maya's many-eyed alien was functionally complex, thanks to Carlos, but none of the group members attended the design exhibit to demonstrate the project to the community. Jamal's zebra was carefully crafted physically and represented virtually and lived inside a Scratch project that Jamal began from nothing on the final days of the project. Jamal and his friend came to the design exhibit. Jamal was the only student to invite someone who attended. Tegan, Rocky, and Ted's project had an elaborately constructed physical pet, Tegan's monkey that she took home with her. Tegan and Ted came to the design exhibit, but Rocky had to work, so could not attend. Steph and Tabitha struggled throughout the project, with ideas, attendance, their personal relationship and programming, but created a feathered physical unicorn that was a feathered hippo on screen. The pet had limited functionality because Steph accidentally deleted their code near the end of the workshops and the girls had difficulty making much progress afterwards. Tabitha came to the design exhibit and explained her troubles to the audience, who were very sympathetic. In the next chapter, I will explore how the students interacted with one another during the project and how these interactions contributed to how project designs unfolded.

CHAPTER 5

HOW STUDENTS INTERACTED

One of the main goals of the project was to use innovative means to allow students access to new domains of expertise and learning. Because the majority of creative artifact production occurs in a collaborative atmosphere (Sonnenburg, 2004), sharing and building upon ideas was anticipated to be a critical component to realizing this goal. In principle, hybrid technologies should be well suited for supporting interaction. Multiple individuals can observe the actions of others, make suggestions, and take turns producing and implementing solutions. Also, and perhaps even more critically, hybrid technologies combine multiple academic disciplines where students can have a sense of expertise. A student who does not feel as comfortable with programming could begin by expressing ideas about the tangible portions and from there, encounter and resolve problems that would naturally appear as they moved beyond physical structure to computational behavior. A student who was more comfortable with the programming would eventually need to refine her understanding by configuring and building the external sensing or response apparatuses. The open and flexible workshop time should allow students to engage in informal sense-making discourse and negotiation as they worked toward a shared endeavor.

However, the quality of interaction ultimately took on different characteristics, some of which were alluded to in the descriptions of individual students' experiences from Chapter 4. In this current chapter, I will describe the nature and frequency of student interactivity observed during the project. Specifically, much more interaction

about the project occurred between students during the structured task than during the open-ended design project. The dual nature of the hybrid technologies on several occasions seemed to actually support modularization over collaboration. Furthermore, there were some qualities of the population that may have mediated their ability to consistently collaborate with one another.

Why Should We Care About Collaboration And How Was It Encouraged?

Collaboration is considered important to a broad range of creative artifact production activities such as improvisation (Sawyer & DeZutter, 2009), creative writing (Vass, Littleton, Miell, & Jones, 2008) and even web-page creation (Fernandez-Cardenas, 2008). Similarly, collaboration is paramount in helping young people to develop *collaborative agency* (Kafai, Fields & Burke, 2011) because collaboration is thought to inherently inspire and improve product development (Vass et al., 2008) through participation, communication and negotiation (Fernandez-Cardenas, 2008; Sonnenburg, 2004). Studies have also shown young novices can learn computer programming better when working collaboratively (Webb et al., 1986). A meta-analysis of computer-based instruction in K-16 classrooms concluded that collaboration made computer-based instruction more enjoyable and motivating for students (Del Marie Rysavy & Sales, 1990). Yet, despite efforts to encourage collaboration, observation and field notes from this project highlight the fact that collaboration was a very limited component of students' group design projects.

There were deliberate efforts to support students engaging with one another. By working in and across groups, I aimed to have students sharing, building upon and negotiating ideas pertaining to the development of their shared design projects. Students were recurrently referred to one another when there were questions, asked to share their project ideas with each other and the class and verbally encouraged to come up with ideas together, especially in opener and roundtable sessions by me, the facilitator. In addition, the project was built upon reusing and modifying another's ideas, namely the prototype developed by the facilitator, the code from which was to be reused and modified by students for their own projects. The small group configurations were left to the preferences of the students so that they could, in principle, select students with whom they had rapport and would comfortably share ideas. Yet, the overall sense I had and recorded in my notes was that collaboration waned. To examine that, I proceeded to analyze in detail video records of student activity.

Operationalizing Collaboration

Collaboration can be thought of in a number of ways both verbal and non-verbal. During the project, students interacted verbally, through speaking exchanges, and non-verbally, by taking turns manipulating the external PicoBoard, pet, craft materials, USB cords or computer keyboard and mouse. For this analysis, collaboration was counted, as when a verbal exchange of ideas, or combination of verbal and non-verbal exchanges by students, occurred with a minimum of three separate turns. Also, the exchange must be in reference to the task at hand. For instance when a group was working on a specific

debugging task, talk and manipulations related to that debugging task was considered collaboratively relevant. In general Student One says something, Student Two adds his or her input or manipulates the technology in response, and Student One incorporates or otherwise responds to the contribution of Student Two's turn either verbally or through a nonverbal manipulation. Exchanges could be comprised of many more turns and extended until the interaction was interrupted or concluded. Those were still counted as a single collaboration episode. Three-turn exchanges of ideas in which one of the contributors was from the facilitator were not counted as collaboration between the students. However, if a three-turn (or longer) exchange between the students took place a few moments after a contribution from the facilitator, then that was counted as an instance of collaboration. Verbal (or verbal with nonverbal) exchanges were not counted as collaboration when one group member commanded another group member to do something and then reacted when the other member fulfilled his or her task because, although enacting the behavior could be seen as a turn, the responding student did not visibly share his or her thoughts or ideas. However, unsolicited behaviors were included as part of a turn.

The following excerpt provides an example of collaboration that fits the operationalized definition above and includes an example of how physical manipulation as well as verbal utterance was important to collaboration in this project. In the following excerpt, during the structured debugging task, Rocky identified an issue in the program – namely, they lost their sprite on the screen - after the pair believed they had solved bug #1. This first portion of the transcript was not coded as collaboration because there were

only two exchanges, Tegan talking as she changed the code and Rocky expressing to the facilitator that they had fixed the bug. The second portion of the transcript was coded as collaboration because Tegan and Rocky participate in four verbal/non-verbal exchanges that result in successfully testing the bug fix.

Tegan	(*scrolling through code)* Mmm. Oh hello. Ok. He should say pesky bug. *(finding "say Hello." command typing "pesky bug" in textbox)* Ok. Ok. We did it!	**manipulation computer** **referring to computer** **code change** **exclamation of success**
Rocky	I think we did it!	**calling Facilitator to view**
Facilitator	I'm coming.	
	- - -	
	(*trying to test program but they can't see the grasshopper on screen)*	
Rocky	We lost him *(the grasshopper)*. Push the green flag again.	**#1 -** **idea about virtual program**
Tegan	*(clicks red sign, clicks green flag to start program)*	**#2 -** **manipulation computer**
Rocky	*(moves pet to hopper using slider, pet says "pesky bug!")*	**#3 - manipulation PicoBoard**
Tegan	Yay! Ok, we did it!	**#4 - reaction of success**

Figure 32. The prototype Scratch code for debugging task #1 that Tegan and Rocky are engaged in. The collaborative exchange is considered as beginning with contribution #1 (from Rocky).

In this episode, Tegan, as the "programmer" made changes to the code so that the words "Pesky Bug!" would appear on screen instead of "Hello." (See Figure 32 for the Scratch programming code Tegan and Rocky were working on.) Meanwhile, Rocky closely attended to Tegan's activities on the screen even though he was not touching the computer or making code changes, observing, "We lost him" when the grasshopper moved off screen. He suggested a test of the code and directed Tegan to "Push the green flag again." Tegan followed Rocky's instructions while Rocky then decided to also make some adjustments to the slider on the Picoboard that allowed them to test whether Tegan's new code worked. Tegan also focused on Rocky's manual adjustments, alternating between her focus on the screen and Rocky's adjustments on the pet. Upon seeing the pet and hearing the new sound "Pesky Bug!", they were able to recognize that they had resolved the bug. The result was a success and Tegan exclaimed, "Yay! Okay, we did it!" The "we" in both Rocky's and Tegan's comments in addition to their attention

to each other's work on and off screen manifested the shared nature of their collaborative work.

Tegan and Rocky were both part of the group that built the Monkey pet and thus this was intragroup collaboration. Collaborative exchanges could also take place between groups. However, intergroup collaborations were quite rare during the project. During the structured debugging task workshops (workshops 1 and 2), the debugging task sheet I created stated specifically that students could seek out other groups for advice and help.

> "Here are Cujo's bugs. Your job is to figure out how to fix them. Save often. You can get help from other teams or ask me for advice. When you figure one out, wave me down so I can take a look at it. If I think it works, then write down what you did (the code you changed or added) on this sheet. Good luck!!"

Since all students were working on the same tasks with the same prototype computer program, it would have been very easy to share ideas across groups. However, only six instances of inter-group collaboration occurred, for a total of 6 minutes of multi-group interaction, and all of these exchanges included the same group (Tegan and Rocky) as one of the collaborators. Tegan and Rocky's group always took the role of the provider of knowledge and two other groups, Dino, Carlos, and Maya (group 1) and Tabitha, without Steph, (group 2) were on the receiving end. Rocky was especially interested in sharing his expertise, sometimes offering debugging advice, and subsequently giving it. He did this even when the other group did not make a request nor accept Rocky's offer to provide it.

Collaboration Analysis

Based on careful consideration of all project days, I selected four days for which all groups were fully transcribed for analysis. These transcriptions included utterances, physical and computational manipulations to the technology or pet (for example scrolling through code, changing code or pressing the PicoBoard button etc.), and student activity (for example leaving the room, going to get a piece of fabric etc.) Reducing the data to classes that meet certain criteria for the purposes of fostering productive analyses in this way has been seen to be an effective sampling strategy in other design research projects (Berland, 2011). It is also a practical matter as the amount of data collected for such a project can easily exceed the amount that can be fully prepared and analyzed within a reasonable amount of time.

The selected workshop days were workshop sessions where student groups were engaged in development work on their designs and would have occasion to share and build ideas. I further determined that days when students encountered a lot of bugs would provide insight into how those student groups shared and built ideas together on their projects, instead of capturing days where groups were gossiping most of the time for instance. I also wanted to compare what the bug heavy groups were doing to what the other groups were doing on those days. In the end, each group would have been productively engaged in developing some aspect of their pets for at least one of the days chosen.

The four workshop days selected were workshop 1, the first of the structured debugging task workshops, because all groups were involved in aspects of programming and debugging on that day. In addition to workshop 1, I selected the three independent

project work days during which each student group encountered the most bugs (see Table 6). For instance, Jamal encountered six bugs in workshop 1, the structured debugging workshop, and then between zero and two bugs per workshop day until the final workshop, 12, where he encountered four bugs.[24] Therefore, I selected workshop 12 to represent Jamal's most buggy independent project work day. I also chose workshops 4 and 9 to represent the other student groups during their most buggy independent project work days. This strategy would have resulted in five selected days, the structured debugging task day plus the most buggy day for each of the four student groups, but the Monkey group and the Hippo/Unicorn group coincidentally had their largest bug day on the same workshop, workshop 4. Because of the selections I made, I had a chance to investigate what all the groups were doing during the chosen days as well. Recall that I transcribed the activities, manipulations, and utterances of all student groups for all of the selected days, thus giving me a more well-rounded picture of all groups' working patterns.

In the transcripts from each of the sampled days (16 transcripts in total), each instance of three or more turns of interaction, taken as a combination of verbal and non-verbal, was highlighted as a potential collaborative episode. For each highlighted episode, I reviewed the corresponding video excerpt two to three times to determine what the students were engaged in doing, what the verbal utterances were referring to, and where the students were focusing attention to determine whether the episode fit under the

---

[24] Although Jamal worked alone and was not considered part of the intragroup collaboration analysis, he did encounter bugs and so his most productive bug day was also included in the overall selection of workshops to analyze. The same four selected workshops were used for analysis of collaboration and debugging.

working definition of collaboration. For instance, if one of the students in the exchange was talking about a piece of programming code and the other student was speaking in turn about the feathers he or she was putting on the physical pet, an instance of collaboration was not coded even though it may have appeared on paper as though the students were interacting with one another and not just in tandem. In this case, it would be more useful to call the exchange one of cooperation, defined as individual pursuits combined to make a collection of results, rather than engaging in a shared task together through negotiation and joint knowledge building, known as collaboration (Stahl, Koschmann & Suthers, 2006). Therefore, students had to be attending to the same idea for collaboration to be coded. Every change in control over the technology/design was also viewed again to ensure the exchange took place. Finally, each episode deemed collaborative was placed on a timeline to illustrate a collaborative summary of the project.

Table 6

Student Groups, Projects and Number of Bugs Encountered on the Four Days Selected for Analysis. The Student Group for Whom the Day Represented the Biggest Bug Day is Highlighted

Workshop Day	Student Group	Project	Number of Bugs Encountered
1	Rocky, Tegan & Ted	Monkey	7
	Carlos, Dino & Maya	Alien	9
	Tabitha	Hippo/Unicorn	6
	Jamal	Zebra	6
4	Rocky, Tegan & Ted[25]	Monkey	4
	Carlos, Dino & Maya	Alien	10
	Steph & Tabitha	Hippo/Unicorn	3
	Jamal	Zebra	2
9	Rocky, Tegan & Ted	Monkey	1
	Carlos, Dino & Maya	Alien	1
	Steph & Tabitha	Hippo/Unicorn	4
	Jamal	Zebra	1
12	Rocky, Tegan & Ted	Monkey	0
	Carlos, Dino & Maya	Alien	4
	Steph & Tabitha	Hippo/Unicorn	3
	Jamal	Zebra	4

[25] Tegan, Rocky, & Ted encountered four bugs on workshop day 4, which was the most bugs they encountered during an independent workshop day. Thus workshop 4 was selected for analysis.

Collaboration Results

The resulting analysis showed that student interaction differed between the more structured workshop days and the independent project days. On average, the collaborative episodes of groups whose members were present[26] accounted for 67% of the overall time of workshop 1, a structured debugging task workshop. For example, Tegan and Rocky, for Ted was absent workshop 1, engaged in 18 episodes of collaborative exchange for a total of 169 exchanges or turns considered to be collaborative (see Table 7). These data suggest groups were highly interactive and also spent a great deal of time on task. In contrast, groups spent an average of 16% of time collaborating during the other three workshop days combined. For example, Tegan, Rocky, and Ted's group engaged in 13 collaborative exchanges during the three other workshop days, workshops 4, 9, 12, for a total of 43 collaborative exchanges in workshop 9 and 45 collaborative exchanges in workshop 12. Tegan worked alone and therefore could not collaborate with her group members in workshop 4. These data suggest that Tegan, Rocky and Ted participated in about 25% of the collaborative exchanges during an independent day versus during the structured debugging task day. These results are representative of how all the student groups collaborated.

Carlos, Dino, and Maya collaborated for 24 minutes the initial day and then for a combined 30 minutes the following three workshop days taken together (see Table 8).

---

[26] Some groups had only one member present on workshop 1, like Tabitha was the only member of the Tabitha & Steph group. She did not collaborate on day 1 because she worked alone. Therefore Tabitha was not included in this data figure. Also, Jamal worked alone and could not collaborate with himself.

Table 7

Number of Collaborative Episodes, Total Collaborative Exchanges, Length of Workshop, and Percentage of Collaborative Exchanges Versus the Structured Day for Tegan, Rocky, and Ted's Group

Workshop	Group members present	Number of collaborative episodes	Total number of collaborative exchanges	Number of minutes of workshop	Percentage of collaborative exchanges versus structured debugging day
1	Tegan, Rocky	14	169	~ 34	100%
4	Tegan	0	0	~ 40	0%
9	Tegan, Rocky, Ted	6	43	~ 50	25%
12	Tegan, Rocky, Ted	7	45	~ 49	27%

Tabitha worked without Steph on the initial day because Steph was absent. In the following three workshop days, Tabitha and Steph collaborated for a total of 17 minutes. Finally, Jamal worked alone and therefore is not counted in the within group collaboration analysis.

All inter-group collaborative episodes coded involved a member of the Tegan, Rocky, and Ted group (see Figure 33). Carlos, Dino, and Maya, any or all of the members therein, collaborated with Tegan, Rocky, and Ted five times during the project. The two groups exchanged ideas for eight total minutes. A member of Steph and Tabitha collaborated with Tegan, Rocky, and Ted seven times for a total of seven minutes. Jamal

did not collaborate on any occasion with another student in the class. In total, 22 minutes of workshop time (in roughly 10 group hours of workshop time where approximately 7 of the total hours were spent working independently) were spent sharing ideas across groups. This accounted for approximately 5% of workshop time.

Table 8

Minutes of Collaboration Time for Each Group During Workshop 1 and then During the Combined Independent Workshops Numbered 4, 9, and 12

Workshop/s	Steph & Tabitha	Tegan, Rocky & Ted	Carlos, Dino & Maya
1	n/a	24	24
4, 9, 12	17	14	30

One of the design commitments for the project was to encourage productive exchange of ideas through the cultivation of a learning community. The initial collaboration analysis suggests students were able to effectively collaborate when working through the structured debugging task. However, after the structured debugging task took place and despite efforts to encourage student sharing and building of ideas throughout the project sequence, through facilitator guidance, opener and round table discussions and by having students fill out a design journal each workshop that asked

specifically about whether students worked with one another that day (see Appendix D.), students then did much less interacting about their projects when working on their independent projects.

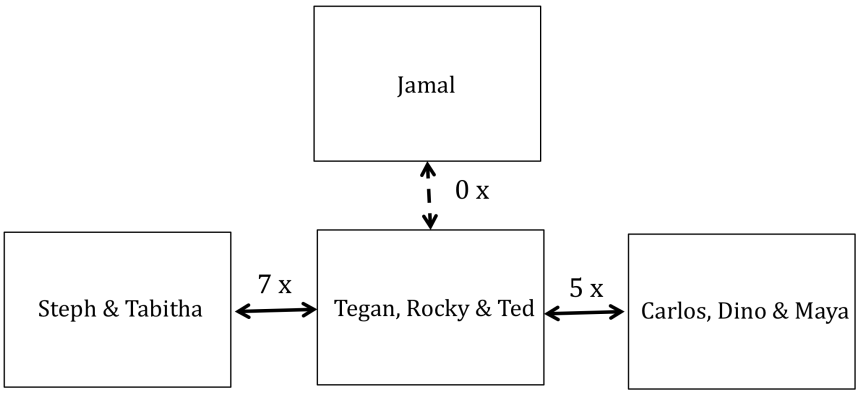

Figure 33. Collaboration between groups during the sampled workshops. All groups collaborated with Tegan, Rocky, and Ted's group.

Many reasons may account for why collaborative occurred as observed. One possible reason for less interaction during independent designs could be that students did not work well together, even though the students chose their own partners or decided to work alone. Indeed there was evidence of this during workshop 9, when Tabitha and Steph, best friends, refused to speak to one another for the duration of the session because of an out of school conflict. It is also possible that debugging may have lent itself to solitary pursuit. Working through difficult problems may naturally have promoted

individuals to work alone. But that does not explain why when asked specifically to debug, on the structured debugging task, students were more highly collaborative. Also, other studies have shown debugging to be a distributed activity (Berland & Lee, 2010). These hypotheses do not adequately describe the way students interacted during the project. Students were able to collaborate on structured debugging tasks then chose to interact more infrequently on their shared projects. In the subsequent sections, I describe potential reasons for the interaction observations and provide collaborative episodes from the transcripts that help illustrate the collaboration observed in the project. But first, I highlight an aspect of the population that may have confounded the overall picture of collaboration.

Accounting for Absence

First, before outlining hypotheses for the collaborative structures observed, I should address the issue of absence. Absence within the alternative school was prevalent. Students had chronic truancy problems, were routinely sent to detention centers, jail or into foster care locations and often disappeared for days or months at a time. During workshop 1, Steph, Tabitha's partner was absent, meaning that Tabitha's group could not be included in that days' collaboration analysis. She had no group member to collaborate with. The fact that only two of the three groups could be analyzed for workshop 1 reduces the robustness of the collaboration analysis, but the results from the other two groups are so striking they cannot be ignored. Ted was also absent from workshop 1, but his group Rocky and Tegan were able to collaborate without him. Not to mention the fact

that he was again absent one of the three subsequent workshop days chosen for in depth analysis. Tegan worked alone one of the workshop days because both Rocky and Ted were absent. For her group, the total collaborative minutes in the independent workshop days, 14, was taken from only 100 total minutes of workshop time instead of 140 minutes for the other groups. By counting only days where at least two group members were present, I attempted to alleviate the absentee problem since it was difficult to collaborate with group members that were not present. Therefore, absence was not a mitigating factor in the collaboration analysis.

Why Else Student Collaboration Was Limited

In many respects there are things beyond absence to discuss. Other dynamics were at work to explain students' collaborative structures and were visible empirically. Students in the project were able to collaborate by interacting with one another because they did so during the structured debugging task. However, students did little interacting about their project design or development in subsequent workshops. During the independent project workshop sessions, as I will describe below, student collaboration was highly modularized, more like definitions of cooperation rather than collaboration (Stahl et al., 2006). Rather than jointly building upon ideas and design plans with one another, the emphasis was on students taking distinct roles and responsibilities therefore working on independent goals, within the larger goal of making an interactive pet. It was interesting to observe this modularization because the pet projects were integrated and combining individual portions at the end sometimes led to groups having an end result

that made little sense, for instance in Steph and Tabitha's case, a unicorn physical pet and a hippo virtual pet (see Figure 34) This form of modular collaboration took place, I suspect and will discuss below, because of tendencies among students to treat their work as proprietary and to distrust other students. Also a partitioning of work occurred across the board reifying existing interests and expertise and resulting in a tendency to become emotionally attached to specific parts of the design process.

Figure 34. Steph and Tabitha's final project. The unicorn on the left was Steph's physical pet design whereas the hippo on the right was Tabitha's virtual pet design. The head feathers are one visual aspect the two creatures, that are intended to be representations of the same pet, share.

Expertise And Interest Lead to Divided Roles and Goals

Despite collaborative intentions in both planned activities and in facilitator interactions with students, full review showed that every multi-student group naturally distributed responsibilities on their independent projects according to perceived strengths or preexisting interests, not opening new opportunities for students (see Table 9 for details).

Table 9

Student Design Teams and Roles

Pet	Student	Sex	Role
Monkey	Rocky	M	Programmer
	Ted	M	Comic Relief
	Tegan	F	Tangible Pet Designer
Unicorn	Steph	F	Tangible Pet Designer
	Tabitha	F	Programmer
Alien	Carlos	M	Programmer
	Dino	M	Tangible Interaction Designer
	Maya	F	Tangible Pet Designer
Zebra	Jamal	M	Programmer & Pet Designer

Other work highlights this same problem, in that naturally distributed roles in collaborative Constructionist learning have been shown to potentially exacerbate students' social and academic identities (Abrahamson & Wilensky, 2005). In the DigiblePets Project, each team had one programmer, one physical pet designer and if applicable, a third member who contributed predominantly when asked to assist with a specific task or took on the role of tangible interaction designer.[27] Once the craft materials arrived, teams deviated from this structure only in circumstances when a student was absent.

Perceived expertise influenced students' participation within their groups. For instance, when asked why the group broke up responsibilities in an interview, Tegan replied, "Well, cause they're (Rocky and Ted) not really crafty and also they'd just mess it up because I had an idea in my head." The perception that the boys were not crafty, shared by the boys, resulted in the boys' lack of opportunity to design with the crafts. The assumption that computational crafts provide a pathway to computation and also deliver students to engage in new disciplines does not always hold. In this case, the opposite was true, as described in Chapter 4, Tegan abandoned programming in pursuit of her interest in physical pet design; Rocky and Ted were not encouraged to craft and were rebuked by Tegan when either one attempted to contribute. Similarly, Tegan and others in the role of crafter rarely accessed the computer and did little to no programming. Carlos echoed the same idea in his interview regarding role assignment based on expertise, stating that his

---

[27] A tangible interaction designer's role was to integrate the sensors, buttons, clips, and slider within the physical pet to allow for the user to interact with them and cause virtual reactions on screen.

group divided responsibilities because, "They're better at making stuff with their hands and I'm better at the computer". Strict role distribution contrasts flexible role shifting styles of groups of youth making music videos (Peppler & Kafai, 2001). In this case, students assumed roles within the overall context of the project and remained within them.

However, distribution of work and collaboration are not necessarily mutually exclusive. Just because each individual has a specific role within a larger task, does not mean the individuals within a group will not interact. Highly collaborative yet distributed working styles have been observed in classroom implementations of other computer-based technologies including seamless thinking of a pair using the Constructionist software tool, Boxer (diSessa, 2000). Different from a computer-based technology like Boxer, the DigiblePets technologies can support multiple designers at the same time, which might suggest that they would be even more collaborative. Also, by having both craft, popular toy, and computational components, DigiblePets technologies aimed to make use of multiple areas of potential expertise, meaning giving youth who do not relate to computing an opportunity to play a vital role in other aspects of the project development process. As a result, I expected to observe more interaction within groups. However, not only did students in the DigiblePets project distribute roles, they also isolated themselves by attending to individual goals. This is a critical difference. Students using the Boxer programming environment had the same goals, much like the students in the project during the structured debugging task, whereas during the open-ended workshops students divided both roles and goals, separating themselves and their ideas.

For instance, during workshop 8, Maya (the crafter) asked her team member Carlos (the programmer) for advice on the design of their physical alien.

Maya	Wouldn't it be cool if he had so many eyes?
Carlos	I don't know. I don't care.

Instead of participating in Maya's design process, Carlos simply responded, "I don't know. I don't care." This is striking because the project as a whole reflected all of the students' efforts. Although Carlos did not see it, a functionally superior pet with a visually poor physical pet would make the whole group's project appear to me to be less integrated and less successful overall.

The Difficulty of the Task Prevented Students from Taking Up Collaborative Opportunities

Additionally, computer programming is conceptually difficult (Guzdial, 2003). The difficulty students had implementing their own personally meaningful ideas especially because the students were isolated by roles and goals often caused collaborative opportunities to not be taken up. Even in instances when one partner was seeking collaborative interaction, many times the other student/s in the group were so preoccupied by the demands of their own work they were not able to take up the collaborative interaction. For instance, Tabitha and Steph worked side by side on the programming code (Tabitha) and pet construction (Steph). In the following two-minute episode, during workshop 9, Tabitha was working on getting the unicorn to dance by

repeating a switching costumes (see Figure 35) command that would make the character look as though it was rearing up and down.

Figure 35. Tabitha's two-costume approach to dancing.

Tabitha had worked for four minutes on implementing the two-costume approach and was getting increasingly frustrated with the lack of results. Tabitha was in the process of identifying a two-fold bug because the PicoBoard became un-paired[28] and needed a reboot, meaning no buttons were working, and although her dancing code would work, she added a glide command as an afterthought that caused the unicorn to glide out of the

[28] The PicoBoard needed to be paired with the computer in order for the computer to recognize the device. On occasion, for no discernable reason, the two would come unpaired, causing no data to be transferred to the computer. In this case, the PicoBoard would need to be essentially manually reintroduced to the computer.

stage area. This caused a situation where essentially the unicorn would leave the visible area of the screen and then proceed to dance where no one could see it. Steph was working on building the physical unicorn. Even though Tabitha was quite vocal about her difficulties and frustration, Steph never acknowledged that Tabitha seemed to be struggling. Instead, Steph asked Tabitha for advice on her physical construction. Tabitha, so deeply entrenched in her programming bug did not answer Steph's question, instead reiterating her own difficulty. At the end, both girls continued to work on their own pieces of the design without ever helping one another.

Tabitha	*(double clicks unicorn icon, stops and starts again)*
	Where the fuck did our unicorn go?
	(steph no response) *(moves monitor back, checks through code)* *(goes to costumes, goes back to scripts)* *(double clicks some code, clicks arrow, clicks stop, start, double clicks to start)*
Steph	Hey, can I glue this to this button or no?
Facilitator	Sure.
Tabitha	*(clicks on forever loop by itself in corner)*
Steph	You want these things on the bottom?
Tabitha	I don't care dude. I don't know where the fuck our horse went.
	(stops program)
Steph	*(working with button on pet)*
Tabitha	*(stopped working on bug)*

(changes steps, degrees to 10 and degrees to 50)

Steph, the physical pet designer, and Tabitha, the programmer, have different roles within the project and seemed wholly engrossed in their own component of the overall design. Tabitha was vocally frustrated with her programming bug, which seemed to prevent her from being able to switch gears and attend to Steph's question. Similarly, Steph continued to work on her physical design while Tabitha struggled with the disappearing unicorn. Distributed roles and goals combined with the difficulty of the project seemed to discourage productive interaction, even in instances when one or more the students expressed an interest in sharing ideas. This episode reflects an interesting set of problems. That students became so occupied in implementing their design ideas within their individual domains was exciting, however the design ideas could become cumbersome because of the difficulty of the programming and the potential for multiple embedded bugs, leading to frustration, which, as the episode suggested, did not lead to sharing or building ideas. This excerpt of parallel but separate work was representative of observed student interactions within groups during the independent design.

Modularity: A Population Characteristic

Several observed factors contributed to an isolated working style within and between groups. In the previous section, I described how divergent roles and goals combined with the difficulty level of the task contributed to a more isolated working style. In this section I will talk about how characteristics of the population further

exacerbated students' isolated working structures and contributed to why students chose to both segregate roles and hold on to their ideas.

A proprietary, wary population. The alternative high school was chosen for study specifically because students struggled academically and were not accustomed to learning in open-ended environments or with new technologies. I hypothesized students would work together, especially when encouraged, in a creative, community environment that was not graded. Contrary to this assumption, students were highly proprietary about their ideas and distrusting of others' capabilities.

One factor affecting collaboration concerned students' beliefs that design ideas were proprietary. In several instances, students showed off aspects of their designs to other students. For instance, Tegan worked alone during workshop 4 because her group was absent. At the end of the workshop, Tegan showed off her pet design to Anna, the classroom teacher, who was developing her own pet. I alluded to this episode when talking about Tegan's engagement in the project, during Chapter 4. Here I will describe it in full detail. In this episode, Anna visited Tegan's computer and looked at the new functionality. Recall from Chapter 4, Tegan had been working on painting a ladder character and putting it in the scene so that her monkey, the pet, could use the ladder to climb onto the bed. In the excerpt, Tegan was proud to show Anna her work but wanted to make sure that Anna did not use her idea.

Anna	Oh and you made a ladder as a character. So you can make it do different things, like.
Tegan	Yeah.

Anna	Like you can move it and stuff.
Tegan	Yeah, but I'm going to figure that out next time.
Anna	Oh, that's cool! And if he touches that maybe he jumps on the bed or something.
Tegan	Yeah, I guess so.
Anna	Cool. That's a good idea!
Tegan	Thanks. Don't take it!
Anna	I won't. I'm totally on a different track. Don't worry.
Tegan	K.

In this episode Tegan did not really reveal what she planned to do with the ladder, Anna filled in some possibilities. Despite the lack of in depth ideas shared, Tegan quickly claimed ownership, "Don't take it." It seemed she was simultaneously grateful for the positive feedback, "Thanks" and afraid that the praise may mean she would lose some of her autonomy, "Don't take it!"

Proprietary feelings over design ideas prevailed throughout the independent workshops. Tegan's sentiment was representative of all students observed during the project. In another example, during workshop 3, when another designer expressed interest in having an outer space theme, Carlos shouted, "Get off my moon!" Carlos believed he came up with the idea to use the moon background, which he did not create just simply imported from the stock options, and wanted to prevent anyone else from using it. Again, in workshop 9, Anna sought advice from Maya on her tangible pet design. In the following episode, Carlos refused to allow Maya to share ideas with Anna.

Anna	Hey, give me an idea. How do I make this dinosaur?
Carlos	Why are you asking her?
Anna	Because she's like really clever. She did all that *(gestures to tangible alien pet)*.
Facilitator	Yeah, look how creative she is.
Dino	Oh yeah.
Carlos	Why don't you make it yourself? That's what this is about.
Anna	I'm just looking for ideas man.

Carlos was reluctant to allow Anna to brainstorm with Maya. Anna attempted to foster the building and sharing of ideas, but was met with resistance. The potential for collaboration broke down due to Carlos' ideas of ownership and fairness, "Why don't you make it yourself, that's what this is about." The students' ideas of proprietary knowledge is in stark contrast to the collaborative processes of students engaged in software design (Kafai & Harel, 1991) and cooperative, code-sharing working style of computer clubhouse youth designing with a combination of computers and repurposed materials (Millner, 2009).

Along with a proprietary nature, another factor contributing to a distributed working style stemmed from students' wariness of others' capabilities. For instance, in her interview, Tabitha said she programmed by herself because her partner and best friend was, "Kind of a slacker". It is true that during workshop 11, when Tabitha was absent, Steph decided to try some programming for the first time. She incidentally deleted all the

girls' programming code by deleting their main sprite and was unable to retrieve it. It could be that Tegan referred to Steph as "a slacker" because she was both discouraged by the code deletion and had thought something like that might occur. Tabitha was not alone in having doubts about the capabilities of her partners. In another example, Tegan claimed in an interview that she developed the tangible pet by herself because her partners would "Just mess it up." In final example, during workshop 4, Carlos made it apparent what he thought of Dino's ability. During this example, Carlos asked the facilitator for help on a design idea. Dino was not in the room at the time. When Dino reappears, the facilitator addressed him instead of providing an answer, trying to encourage the two to build ideas together.

Facilitator	Are you going to help out? Because I think Carlos needs some help.
Dino	I know.
Carlos	Dino's not smart.
Dino	Really Carlos, you need help again?

The boys exchanged remarks that I am interpreting as put-down statements between friends, but the underlying message was one of discrediting one another's competence. Despite the idea that the boys were in the same group, working on the same project together, Dino said, "Really Carlos, you need help again?" This suggests that Dino felt the programming component of the work belonged to Carlos and that helping him with it would be somewhat of an imposition on Dino, who had up to this point not

been engaged in the development of the pet in any way. After this exchange, Carlos continued to work on the programming bug he had been attending to and Dino continued to make off-handed comments, never working to help Carlos. The pervading atmosphere of distrust permeated all aspects of students' projects from how students worked in their groups to how students treated other groups. These short episodes summed up how students felt in general about one another and how they questioned another's competence in contrast to their own. There were no instances observed where a student sanctioned the borrowing and reusing of an existing idea. Students believed that ideas should not be shared or appropriated by others. Students were predisposed to claim and delegate ownership over a specific segment of the work and ensure creative ideas were not communal. In some respects, this is not surprising. The culture of school these students have encountered has potentially involved them getting in trouble for ‘cheating’ off of other people.

An emotional technology. The observed distributed approach to design projects can be partly explained by characteristics of the participating population, but technology-specific factors also affected how students interacted. The physical portion of the design task promoted the cultivation of strong emotional connections and for some students, very positive sentiments about their resulting work. Indeed, it appeared the technology was perhaps too effective in promoting these ideals thereby further contributing to students' segregated structures.

Students exhibited some sense of pride with respect to what they made and were able to demonstrate with their pets. Showing off a design to others was also a way

students exhibited personal involvement in the project. For example, Steph ran out of the room during workshop 9 with her pet to show it off to a staff member in another part of the school saying, "I'm going to show Evelyn". I was surprised when reviewing the video to see this episode because I had assumed through my observations that Steph, especially with her sporadic attendance, limited productivity, and constant gossiping, had not felt a personal connection to the project. Her desire to seek out approval on her physical design from another adult in the building suggested she was more involved than I had thought. Steph's interest in showing off her pet to others was representative behavior of the students in the project.

To illustrate this further, consider that Tegan showed her monkey off five times during workshop 4 (40 minutes in length) including to the facilitator, two students and the classroom teacher. In a display of connection to the project also during workshop 4, representative of students in the project as a whole, Carlos showed off his group's pet design to the principal of the school, by getting the principal's attention, who had arrived in another part of the room for other reasons, and guiding her over to his computer.

Carlos	Look, look, look! What I made it do. I'm the most advanced right now.
Carrie	(coming over) Are you? Ok. Show me.
Carlos	I made it so it rides the carpet on the moon to the tramp and then it jumps to the stars.
- - -	
Administrator	Wow, I'm impressed.

Carlos	See? No one had a magic carpet to ride to a trampoline.
Carrie	I agree. How did you get a magic carpet?
Carlos	Cause I'm pro like that.
Carrie	Ooooh!

In this excerpt Carlos was impatient to get the principal's (Carrie's) attention, "Look, look, look." Then he shared with her, "I'm the most advanced right now", suggesting he wanted her to be proud of his accomplishments during the project. Notice, Carlos did not say "we," he said "I." When he described the pet to Carrie, "I made it so it rides the carpet on the moon to the tramp and then it jumps to the stars", he told her the functionality of the pet but nothing about what the pet looked like, a domain of the design project that was his partners' responsibility. As discussed in Chapter 4, Carlos was very functionality focused. Following watching the pet in action, Carrie said "Oooh!" and another administrator, who came by to look, said, "Wow, I'm impressed." Carlos received very positive feedback on his work.

Another way students showed their connection to their projects was in how they spoke of their pets while working. Most commonly, students referred to their pets as "he" or "him" instead of "it." By using a pronoun students personified their designs, viewing the pets as having life-like qualities. For example during the project, Steph, Tabitha and Tegan always systematically referred to their pets as "he." Using pronouns is one way researchers assess the effectiveness of computational agents in appearing real or life-like, in promoting affect and relational qualities (Catrambone, Stasko, & Xiao, 2002; Lee,

Kiesler, & Forlizzi, 2010). This was representative of how students saw their projects as more than just glue, feathers and pixels. This accompanied by other positive language related to their pets. For instance, Steph and Tabitha talked fondly of their pet. During the course of workshop 4, after trying out seven different potential pet characters and settling, for the moment, on a lion, the girls made comments suggesting their enthusiasm for their character. In the excerpt, Steph and Tabitha, with input from Byron, another student who dropped out of the project, work together to come up with their lion pet. The girls expressed excitement during this playful episode.

Steph	I like the lion. Let's do him.
Tabitha	Let's paint him.
Steph	I wanna paint him! (*chooses paint brush and orange. puts a dot on his mouth.*) No I need.
Tabitha	Make his eyes red.
Steph	Ok. (*selects paint bucket tool. clicks eyes they turn red)*
Tabitha	*laughs*
Byron	Rrrr!
Steph	*laughs* Let's make him all cool looking. Rrrrr! (*makes lion green outlined.)* Ahhh! What color was that before?
Tabitha	I don't know.
Steph	(*clicks cancel.)*

Let's make him tongue tie dye.
Let's leave him like that.
(*clicks enter.*)
Our lion's chilly chill!

The girls were able to customize their pet on screen to look how they intend. After several iterations of changing certain aspects of the lion, Steph referred to the lion as "chilly chill", a positive comment on how cool the lion with red eyes was. Making comments such as these about the pet as it developed to look and act more like what the students intended was representative of how all students in the project reacted to and connected with their pets' development.

Tegan showed the most attachment of all the students to her pet. In addition to showing of the pet five times, she referred to the monkey as "cute" 19 times during the 40 minutes of workshop 4. "Cute" was unambiguously the only word she used. However, Tegan worked alone on this particular day. The majority of her utterances about how "cute" the monkey was are simply to herself. It was as if she could not hold back how fond she was of her design, it came spilling out. Tegan's attachment to her pet monkey was more extreme than the other students in the project, however all student exhibited portions of attachment to the project through the way they talked about their pets, customized their pets and showed pets off to other people.

It makes sense that students would feel a sense of emotion and ownership over their project ideas and pets. Hybrid technologies are designed to be emotional and absorbing (Eisenberg, 2003). For example, students showed pride in their PicoCricket creations when demonstrating them at an exhibit (Rusk et al., 2008). Interactive pets themselves have these qualities as well. For example, interactive pets have begun to be

used for therapeutic reasons to encourage emotional response and attachment in severely disabled children (Marti, Pollini, Rullo, & Shibata, 2005). Also, in a study, children showed rapid emotional attachment to an interactive dog, calling it a playmate and empathizing with it, after interacting with the pet for an average of only 20 minutes (Weiss, Wurhofer, & Tscheligi, 2009). However, the personal connections students made, instead of leading students to share and build ideas with one another, instead facilitated students' isolated working styles. Students held their ideas close because they seemed to genuinely care about their projects and felt they owned those ideas. This, however positive, subverted the idea that students should share and build ideas together.

Conclusions About Student Interactions

As hoped, the tangible/digital pet design project provided multiple starting points for students with different interests, some interested in crafting or building and others in programming. However, rather than being interdependent, the craft and computational media were dichotomized by students, allowing prior interests and expertise to dictate participation. Characteristics of the student population may have further encouraged the distributed working structure. Students divided tasks and attended to different portions of the design, which has potential implications for the development of new interests and learning. Students took control over the separate parts of the hybrid technology because it made sense for everyone to be working at the same time. The observed distributed working structures were naturally devised by students as a way to divide tasks in an effort to efficiently complete the project task, which all student groups did. Students did not

share and build upon ideas as much as intended, and that could be a feature of the population or the hybrid technology. However, the initial collaboration was there during the structured debugging task, suggesting that collaboration was possible, perhaps in a more deliberately constrained environment.

CHAPTER 6

DEBUGGING

One emphasis of this study was debugging, how students approached and dealt with bugs, often described as unexpected results from executing programming code (Pea, 1986). As described in the previous chapter, one hypothesis was that students would spend a great deal of time engaged in elements of debugging. For example, a recent study of third year college computer science students showed that students spent on average from 38% to 47% of their programming time debugging (Chmiel & Loui, 2004). I designed the project to revolve around debugging and remixing code as strategies for learning aspects of programming and design thinking. Bugs were fundamental to progress and the learning environment centered on finding and fixing errors in a playful way, free from academic stigma or personal consequence. The field continues to highlight the need for research on debugging as part of the larger landscape of understanding student computational practices, especially in learning environments with Constructionist-inspired technologies (Grover & Pea, 2013).

As mentioned in earlier chapters, debugging is important because it requires that students can both read and understand aspects of a computer program and can invent and implement a strategy for finding and fixing the bug (Winslow, 1996). Research suggests that debugging is intellectually challenging for novices (McCauley et al., 2008; Murphy et al., 2008). The debugging process often incites novices, even in Constructionist programming environments, to work around errors or give up in frustration (Murphy et

al., 2008; Pea, 1983). In addition to seeing how students engaged with each other (Chapters 4 and 5), I was interested in looking at what kinds of bugs students encountered with the hybrid media and how students reacted to and, in most cases, used strategies to resolve bugs.

Students encountered bugs in three ways during the project. First, students were introduced to programming through finding and fixing a series of preprogrammed bugs in my prototype pet Scratch project during the first two days of the workshop (see Chapter 4 for more detailed discussion of student activity during this task). Second, students faced bugs that appeared by virtue of completing their independent project work while trying to reuse and modify existing and sometimes create new programming code. And third, during interviews after the project, students performed a debugging task assessment on the computer while I observed and recorded them. The assessment was similar to the original structured debugging tasks. Students used a different prototype Scratch program I created that they had not seen previously and worked through a set of bugs I developed based on functionality I observed students implement in their own projects.

Though the design of the activity allowed students to wrestle with debugging as part of the process of developing independent hybrid design projects, it was necessary as a first step for me to examine the nature and variety of bugs students encountered. Hybrid design media are so new that we are still understanding the types of bugs students produce using Scratch and the ways in which physical media can have bugs [e.g. for an introductory exploration of the use of debugging in Scratch see Griffen et al., 2012)]. This chapter begins with a characterization of the types of hybrid design technology bugs

observed during the project. Next I examine the general bug landscape during the project by student group. Then I describe the ways in which students handled various bugs and illustrate contrasting student debugging methods that were recorded during the project. Finally, I discuss results from the debugging task assessment that occurred after the project was complete.

In this chapter, I will highlight instances of where students encountered unexpected results while engaged in design and development. Bugs are typically thought of as programming problems. However, I observed that bugs can occur in both the virtual environment, for instance when a new piece of programming code does not perform as planned, and also in the physical environment, for instance when a method of user interaction with a button subverts the user's ability to move a slider. This idea, that programmatic thinking is not necessarily limited to computational domains has been raised previously (Eisenberg, 2003; Berland & Lee, 2011). However, a systematic examination of the types of virtual and real-world bugs that can occur and how students address different types of bugs in different domains is a unique contribution of this work.

Analysis Methods

To analyze bugs and debugging, transcripts of daily student activity, video, and field notes were used. I took multiple passes through the data to generate codes. First I identified discrete episodes of bug occurrences for each student group over the course of the entire project. I also captured duration of each debugging instance, marking the beginning and end of each episode. This reduced the data to distinct episodes comprised

of activity, what students did, and interaction, what students said or conveyed around a single problem-solving event. Even situations where students found a discrepancy and chose to ignore it are important in this analysis and were considered to be potential debugging instances. This initial collection of bugs included 102 total bugs encountered by students over the 12 workshops. To reduce the data, I created a bug timeline for each potential debugging instance for each student group for just the sampled workshop days, which recall were chosen for specific reasons (see Chapter 4 for more details). This meant that 64 bug instances were analyzed in detail.

In the next analysis cycle, I reviewed original video and transcript data for each elected episode. I needed to capture both the kinds of bugs students were encountering and also what students were doing in the face of bugs. Following a descriptive coding progression (Miles & Huberman, 1994), I extrapolated from each episode the attributes of the bug and reduced the bugs to a short phrase summary.[29] I then synthesized the coded data by comparing and contrasting codes for types of bugs, looking for themes, overlap and inadequacies. I combined codes that had redundancies and expanded codes in instances where one code represented discrepant kinds of bugs. For instance, tangible bugs had different qualities depending on whether they were hardware or user-interaction related. This process uncovered two central types of bugs, virtual, having to do with the computer program, and tangible, having to do with the physical pet or user interactions with the physical pet and/or PicoBoard.

---

[29] In descriptive coding, usually a one-word summary is used. In this case a few words helped me better synthesize the data.

For each bug instance, I then described characteristics of the activities the students engaged in when faced with the bug. I took another pass at the same bug instance timeline data looking at a separate dimension - that of student processes rather than the characteristics of the bug. For each bug instance, I detailed what students did to deal with the bug. Student activities ranged from ignoring the bug to asking for help. The resulting codes were then examined. Instances where codes could be collapsed, refining the essence of what students were doing in the code names and adding codes when I determined students were doing more than one activity during a single coding instance. For example, in some cases a student would try to implement some changes to their programming code to fix a bug, but then asked for help when that method did not solve the bug quickly. When collapsing codes, I reviewed the video excerpts that represented the disparate codes to ensure that collapsing the codes would capture the nature and essence of each debugging instance. For instance, for the *tinkering* code in the face of a virtual bug, students had to exhibit the following: a cycle of at least two instance of scrolling through existing programming code, changing an element or more of the programming code, and testing the change without asking the facilitator or another student group for support. If at minimum all of these activities existed, then *tinkering* was used.

Process codes were not mutually exclusive but were sequential in that students often used more than one method to identify and fix a bug. Although multiple codes could be employed in a single debugging instance, normally students employed methods in a linear fashion. For instance, a student might initially have implemented a set of

solution ideas to fix a bug, but then might decide to delete the buggy code instead of continuing to figure out how to resolve the bug. By having more than one activity in each debugging episode, I hoped to better capture the nuances of students' activities and find parallels in students' methods.

Operationalizing Bugs in Computational Crafts

As described earlier, a bug is most often thought of as a situation where executing a computer program reveals a discrepancy between what the programmer intended and the program's output (Pea, 1986). Encountering a bug is a type of what Schank, Fano, Bell, and Jona (1993) called *expectation failure*, when an individual expects something to hold true based on prior patterns but instead it does not, creating a memorable and important opportunity for learning. In the case of hybrid design technologies, as is probably the case in other design and engineering endeavors that require design thinking around physical artifacts, bugs can occur with computer code and also with physical artifacts. This is different from many other uni-modal design projects, like projects that just use Scratch or other software programs. Researchers have begun to explore the idea of computation without a computer (Berland & Lee, 2011; Eisenberg, 2003); debugging with physical artifacts is an example of this type of thinking.

Instances of bugs are somewhat nebulous to define in retrospect, especially in others' work. The key was to be able to identify situations when students encountered an inconsistency between intent and result; this often produced a puzzling moment. Therefore, when a student was in the process of trying to develop new functionality, the

student may be frustrated, seek help, have questions and so on, however, none of these qualities are exclusive of instances where there is a bug although they may be symptomatic of many bug situations. To capture instances of bugs, I looked for two main elements. One, a student verbalized an expectation problem after running his or her program (called testing) or testing an implementation of an idea with the physical artifact. The student might say something like, "Wait a second, why did that happen?" for example. Or two, the student made known what he or she was trying to do but when testing, the program or physical artifact, a different outcome occurred. For instance, a student might say aloud, "I want to turn him green" then she might engage in changing the color of the animal using the painting tool and then run the program, signaling that she wanted to test her implementation of "green". If the animal remained white during the execution, this would indicate a bug instance. Running the program, in this case, showed the student's intent to see if her idea worked, suggesting that she believed the changes she made should affect the program the way she anticipated.

If I could not determine with certainty that an instance was indeed a bug or something different, it was not included because I wanted to ensure I was capturing students in the process of dealing with *expectation failure*. By the same reasoning, instances where students noticed something was wrong but decided to ignore the problem, for example by saying, "Oh forget it. If it wants to stay white then that's fine" and moving on to another idea, were included as potential debugging instances. Therefore a bug instance was any situation where a bug was detected regardless of whether the response was to identify and fix the bug or ignore the bug. It was important to note not

only all types of bug instances, for classification purposes, but also how students dealt with bug instances, meaning that choosing not to deal with a bug was telling of students' methods. In some cases it might have been too much effort to reconcile a design idea with a problem in code, as some researchers have observed in other Constructionist media-design projects (Kafai, 1996).

Categories of Bugs

As part of the overall understanding of bugs and debugging during this project, I wanted to capture, classify and categorize the types of bugs students encountered. [30] As discussed earlier, virtual bugs are most commonly thought of as computer programming bugs and tangible bugs are unique to design and engineering endeavors where a physical artifact is created. I developed the taxonomy of bugs based on literature about conceptual and syntactic programming bugs encountered by novice programmers. According to the literature, bugs, in this case virtual bugs, occur in two varieties, syntactic and conceptual (Kelleher & Pausch, 2005). Syntactic bugs are defined as how to convey instructions to the computer and conceptual bugs are how to arrange the instructions to the computer. Recall the designers of Scratch intended to reduce the occurrence of syntax bugs through the puzzle piece metaphor of the software program. Syntax bugs are very common for novice programmers (Guzdial, 2003; Kelleher & Pausch, 2005). I further attempted to reduce syntax errors by providing students with working prototype code that they could

---

[30] I did no bug analysis with the structured debugging workshop because the bugs during those two days were provided by me and did not occur naturally during students' development work. I discuss some aspects of these structured workshops in Chapter 4.

reuse and modify as they wished, on which to build their programs. Syntax bugs did occur during the project, as described subsequently, but with less frequency than conceptual bugs.

Conceptual bugs are described as errors that transcend programming language, pertaining instead to ways of thinking about programming (Pea, 1986). The study of conceptual bugs, rather than simply syntax bugs, has been deemed especially important and essential to understanding how students develop computational thinking skills (Grover & Pea, 2013). However, conceptual bugs are not often the focus of study with novice programmers (Grover & Pea, 2013). Cunniff, Taylor, and Black (1986) created a framework for categorizing conceptual bugs that novice programmers' encounter. The framework consisted of categorizing errors based on programming code that has elements either missing, spurious, misplaced or malformed. Evidence of missing, misplaced, and malformed programming code bugs was observed during this project. These bugs were especially well suited to describe some of the types of errors students made when creating new code. Recall, in this research project, students made Scratch projects from a combination of code reuse and existing code modification combined with the creation of new programming code. When students reused and modified code, they sometimes encountered conceptual bugs not adequately described by Cunniff et al. (1986).

Pea (1986) also developed a set of categories of conceptual bugs novice programmers encounter, based on the hypothesis that all conceptual bugs stem from the same underlying assumption young novice programmers have that the computer has a kind of hidden mind, much like a person. The idea that conceptual bugs are due to an

overarching misconception about computers having minds (Pea, 1986) helps to provide insight into some of the remaining bugs observed during the project. According to Pea (1986) conceptual bugs include erroneous assumptions having to do with: parallelism, intentionality, and egocentrism. Parallelism is an order of operations problem that occurs when a programmer assumes a computer will be able to interpret what comes earlier in a program based on information given later. Intentionality occurs when a programmer assumes the computer has goals that it will enact even if they may conflict with the instructions given. For instance a programmer may assume that the computer wants to continue within a loop even when the condition is not met. Finally, egocentrism occurs when a programmer attributes more meaning to a collection of code than what the code explicitly states. An example would be a student who believes the computer knows what the programmer was trying to create and will simply fill in the missing details as a human might. Specific instances of both intentionality and egocentrism were observed in code reuse bugs during the project. Parallelism was not observed, probably because the students were not creating typical novice programming code with conditionals, variables, and loops used to create precise objects on screen like geometric shapes.

To create the categorization of bugs observed during the project, both virtual and tangible bugs encountered by all student groups had to be considered. I used the categorizations from previous literature of novice programming bugs as a frame to help me understand and classify the types of errors I observed during the project. Pairing the codes developed by previous research with the specific consolidated instances of bugs observed with the hybrid (Scratch and PicoBoard) design technology was considered the

best way to account for the types of problems students encountered. The types of bugs and their frequencies during the project are listed in Table 10 and Table 11. The total bugs and frequencies are shown in Figure 36.

Table 10

Tangible Bugs Encountered During Representative Workshop Days

Tangible Bugs	
Hardware	6
User interaction	1
Craftware	2
Total	**9**

Students encountered three times as many virtual, or traditional, bugs (27) than tangible bugs (9) during the sampled workshop days (the sampled days are described in Chapter 5). This may be partially related to how difficult it was to determine when a physical artifact was providing unexpected feedback. Determining a virtual bug was oftentimes much more straightforward because students systematically tested their new code by restarting, or executing, the program. Each time a student ran a program indicated a potential bug instance whereas other indicators were needed to mark a tangible bug instance. To find tangible bug instances I noted "testing" implementations in

the same way students tested their computer programs, they also tested their physical pets in more subtle ways. For instance a student might have put the pet on the table and sat back to observe it carefully. Or, a student might have run their computer program and tested a piece of tangible functionality they had just created or refined, like pressing the button. If something unexpected occurred, an eyeball that a student had just finished figuring out how to adhere fell off and the student said something like "Why did that happen?" to reveal his expectation being different from the observed outcome, or if the button could not be pressed as expected, an instance of tangible bug was recorded. These were considered tangible bugs because in both instances of expectation failure, students used similar processes to address the errors as they did when encountering virtual bugs.

The variety of virtual bugs (7) also exceeded the types of tangible bugs (3). PicoBoards are relatively simplistic compared to their Arduino counterparts and do not require soldering or interaction with output sensors which may have made tangible bugs more prevalent. Furthermore, because more tangible bugs existed, the tangible bugs could be broken down from a broad category like "conceptual bugs" to more nuanced but universally encountered novice difficulties like "Misplaced/malformed initialization" and "Deletion of code presumed to be spurious." Thus creating more categories.

Table 11

Virtual bugs encountered during representative workshop days

Virtual Bugs	Type of Programming code	Basis in Literature	
Generic computer knowledge	N/a	Generic computational knowledge	2
Incompatibility of reused code	Reuse	Intentionality (Pea, 1986)	5
Deletion of code presumed to be spurious	Reuse	Egocentrism (Pea, 1986), Missing (Cunniff, Taylor & Black, 1986)	1
Misplaced code	New	Misplaced (Cunniff, Taylor & Black, 1986)	5
Insufficient knowledge of new programming code (Syntactic)	New	Syntactic (Kelleher & Pausch, 2005)	6
Initialization missing or not updated	New/reuse	Missing or malformed initialization (Cunniff, Taylor & Black, 1986)	6
Uncategorizable	New/reuse		2
Total			**27**

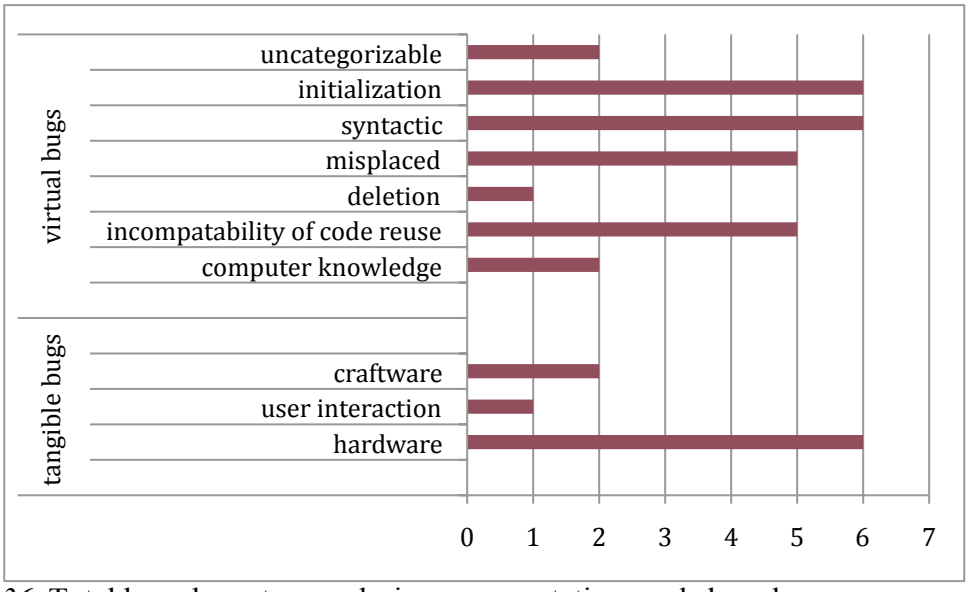

Figure 36. Total bugs by category during representative workshop days.

The final set of codes (see Table 12) included three types of tangible bugs: hardware, user interaction and craftware, and seven types of virtual bugs: computer knowledge, incompatibility of code reuse (an instance of intentionality), deletion of code presumed to be spurious (an instance of egocentrism), misplaced code (misplaced), insufficient knowledge of new code (syntax), initialization missing or malformed (malformed initialization), and uncategorizable. Hardware bugs involved some aspect of the physical technology and often were due to oversights or errors in plugging in or pairing devices. User interaction bugs described problems in design implementations that prevented users from interacting with the physical technologies through the pet as the designer intended. Craftware bugs encompassed all unexpected outcomes from working with craft materials to make the pet's physical body and often happen during adhering and/or molding parts of the pet. Generic computer knowledge bugs referred to instances where limited experience with computers caused a problem. Incompatibility of reused code bugs were a form of intentionality (Pea, 1986) bug that occurred when a student expected existing prototype programming code would accommodate a new design idea because the computer had a certain human like ability to understand what it should do. Deletion of code presumed to be spurious bugs were instances of egocentrism (Pea, 1986) and missing (Cunniff et al., 1986) bugs where a student deleted a piece of pertinent code, usually because he or she thought the programming code left in place was sufficient to achieve the desired result. A misplaced code bug, borrowed from Cunniff et al. (1986) occurred when students created the correct code to achieve their objective but have put the code in the wrong physical (on screen) location. These bugs are pertinent to Scratch

where each sprite & background had its own scripting screen. Insufficient knowledge of new programming code bugs were syntax bugs or problems with how to express instructions (Kelleher & Pausch, 2005).

Table 12

Summary of the Code Categories, Examples from the Data Corpus and Fixes Implemented

Category	Type	Description	Example	Fix
Tangible Bugs				
Hardware	PicoBoard	In a hardware bug some physical part of the technology is not functioning as expected primarily due to the user being unaware of some part of the technology, cords etc. but also sometimes due to unforeseen circumstances, like device pairing, that cannot be readily explained.	Rocky noticed the pet was eating haphazardly without him interacting with the PicoBoard. Eating normally occurred when the pet's alligator clips were manually touched together.	The USB cord was plugged into wrong computer port.
User interaction	PicoBoard	A user interaction bug occurs when the designer tries to implement a way for the user to interact with the furry pet body and at the same time cause a sensor to read	Maya and Dino worked to embed the PicoBoard within their alien's cardboard box body. They made strategic slits in the side of the alien and attached a	The button extending device needed to be made taller, rather than changing the slit/paperclip implementation.

		this interaction so that some reaction can occur on screen.	paperclip to the slider so that a user can move the slider inside the pet's body by moving the paper clip outside the pet's body (see Figure XX.). However, when Maya and Dino tried to put a cork on top of the button so that pressing on the alien from the top would depress the button it doesn't work. They realize the slider slit/paperclip was too far down.	
Craftware	Craft	A craftware bug refers to any bug that concerns the craft materials being used.	Jamal worked on the physical appearance of his pet zebra. He attached some googly eyes to the front and a pom pom tail to the back using glue, then set the pet down and looked at it. The tail fell onto the table.	After much trial and error and abandonment of some eyes, Jamal used duct tape to adhere the pet.
Virtual Bugs				
Generic computer knowledge **(Computer)**	Computer	Generic computer knowledge bugs refer to instances where limited experience with computers causes a problem.	Carlos and Dino, forgot what they named their newest program version on workshop 4. So they opened a program in Scratch that made it seem like all their code from last	The group figured out to open the correct version of their Scratch program.

			workshop was lost.	
Incompatability of reused code (**Intentionality - Reuse**)	Reuse	Incompatibility of reused code bugs are a form of intentionality (Pea, 1986) bug that occurs when a student expects existing prototype programming code will accommodate a new design idea because the computer has a certain human like ability to understand what it should do.	Tegan expected the pet monkey to move all the way to the edge of the stage where she had imported a bunch of bananas. The existing "slider movement" code did not allow the monkey to walk that far to the right of the screen.	Tegan reworked the code that interpreted slider location and translated it via mathematical manipulation to an x coordinate that extended the width of walking.
Deletion of code presumed to be spurious (**Egocentrism**)	Reuse	Deletion of code presumed to be spurious bugs are instances of Egocentrism (Pea, 1986) and Missing (Cunniff, Taylor & Black, 1986) bugs where a student deletes a piece of pertinent code, usually because he or she attributes more meaning to the programming code he or she keeps in place.	Carlos pressed the PicoBoard button but did not hear the pet meow, as programmed. The meow functionality was in place but the meow sound import was deleted.	Carlos and Dino recorded a new sound.

Misplaced code (**Misplaced**)	New Code	A misplaced code bug, borrowed from Cunniff, Taylor & Black (1986) occurs when students create the correct code to achieve their objective but have put the code in the wrong physical (on screen) location. These bugs are pertinent to Scratch where each sprite & background has its own code screen.	Jamal wrote code to make the zebra he painted dance in workshop 12. When he tested the functionality, the background character danced while his zebra remained still.	Jamal copied and pasted his new code from the wrong character's scripting area to the zebra's scripting area.
Insufficient knowledge of new programming code (**Syntactic**)	New Code	Insufficient knowledge of new programming code bugs are syntax bugs or problems with how to express instructions (Kelleher & Pausch, 2005). With syntax bugs, a student thinks he or she understands what a chunk of new Scratch code will do but then realizes the command does something different than expected.	Jamal wanted the entirety of the song he imported to play when the zebra heard a loud noise. He used a forever loop to mean the song should play forever length of time until it was done. However, when he tested the change, the first two notes of the song played again and again and again because the forever loop got called over and over.	Jamal replaced the forever loop with a play until done code chunk.
Initialization missing or not updated **(Malformed**	New Code /Reuse	An Initialization missing (in the case of new code) or not updated (in the case of reused code) bug was taken from	Carlos programmed the alien to jump on a trampoline on the moon and end up in outerspace. It	Carlos did not realize that he must set the moon to be the initial stage if he wanted the stage to revert

Initialization)		Cunniff, Taylor & Black's (1986) conceptual bug by the same name. Although a specific variety of missing or malformed bug according to the authors, it occurred quite frequently in the research project. This bug occurs when programming code change the state of an object that must be put to its original condition when the program begins again.	worked when tested. However, when tested again, the alien started off in outerspace instead of starting on the moon as intended.	back to the moon every time he executed the program. He added initialization code to the beginning of the program to make the stage begin at the moon.
Uncategorizable (**Uncategorizable**)	New Code /Reuse	The uncategorizable category refers to bugs that did not fit the other codes because it cannot be identified.	Maya and Dino cannot figure out why the alien was placed so low down on the screen when they ran their program.	The facilitator and both students could not figure out where the bug stemmed from. We created a work around that overwrote the original placement.

With syntax bugs, a student thought he or she understood what a chunk of new Scratch code would do but then realized the command did something different than expected. Finally, the uncategorizable category was a catchall for any remaining bugs that could not easily be assigned to a code. For the most part, these bugs defied

categorization because the students and facilitator could not identify the root cause of the bug. In some cases these bugs were solved despite not knowing their origin and in others the facilitator told students to ignore the bug and developed a work around solution.

The codes were not intended to be exhaustive, but to represent the bug landscape I observed during the three sampled workshop days. The resulting bug catalog provided an illustration of the panorama of bugs students encountered and a lens through which to understand students' activities in the face of bugs.

What Did Debugging Look Like?

In the previous section I categorized the types of bugs students encountered during the project. In this section, I provide an overview of the student groups and the bugs they faced. Based on the data, I discuss trends, patterns and their implications.

Overall, the students did encounter a considerable number of bugs during the workshop (see Figure 37). Together, student groups encountered between three and 27 bugs.

On some days, some groups dealt with no bugs, like Rocky and Tegan on workshop day 12. However, since bugs and debugging are only one of a set of productive activities students could have been engaged in doing, like implementing design ideas, showing off their design to others, coming up with new ideas, working with the physical materials, stewing over what to do next, it cannot be said that groups with few bugs did less work or that those days were not fruitful. Day dreaming, observing others, and

starting over are all part of a host of activities acceptable to Constructionist learning and explain the importance of time in personally relevant projects (Papert, 1980).

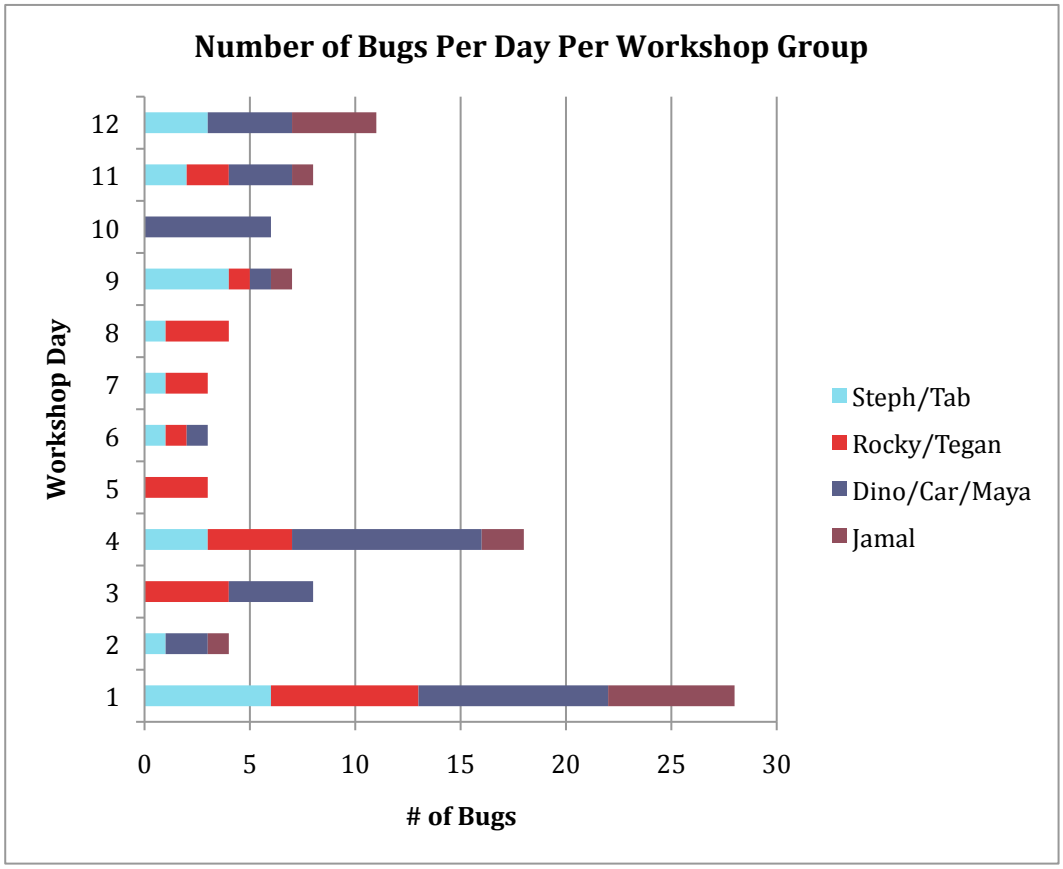

Figure 37. Overall bugs per day by workshop group.

This process is described as "diving in and stepping out" (Ackerman, 1996, pp. 29), where diving in refers to the deep personally connected way of learning through

developing a project, or accommodation of knowledge, and stepping out coincides with reflection, or assimilation of knowledge by stepping away from the project in a reflective way. This also helps explain why some days were big debugging, or diving in days, and others were more reflective, stepping out days, with little debugging activity. Of note was that there tended to be a lull in bug activity during the middle of the project, on days 5 through 8, for example. Looking at each group independently (see Figure 38), a similar ebbing trend appeared across all groups.

There were more debugging instances during workshop 1. This was likely because students encountered more bugs during the structured debugging task workshops, when the intent was to have students work through a list of itemized, solvable novice bugs that gradually increased in complexity. Therefore the structured, intentional bug-solving environment was more effective at getting students to face bugs than their strict independent design work. However, as the independent project commenced and perhaps students began to feel comfortable trying to implement their design ideas, for instance on day 4, the number of bugs was large, indicating the divide between students' ideas and their programming capabilities was also large.

The craft materials arrived in class on the 5th workshop, explaining why each group experienced a dip in bugs as they played with the new materials and thought about how to transform them into a fluffy companion. Evidence from field notes suggested the next several workshop days were comprised mainly of tangible pet development and little new programming efforts. As described in Chapter 4, Jamal, for example, did no programming or computer work at all between workshops 5 and 11.

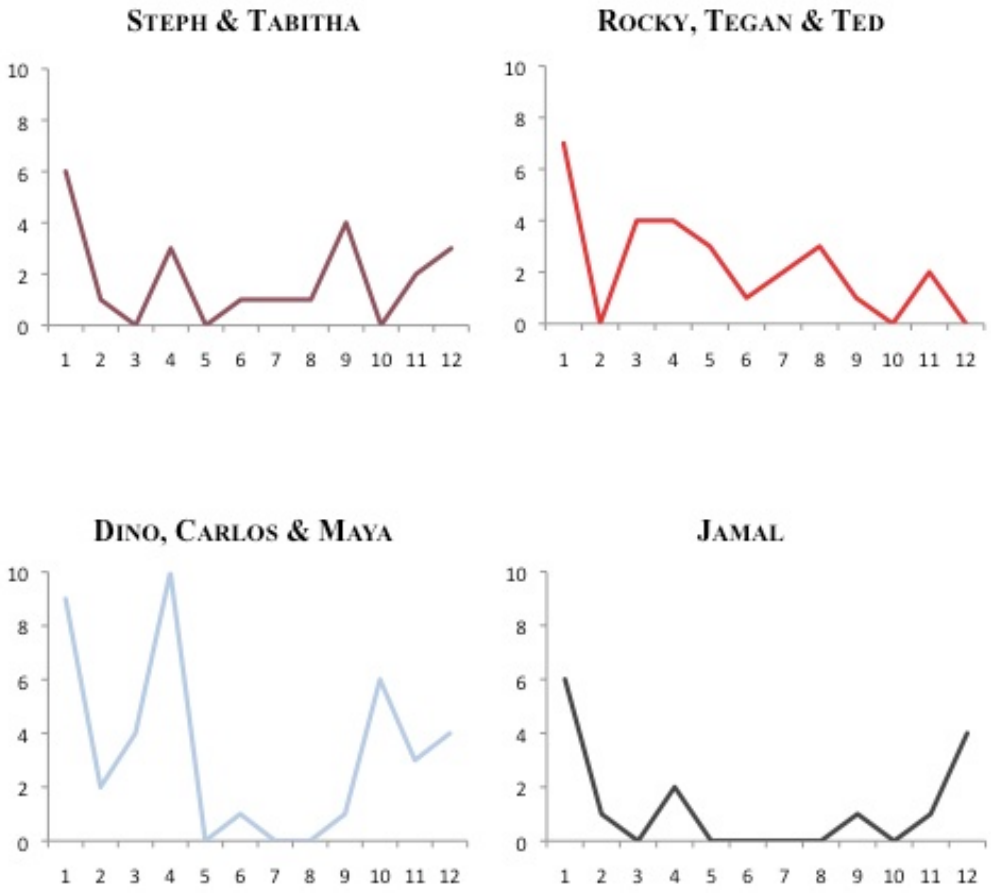

Figure 38. Number of individual bugs by student group per day.

There were fewer bugs during this central time, the middle of the project. Whether students' focus on tangible creation correlated to fewer bug encounters or whether students' stamina for the project and general enthusiasm waned during the middle of the project accounting for less programming and therefore fewer bugs was not clear. Both ideas will be explored further later in this chapter.

Finally, the last two days of the project saw a renewed flurry of bugs, with eight bugs during workshop 11 and 11 bugs during workshop 12. During these final days, students began to ramp up efforts to complete their projects in time for the design exhibit, held after school. Again, Jamal programmed his entire project on the 12th workshop day. The culminating event and along with it the promise of course credit, was effective in prompting students to complete their work. Finishing projects meant tying up loose ends and ensuring everything worked as expected. This process naturally uncovered bugs, especially when the tangible pets and virtual programs were largely developed in isolation and had to be somewhat integrated together, though not always effectively, as explained in chapter 5.

Accounting for Overall Numbers of Bugs

The overall number of bugs encountered during any given workshop day may seem relatively low. For instance, four groups of students faced 11 total bugs during the 45-minute workshop 12, which was a lot of bugs for the project in relative terms. In reuse and modification of code, simple changes, like making a sprite meow instead of bark when the button was pressed, are very different in programming complexity than big changes, like making a sprite do back flips instead of meowing when the button was pressed. Students were able to make simple changes without much difficulty. However, in general, students sometimes struggled with three aspects of their pet project designs. First, students struggled to come up with their own design ideas. Second, students struggled to sustain interest in developing new functionality, especially during the middle of the project. Third, students had trouble conceptualizing a narrative focus for their

"pets". By a pet narrative I mean an encompassing story for the companion, for instance a story would include personality traits, a setting, needs and desires, and ways of addressing them. For instance, a dog needs exercise and food every day and might be made more content by being petted, fed treats, given a bath or a fluffy bed to sleep on.

One of the exciting potentials for using interactive pets as a design technology was to tap into the idea that digital pets have "alive" qualities. Pets, by their definition, need to be nurtured. Pets, as companions, promote attachment. I hoped this attachment would be apparent not only for the user but also for the designer, the students. I hoped students would naturally incorporate some of these ideas about the nature of pets and pet ownership into their pet projects. However, most of the students' pets were piecemeal functionality with little or no overall direction. The closest a group got to an integrated narrative was Carlos' group who created an alien that ate people's hands, rode a magic carpet and jumped on a trampoline to the moon. But yet, the alien had no needs, fears or desires, or a name for that matter. Because students' projects were less an integrated whole "pet" and more small cool bits of reuse, the groups spent less time programming new functionality for their pets. On a similar note, students spent a great deal of time on the physical appearance of their pets. This led to fewer bugs overall because students programmed less. As a result, the groups were implementing fewer entirely new ideas than I expected and therefore encountered fewer bugs than expected as well. However, because the project was 12 workshop days, students did encounter many bugs in total, enough to spend a good deal of time dealing with bug encounters. Carlos, Dino, and Maya had the most bug instances (39). Recall, their group implemented intricate eating,

flying and jumping functionality. Jamal had the fewest bug instances (15). Recall Jamal did no programming at all for seven of the workshops.

Bug Fates by Group

Students encountered a large quantity of bugs overall meaning that students had many opportunities to deal with bugs in different ways. I expected bugs to occur and was interested in what students did when encountering instances of bugs during their projects. Looking at the overall project, the majority of bugs students faced, 81%, were resolved (see Figure 39) Most often, when students encountered a bug, they identified and fixed it. This was encouraging because I wanted students to develop debugging strategies and be willing to persevere through the process of finding a solution to bugs. Research shows that in personal open-ended projects, students will abandon their ideas, scrapping large portions of code, instead of facing difficult bugs and coding issues (Kafai, 1996). In this project, only 8% of bugs were ignored and 5% were worked around, meaning students either deleted code or altered a design idea to more easily accommodate a bug. During the project, only 6% of the total bugs remained unresolved. There were three main reasons for unresolved bugs. First a bug could occur that did not directly affect students' design ideas and could be left. Second, the students and the facilitator could not identify and resolve some bugs. Third, a rare case, students gave up on a small number of bugs even though the bugs detrimentally affected the functionality of their projects. From these data, despite expectations from previous research, students were fairly successful at debugging and left bugs unresolved, ignored or worked around relatively infrequently.

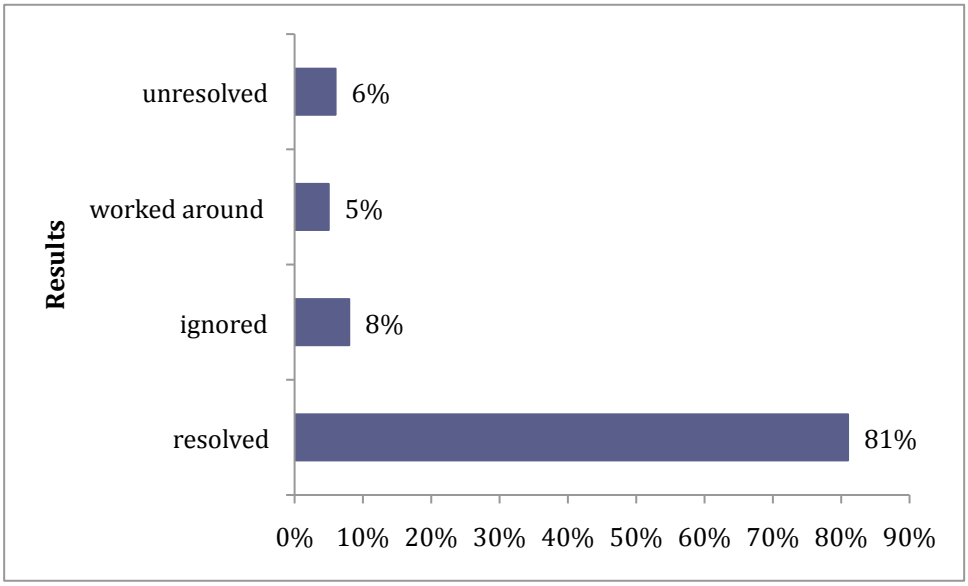

Figure 39. The fates of all bugs encountered during independent design workshops by percentage.

Another factor affecting the number of bugs encountered was the length of time spent on debugging. Since the data show students were most often solving the bugs they encountered, instead of ignoring them or leaving them unsolved, I knew students were engaged in a good deal of debugging. In the data, if groups were engaged in solving one bug for a long duration, it would appear as one bug. Simply counting the number of bugs provided an inadequate picture of how much debugging really happened. For example, in independent workshop sessions, Tabitha and Steph's group spent nearly 20 minutes on a single bug, and Dino, Carlos, and Maya's group spent more than 25 minutes on a bug on two different occasions (between workshops two and three and again between workshops four and five). These data suggest students sometimes spent more than half a workshop

period engaged in finding and fixing one bug and provide another reason for a smaller number of total bugs, in some cases. For these students to spend a significant amount of time struggling with a single problem was unusual because they were used to solving copious drill problems during school time in their make-up packets.

Individual Student Groups and Bug Fates

Overall, students solved the majority of the bugs they encountered, but individually, student groups had varied success fixing bugs, choosing sometimes to ignore, work around or leave bugs unresolved (see Figure 40 and Table 14). The differences in bug fates between groups can be explained in part by the groups' working styles. Jamal, a student who chose to work alone, had the fewest bugs (15), but resolved them all. His working alone may account for this because he did not have to reconcile others' opinions on whether to persevere and how to proceed. He also did the majority of programming on the final day when he was invested in making the pet he had worked so hard to perfect physically interact in the way he wanted (recall from Chapter 4 that he was the only student to ask to stay late). Conversely, ignoring bugs suggested a willingness to have partially functioning programming code, to have imperfections in code and to be able to move on to other ideas when one was not working as expected. Two groups, the hippo/unicorn group and the monkey group, had the highest percentage of ignored bugs. Tegan, Rocky, and Ted, the monkey group, had a style that included developing functionality into the computer program piecemeal, as ideas arose. The group also had the highest percentage of worked around bugs, suggesting that when they discovered a problem with implementing a new idea, the group sometimes altered their

ideas to get around difficulties. Steph & Tabitha, the hippo/unicorn group, struggled to get their project underway and had trouble maintaining continuity between days, most workshop sessions they introduced new pieces of functionality haphazardly along with new characters. The girls lost all their programming code one of the final days of the project and as Tabitha explained in interviews, they suffered from frustration and lack of confidence. This reflected in the girls' bug fates. The girls had the lowest percentage of resolved bugs, and highest percentage of both ignored and unresolved bugs.

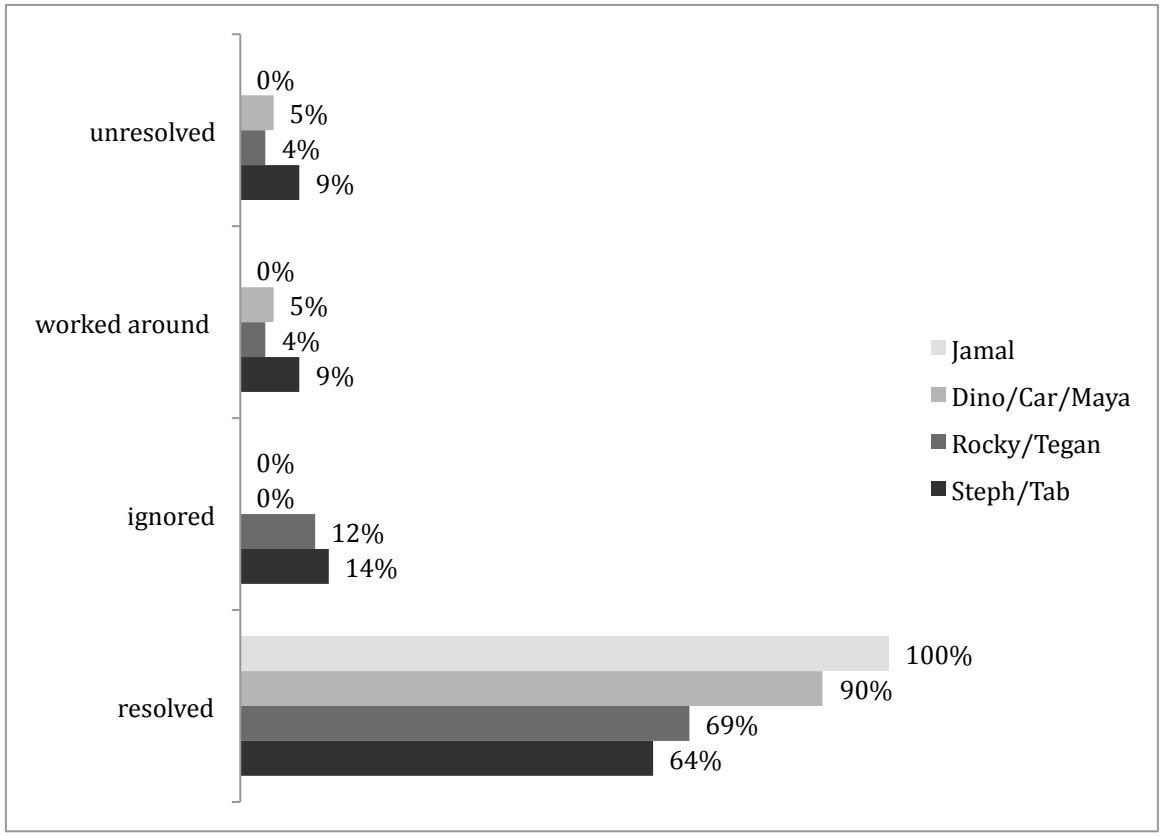

Figure 40. Students' bug fates by group (in percentage).

In contrast, Carlos' working style, from the alien group differed from the others. Carlos was dedicated to his design ideas, coming up with the grand picture of his virtual pet's complex functionality from the start and working tirelessly to succeed in his endeavors, as described in Chapter 4. He did not participate much in the physical pet design. The group had the most total bugs (39), a high percentage of resolved bugs (~90 %), no ignored bugs and very few worked around or unresolved bugs. Interestingly, the unsolved bugs both came on the day Carlos was absent and his partners, who did not program previous to the absence, struggled to figure out how to run the program let alone make changes to code that was not working properly. The episode that followed between Carlos and Dino helped further illustrate how serious Carlos was about realizing his ideas. After a long bout of bugs in workshop four, Carlos ran into yet another problem. He programmed the alien to ask to ride the magic carpet then ride over to the edge of the screen, jump on a trampoline and end up in an outer space scene. The final difficulty, the 9th bug of the day, was that the alien did not get off the carpet once he arrived in the stars. Dino created a workaround that deviated entirely from Carlos' idea. Dino had not participated in solving bugs that day except for this instance where he essentially chose to tell the alien not to ride the magic carpet, thereby alleviating the need for the alien to get off the carpet.

Carlos	You messed it up! You're supposed to write three letters. Enter. It rides it. Uhn! To the stars!

Dino	Well why don't you just press "no" so he doesn't get on the carpet and you don't have to worry about him getting off in the first place?
Carlos	Cause I want to.
Dino	Uuuhhh. Fucker.
Carlos	Alright he didn't fix anything. He just fucked it up. (*talking to facilitator)*

Carlos blamed Dino for the way he worked around the problem, "You messed it up!" Then Carlos showed Dino how to correctly use the functionality. Dino contended that it would be much less work to do it his way, "Then you don't have to worry about him getting off". Finally, Carlos rejected Dino's idea again, explaining his motivation to do the work, despite realizing how hard it had become, "Cause I want to." Carlos, regardless of the fact that he had to fix nine bugs that day and cannot figure this bug out, regardless of the amount of extra effort it will take to make the alien flying idea work.

Modes of Debugging: Student Problem Solving Activities

During the project, students encountered a variety of bugs, solved many bugs and dealt with others in accordance with how they worked on their projects in general. However, still left to explore was precisely what novice programmers did when faced with a bug. One of my fears that I noted in my field notes was perhaps students simply always sought immediate help from the facilitator, never allowing themselves to explore potential problem-solving approaches. Even more concerning, perhaps, when seeing students struggle, I had unintentionally provided students with step-by-step instructions

for fixing bugs more often than I intended, never enabling students to learn debugging strategies.

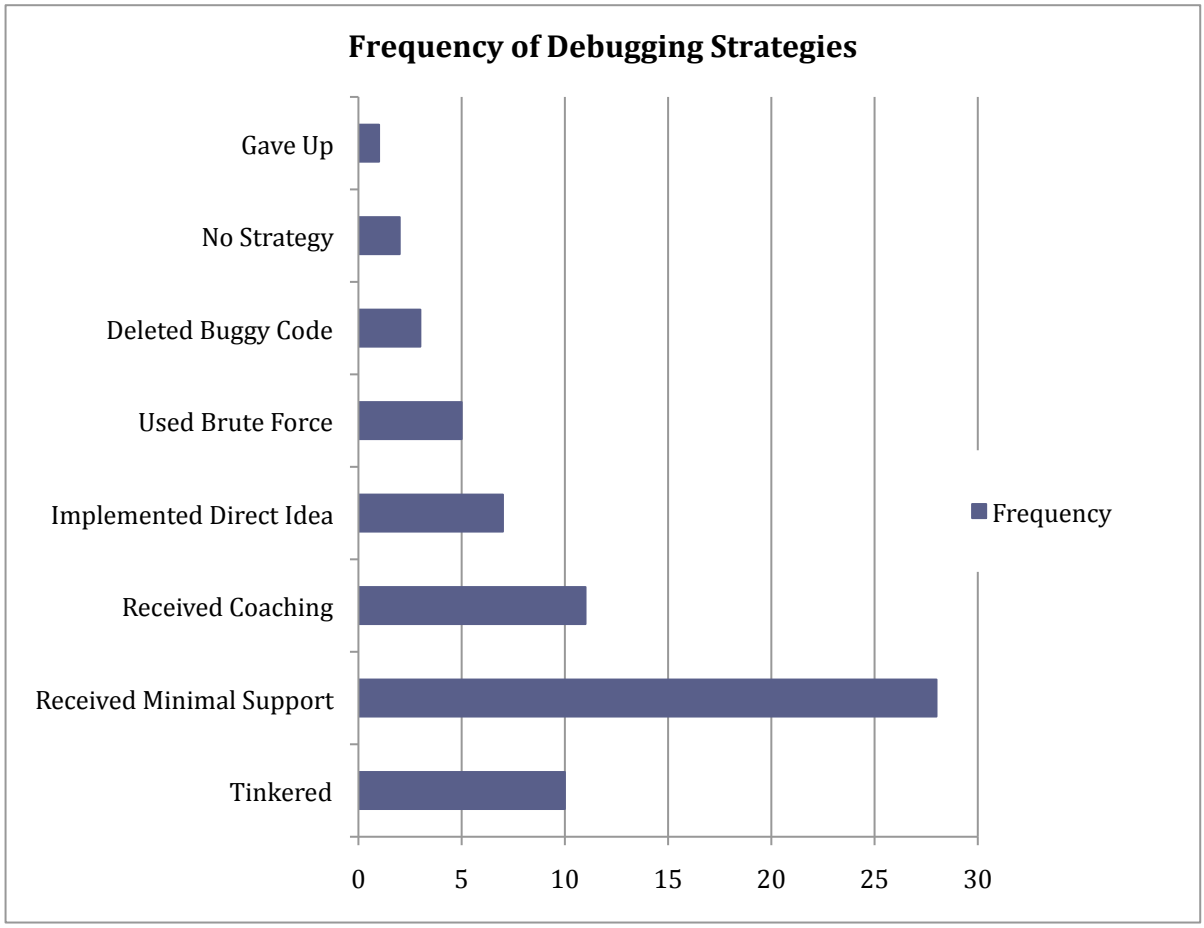

Figure 41. Frequency of debugging strategies used by students.

It turned out that most often, students did ask the facilitator for advice or help during the course of debugging (58% of the all debugging activities were either *asked facilitator, received minimal support* or *asked facilitator, received coaching*) (see Figure 41). Students cannot be blamed for employing this strategy, as asking a knowledgeable adult is a sensible activity especially in a school setting. However, I developed a distinction in the codes between asking the facilitator and receiving minimal support (42% of all debugging activities) and asking the facilitator and receiving coaching (17% of all debugging activities), with the latter reserved for times when the facilitator gave explicit instruction and the former for instances when the facilitator provided more general strategy tactics to guide students towards finding their own solutions (examples to follow). The difference between the pedagogical methods being teaching students ways of approaching debugging problems, a strategy modeling method, versus helping students out of an immediate dilemma, a fire fighting method with less potential for extrapolation. Also of note was the number of other activities in which students also engaged. For instance, students *tinkered*, *implemented effective direct solution ideas*, *used brute force repeated failure*, a frustration inducing novice strategy, *deleted buggy code* and sometimes, but not often, *gave up*. Significantly, students sometimes tried their own strategies to identify and fix a bug first, and then baring failure asked for help or asked first and then realized they could solve the bug on their own.

Table 13

Debugging Modes, Counts, Frequencies, Student Group Employment of Modes and Percentage of Bugs Where Mode Was Employed by Group

Modes	#	Frequency	Student Group	Student Group Employment	Percentage of Bugs Where Mode Was Employed
Tinkered	9	14%	Jamal	4 in 7 bugs	57%
			Tegan, Rocky & Ted	2 in 5 bugs	40%
			Carlos, Dino & Maya	2 in 14 bugs	14%
			Steph & Tabitha	1 in 10 bugs	10%
Asked facilitator, received minimal support	28	42%	Carlos, Dino & Maya	13 in 14 bugs	93%
			Jamal	5 in 7 bugs	71%
			Tegan, Rocky & Ted	4 in 5 bugs	80%
			Steph & Tabitha	6 in 10 bugs	60%
Asked facilitator, received coaching	11	17%	Carlos, Dino & Maya	7 in 14 bugs	50%
			Jamal	2 in 7 bugs	29%
			Steph & Tabitha	2 in 10 bugs	20%
			Tegan, Rocky & Ted	0 in 5 bugs	0%
Implemented direct idea	7	10.5%	Jamal	2 in 7 bugs	29%
			Carlos, Dino & Maya	3 in 14 bugs	21%

			Tegan, Rocky & Ted	1 in 5 bugs	20%
			Steph & Tabitha	1 in 10 bugs	10%
Used brute force repeated failure	5	7.5%	Steph & Tabitha	3 in 10 bugs	30%
			Carlos, Dino & Maya	2 in 14 bugs	14%
			Tegan, Rocky & Ted	2 in 5 bugs	40%
Deleted buggy code	3	4.5%	Jamal	1 in 7 bugs	14%
No strategy	2	3%	Steph & Tabitha	2 of 10 bugs	20%
Gave up	1	1.50%	Steph & Tabitha	1 of 10 bugs	10%

Minimal Support Versus Coaching

When students asked for help, two possible outcomes arose. In line with the pedagogical approach outlined in the curriculum design section (Chapter 3), the first outcome provided students with just enough direction to allow them to be independently successful. The *asked facilitator, received minimal support* code was reserved for instances of modeled meta-debugging skills, intended to foster developing strategies that transfer to other debugging instances, rather than explicit solution support for a specific bug.

The following excerpt was representative of how the facilitator interacted with students who asked for help and was representative of the type of minimal support given.

Tegan	Why does he keep turning green?
Facilitator	*from across the room* Check the eating. Where's he eating? *(directs student to look for eating code to identify the bug)*
Tegan	I don't know. Um. Right here. *locates eating code*
Facilitator	Ok. *comes closer to look at screen* So you're changing costumes. But you didn't make a costume that's green?
Tegan	No.
Facilitator	So look through here *(the eating code)* and find out what might be changing his color.

Note that I did not instruct the student how to find or fix the bug. The process of support was to encourage students to verbally elucidate the problem, then narrow the problem space, and productively focus their efforts without eliminating the students' sense of confidence and accomplishment in finding a solution. I needed to figure out quickly what was going wrong and at the same time help Tegan figure out what was going wrong for herself. I narrowed down potential reasons for the bug, of which I had two hypotheses, a costume issue or a directly coded visual alteration. Finally, I helped Tegan restrict the problem space by directing her to look through a subset of

programming code for something that would turn the character green, focusing her efforts and reminding her specifically of the problem. In this process, known as *cognitive apprenticeship* (Brown, Collins, & Duguid, 1989), is common in other educational approaches like Problem Based Learning (Barrows, 1996), where experts model expert practices and support novices in adapting the practices for themselves. Although Brown et al. (1989) described the method as coaching and fading, I reserve the term coaching for more specific and directed debugging support, the second potential outcome from a student's question.

Carlos, Dino, and Maya most often engaged in *asked facilitator, received minimal support* (in 90% of bugs, see Table 13) because, as discussed earlier, Carlos' stated in an interview he liked being told exactly what he did wrong so he could learn from his mistakes. Carlos almost always asked for help when faced with a bug, oftentimes figuring out the solution before receiving any assistance, but sometimes needing coaching (50% of the time), probably because his coding ideas were complex. Carlos created an effective facilitator calling mechanism. When he got stuck, Carlos would continuously press the PicoBoard's button, making the alien noise repeat again and again, until I ignored all the other students and came over to help. But far from being helpless or inadequate at solving bugs, Carlos also had a high frequency compared with other groups in implementing spontaneous solution ideas (21%).

For *asked facilitator, received minimal support* the facilitator never took control of the computer mouse or keyboard or gave step-by-step instructions for solution. When either of these methods was present, the code *asked facilitator, received coaching* was

used. Coaching was not thought to be a pedagogical ideal, but was reserved for instances when students seemed visibly agitated, discouraged or unreceptive to ask facilitator type modeling. In an effort to keep students engaged in the project, I sometimes resorted to coaching students through difficult situations, especially very complex or frustrating bugs by modeling the exact programming code I would implement. Surprisingly, considering the difficulty of the task for novices, coaching represented only 17% of all debugging activities.

Students most often asked the facilitator for assistance as part of a larger set of debugging activities for each bug encountered. The results suggest that students did not always immediately and merely ask for help. For instance, in workshop 9, Tegan and Rocky encountered only one bug, a hardware bug that occurred because they had inadvertently plugged the USB cord from their PicoBoard into the incorrect hole in their laptop. At first Rocky *tinkered* by testing the issue, moving the PicoBoard directly in front of himself, adjusting the board slightly, and testing the results several times. Then Rocky scrolled through the monkey's scripts to see if he could identify the problem. Then not reaching a solution, Rocky *asked facilitator, received minimal support* and was able to successfully solve the bug because the facilitator asked Rocky a series of probing questions about the bug until we figured out together that the cord was incorrectly attached. In this instance, Rocky used two debugging methods, *tinkered* and *asked facilitator, received minimal support*.

Sometimes students' instincts were to ask questions first, probably a result of years of school endorsed behavior, and try their own solution strategies after receiving

some encouragement and an indirect hint. For instance, on Carlos' group's fourth bug on the 4th workshop day, Carlos encountered a bug because he had programmed the alien on the magic carpet to fly to another stage after touching the trampoline. However, the alien and magic carpet just stayed on the moon. The problem was that the new background was in place but not being accessed when the event occurred. In this case, Carlos first *asked facilitator, received minimal support*. I asked him to recreate the error and then tell me whether the code he had just created was being accessed, which would be highlighted when run. As soon as Carlos ran the program, he realized on his own his error was due to the fact that the alien sprite and the alien + magic carpet sprite were different. The code he had just created had been misplaced under the wrong sprite. After figuring this out independently, Carlos then *implemented the direct solution idea*. I guided him to check whether the code he had written was being accessed and from that he determined that the sprites were different, the code was in the wrong place, and that he needed to edit the scripts for the intended and misused sprite. After solving the bug, Carlos uncharacteristically, he rarely expressed emotion, showed his pride in figuring out the bug fix on his own by stating, "Oh I did it! Yessa!"

Implemented Direct Solution Idea

In some cases (<11% of all debugging activities), students encountered a bug and almost immediately knew what had gone wrong and how to fix it. In the following example, Tegan *implemented a direct solution idea* to a tangible hardware bug. When she ran her program near the beginning of workshop 4 she noticed the monkey was not eating, as he should have been when she connected the alligator clips, from attached to

the PicoBoard, to each other. Tegan acted surprised that the eating did not occur but then immediately replugged the alligator clips into the board and resolved the issue.

Tegan	(*touches clips together)* Wait. Why isn't he eating? (*pulls over PicoBoard, tightens alligator clips in ports, humming)* *(touches clips together)* *(monkey eats on screen)*

This example was described as *implemented a direct solution idea* because Tegan was able to resolve the bug by directly trying an idea that occurred to her. If she had gone through multiple iterations of possible ideas and tests before finding one that worked, Tegan's activity would have been coded *tinkered*. Carlos' group (in 21% of bugs) and Jamal (in 29% of bugs) most often engaged in this activity compared to other groups. Jamal enjoyed working alone and wanted to figure out everything about the project, including bugs, on his own, asking for facilitator support only as a last resort.

Tinkered

The *tinkered* activity (14% of all debugging activities) described how students, when faced with a bug they could not solve right away, many times chose to try out several potential fixes by changing a bit of code, retesting, then returning to the previous code and trying out another fix. The term *tinkered* reflects the playfulness of this strategy. Jamal engaged in the most tinkering (in 42% of bugs) and Tegan, Rocky, and Ted in the next most (in 20% of bugs).

Used Brute Force Repeated Failure

Used brute force repeated failure (<8% of debugging activities overall) referred to an instinct students sometimes exhibited to literally bang on the technology until something changed, especially Steph and Tabitha (in 30% of all bugs). Watching students attempt over and over again to fix something by repeating the same failed execution process with increased force and frustration, to learn only that the program continued to do the unexpected thing again and again, was an act of desperation. An example of *used brute force repeated failure* might be a student who pressed the PicoBoard button again and again and again eventually pounding on the button and continuing to express dismay at the continued result. This type of activity is well documented in design realms as the gulf of execution/gulf of evaluation problem (Norman, 1991). The student, in this case, was unable to reconcile the divide between her action and the problem that ensued. Eventually the student must change her tactic, but she often stuck to the original approach for an unseemly length of time.

No Strategy

The *no strategy* code (3% of all debugging activities) was used in situations where students identified a discrepancy between intent and outcome but pointedly decided to ignore the problem. The bug does not disappear, but often students could continue work on another portion of the project without interference from the bug. Alternatively students sometimes decided to simply leave the problem and abort their original intention. Steph and Tabitha ignored the most bugs, employing no debugging activities for 20% of their total bugs.

Deleted Buggy Code

Students sometimes decided to scrap an idea instead of trying to reconcile a bug. In this case, the *deleted buggy code* (<5% of all debugging activities) code was used to describe the physical erasing of programming code to make the bug go away. For example during workshop 4, Tegan, whose group employed *deleted buggy code* the most often (in 40% of all bugs) had trouble with some existing prototype code that she and her partners had tried to remix from a car to a helicopter that flew around. Instead of trying to figure out why the helicopter kept ending up upside down or sideways after its flight, she just deleted the offending sprite and all its scripts. She got rid of the helicopter and corresponding code without hesitation, throwing in the towel on all the work the group had done to change the original programming code.

Gave Up

In an extreme case, only one documented instance during the three representative workshop days, a student *gave up* on a bug that was causing their program not to run properly. The code described Tabitha's utter frustration and inability to work through a bug on the 9th workshop day. The *gave up* code, used only once, provided a valuable counter to emphasize the unexpected success of most novice students in the project. The *gave up* code marked Tabitha's decision to withdraw from developing her project, although she did attend the design exhibit. During a bug Tabitha found during workshop 9, Tabitha *used brute force repeated failure*, already described as a frustration enhancing activity, then *asked facilitator and received minimal support*, made little progress and finally *gave up*, abandoning the bug and programming. She said, "I am not doing any

more". And indeed, she did not. Tabitha was notably absent the next two workshop days, reappeared the final day, workshop 12, but refused to touch the computer or speak to Steph, who had inadvertently deleted all Tabitha's dysfunctional programming code during workshop 11. This combined with an outside of school incident caused tension between the girls.

The students had tremendous overlap in their individual debugging activities; seven codes accounted for all student employed bug-solving strategies. The strategies stretched along a continuum from productive, expert-like to unproductive, and even disadvantageous. The asked facilitator codes, reminiscent of doing school, were expected. However *received coaching*, the most helpless of activities, was used minimally and students were most often successful in identifying and fixing bugs with limited, strategy modeling support given by *asked facilitator, received minimal support*. *Tinkered* and *implemented a direct solution idea* are both productive, positive strategies used by experts and in many cases also developed and employed by the novice students. In contrast *deleted buggy code*, *no strategy*, and *gave up*, despite the program not working correctly, are unproductive debugging strategies. Finally, *used brute force repeated failure* was a convoluted, deleterious and ultimately unproductive strategy derived from principled intentions mixed with inexperience and frustration.

Post Assessment on Bugs

The previous sections describe how students in their groups approached bugs in situ during the project. How much problem solving skill related to debugging individual

students acquired during the project, however, has not yet been discussed. Debugging skills are important for a number of reasons including the idea that "errors benefit us because they lead us to study what happened, to understand what went wrong, and, through understanding, to fix it" (Papert, 1980, p. 114). In this section I discuss the debugging task assessment administered after the project was completed.

The Debugging Assessment Task Explained

In the debugging task assessment, each student identified in the beginning of the project for in-depth study was asked to complete four debugging tasks on the computer while the facilitator observed. Students were asked to think aloud as they worked. Occasionally the interviewer, who was also the project facilitator, would ask the student what he or she was thinking to prompt think aloud behavior. Students could ask questions but the interviewer specifically told students she might not supply answers because she was interested in how the students were thinking independently. The students were provided with a new, never before seen prototype Scratch project (see Figure 42) a plugged in PicoBoard and a sheet of paper describing the debugging tasks (see Table 14 for debugging task assessment protocol). Students were told they could skip to any task they wished and come back to others as they saw fit but also that the tasks generally increased in difficulty. The tasks were developed in direct relation to errors I observed students encounter, and encounter frequently, and modifications to code students often wished to enact during the project. Tasks ranged from changing a sound associated with a particular interaction, to understanding how and when to initialize objects, to understanding the computer needed explicit instruction to wait between switching

costumes so the human eye could detect the change, to understanding the relationship between actions concerning two sprites at once and modifying how and when the sprites broadcasted and received broadcasts. The debugging assessment was video-recorded. Students had as much time as they wished to complete the assessment.

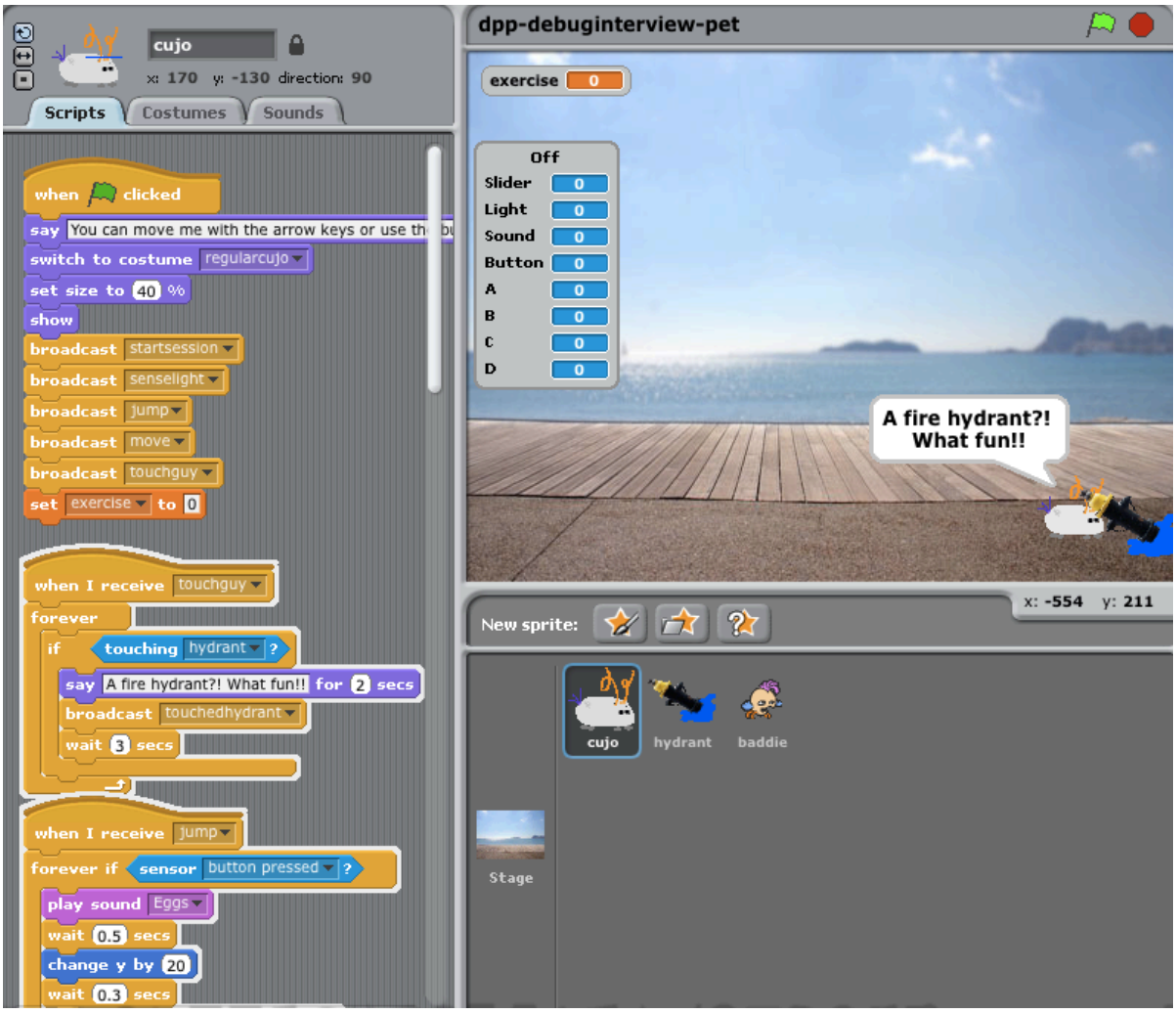

Figure 42. The debugging assessment Scratch project.

Table 14

The Debugging Task Assessment Items, Programming Concepts Covered and Text from the Worksheet

Bug	Idea Behind Bug	Programming Concepts Covered	Bug Text
1	Understanding the computer needs explicit wait instructions between switching costumes	Events; Basic Programming Statements;Multi-media: Costumes, Wait Command	When it gets too sunny, Cujo is supposed to put on his shades. But it's not working. Fix it so we can see that he puts his shades on.
2	Changing a sound associated with a particular interaction	Input: Keyboard Pressed; Multi-media: Sound Editor	Cujo can walk left, right and up and down. When he walks there is the sound of a horse galloping. But I don't like the sound. Make a new and better sound for when he walks.
3	Understanding when and how to initialize sprites	Initializing Sprites,	Sometimes when you start the fire hydrant is upright and sometimes it has fallen over already. I want the fire hydrant to always start out normal when you start a new session.
4	Understanding the relationship between actions concerning two sprites at once. Identifying how, when, and why sprites broadcast and receive broadcasts.	Event Handling Across Sprites	When Cujo touches the fire hydrant, it falls over and spills water. When that happens, the baddie (check the characters) is supposed to fly out, but he's not. Figure out how to make him fly out. The scripting for flying out is already there - the baddie just doesn't know when to go.

The table (Table 15) shows students' successes with the debugging task assessment. For each correct independent or minimally supported answer, students received two points. Minimal support and moderate support were reserved for times when a student was stuck and asked for explicit help. For minimal support to be coded, I provided help akin to the *asked facilitator, received minimal support* code used earlier. This meant that I might ask the student some questions about what they were doing or

read the code where the student was looking or tell the student they seemed to be in the right spot, but would not provide any specific help on the debugging fix. On the other hand, moderate support meant I guided a student towards understanding the bug given but then the student interrupted the explanation to provide the correct fix. I felt that a student who determined part of the bug fix should be given some credit for partially resolving the bug. For each answer achieved with moderate support, students received one point and for each answer incorrect, meaning students either gave up, received step-by-step coaching to achieve the answer or made a fix that was unsuccessful but did not change the fix to be successful, students received zero points.

Results from the Debugging Task

In general, students were quite successful on the assessment. On average, students received a score of 67% correct. However, more telling was that two thirds of the students were able to achieve a score of 87.5% or better, with two perfect scores. Dino struggled to connect to the project at all, and unsurprisingly was unable and unwilling to make progress on the assessment. Dino worked on the first bug task for almost four minutes and then declared, "I don't know. I'm just not even going to try. I hate computer shit." When I asked if he would like to move on to another bug, he said "no". When I asked if he would like to quit he agreed. Tabitha, who struggled to understand programming and conceptualize of the project as a whole, was able to make progress on several of the tasks and solved one independently.

Table 15

Students' Debugging Task Assessment Results. Students Received Two Points for Correct Solutions and Solutions with Minimal Support and One Point for Solutions with Moderate Support

	Task 1	Task 2	Task 3	Task 4	Total
Tegan	Correct (with moderate support) = 1 point	Correct = 2 points	Correct = 2 points	Correct = 2 points	87.5%
Rocky	Correct (with minimal support) = 2 points	Correct = 2 points	Correct (with moderate support) = 1 points	Correct = 2 points	87.5%
Carlos	Correct = 2 points	Correct = 2 points	Correct = 2 points	Correct = 2 points	100%
Dino	Incorrect = 0 points	Not Attempted = 0 points	Not Attempted = 0 points	Not Attempted = 0 points	0%
Tabitha	Incorrect = 0 points	Correct = 2 points	Incorrect = 0 points	Incorrect = 0 points	25%
Jamal	Correct = 2 points	Correct = 2 points	Correct = 2 points	Correct = 2 points	100%

Even though her overall score may seem unimpressive, given Tabitha's struggles during the five weeks, for her to be willing to participate in the individual assessment and to correctly identify and fix a given bug within it was quite remarkable. Also Tegan

would have solved all four bugs without support had she realized there was a piece of paper covering the light sensor on the PicoBoard making the sensor transmit a low instead of the usual high number. In general, the students were highly successful at identifying and fixing bugs of varying difficulty in prototype code after approximately ten hours of total workshop time working with the tools.

For some students, the debugging task assessment was difficult. They did not prepare ahead of time for the assessment and some students mentioned they were uncertain how they would perform when when having me watch their progress. However, despite mild protests on some accounts and the exception of Dino, all the students attempted the debugging tasks with varied success (see Table 15) In many cases, students needed minimal or moderate support from the facilitator to make progress on some bugs, but then were successful in fixing the bugs. Note that debugging strategies were never explicitly discussed during the project; students learned only through direct experience with bugs during the debugging task day, where the bugs were intentional and provided, and through their independent project work.

What I noticed during the assessment was that the students felt personally responsible for fixing the errors given. For the majority, it was important to the students to do well, even though there were no grades or ramifications for doing poorly. Students had already received their elective credit for completing the course and would not receive a grade from me of any kind. For this reason, I chose to provide some support to students who asked for it in the form of asking questions, telling students they had correctly identified the bug and/or helping them translate code language into English. For example,

in the following excerpt, Rocky, asked me to show him how to fix the two bugs he did not immediately fix on his own without any support.

Rocky	So I got two out of three?
Interviewer	Yeah.
Rocky	I mean two out of four? Ok. Now show me how to do it.
Interviewer	You want me to?
Rocky	Yeah.

For Rocky to spend extra time learning about the bugs he could not quite get right on his own was impressive to me. When I began talking about the first debugging task, he immediately knew and fixed the bug on his own. I refer to this episode as one of providing minimal support because I did not provide any explicit assistance or suggestions, I only told Rocky he had correctly identified the bug and then interpreted the computer code already written in that section of the program.

Interviewer	So, the sunny part. Um. Where's the. You were in the exact, in the exact right spot here. That he's sensing the light and then he's switching to that costume and then switching back. And if you look at this costume.
Rocky	I just have to have him Wait.
Interviewer	(*nods.)* Why don't you fix it?

Rocky (*dragging a wait command from the menu to the scripting area and putting it in between the costume change commands)*
I should have known that.

Rocky immediately knew how to fix the bug after the interviewer explained to him that he was "In the exact right spot" for identifying the bug and then proceeded to translate the code there into common English, "He's sensing the light and then he's switching to that costume and then switching back." After fixing the bug himself, Rocky added, "I should have known that."

In another example, Carlos, who solved three of the four debugging tasks quickly, went back to fix the fourth task after bouncing around and completing the others without support. When I asked Carlos whether he wanted to try number four again he said, "Yes. I have to finish it". Carlos felt compelled to get all the answers right and was motivated by this compulsion. When he did solve the bug on his own he added, "Alright. Alright then." This again, from a student who had failed so many times in traditional school that he could no longer graduate and who had no academic incentive to do well on the assessment. Carlos' pride in his abilities propelled him to persevere through successfully fixing all four bugs on his own.

The fact that students expressed a desire to do well on the assessment and had a personal stake in finding and fixing the bugs given was highly satisfying. The exception to students wanting to do well on the assessment was Dino, who was so frustrated with his experience in the project that he worked for only a minute on the first bug before declaring that he did not want to continue with the assessment. He then proceeded with a

poignant and in-depth interview about his experiences, see Chapter 7 for details, which was helpful in allowing me to understand the root of his frustrations and provided me with many modification ideas for a next iteration of the project, as discussed in Chapter 8.

Conclusions About Bugs

One main focus of this project was to engage novice programmers in debugging as a way of learning aspects of computer programming and programmatic thinking. Students dealt with many bugs both those designed and provided for them and those that occurred in situ. Furthermore, students most often figured out ways to solve the bugs they encountered and oftentimes were able to do so on their own or with minimal support. Also, students were generally successful on the debugging task assessment. Students showed they were, on the whole, interested in finding and fixing the bugs. For instance, a student, Rocky, spent independent time engaged in finding out answers to debugging tasks on the assessment he could not complete on his own. Students revealed a personal stake in doing well on the post-project assessment despite the fact that there were no academic consequences to doing so.

CHAPTER 7

STUDENTS' IMPRESSIONS OF THE PROJECT

Despite the successes of student participants, there were definitely times of frustration during the project that escalated to the point that some students quit engaging with their projects or refused do the post assessment. As a result, I wanted to get a sense for how students thought about or perceived aspects of the DigiblePets experience, an experience far different from customary school. I did this in two ways. First, I gave students a brief pre and post survey that I created about their feelings about making mistakes in relation to learning and perceptions of computer programming. Second, I asked questions specific to students' views of their experiences during the project and about computing in general in the interviews I conducted after the project was over. In this chapter I report on some of the information from these data sources to communicate how students felt about the DigiblePets experience after the project had ended. I break the results into three parts. One part is how students responded to questions about mistakes. A second part is how students responded to questions about computing. The third part is a reporting students' impressions of the unit as a new learning experience. The reason I highlight these three ideas is because the activities of the project were so different than what these students normally experience in school and involved technologies that were so atypical from what these students had previously used that I wanted to get a sense for what students' would say about the unit and different aspects of the unit after the project's completion.

Ideas About Making Mistakes

Being exposed to an open-ended, independent learning environment was new to these students in particular who were used to spending their days filling out workbook pages and having them marked correct or incorrect. As described in Chapter 3 (the learning activity design), and Chapter 6, (debugging), making mistakes, and subsequently finding and fixing them, was a large component of this project. To get a sense for what students thought about making mistakes in the context of the project, I asked students to give impressions about mistake making statements in the pre and post surveys and also respond to mistake-related questions during our interviews after the project had ended. From these responses, I tried to get an initial impression of students' views about mistakes related to working on the project, knowing that this was a first iteration and fairly complex project to implement

On the survey, students responded on a scale from 0 - 10 (where 0 was "don't agree at all" and 10 was "completely agree") to a series of statements. Given the exploratory nature of how students feel about mistake making, I created three items to help me see if there was any change in student perceptions. The mistake related statements were, "I am confident I can fix a bug/error when my computer code isn’t working right," "I learn best when I have to figure out my mistakes" and "I like figuring out how to fix my mistakes." Survey numbers based on these statements can be found in Table 16, Table 17, and Table 18. For several students, the numbers seem affirming. All five students reported feeling more confident fixing errors in computer code after the project. This makes sense because none of the students had any experience computer

programming before the project. Three of the five felt fairly confident about being able to debug their computer programs after the project (reporting either an 8, 9 or 10 score). Four out of five students, in the case of learning best when figuring out mistakes and liking to figure out mistakes, reported positive increases in their responses to the statements.

Table 16

Students' Survey Responses for Statement: "I Am Confident I Can Fix a Bug/Error When My Computer Code Isn't Working Right"

Student	Pre-Survey	Post-Survey	Change
Rocky	not answered	n/a[31]	n/a
Jamal	1	5	+4
Tabitha	0	9	+9
Carlos	1	10	+9
Dino	0	2	+2
Tegan	0	8	+8

[31] Rocky refused to complete the post survey.

Table 17

Students' Survey Responses to the Statement: "I Lean Best When I Have to Figure Out My Mistakes"

Student	Pre-Survey	Post-Survey	Change
Rocky	10	n/a[32]	n/a
Jamal	1	5	+4
Tabitha	6	5	-1
Carlos	5	10	+5
Dino	1	3	+2
Tegan	5	8	+3

In the interview, I asked a series of questions asking students to reflect upon their experiences making mistakes during the project. The interviews had a defined protocol, but were intended to be somewhat open-ended as to accommodate specific follow-up questioning based on what the students said. In all cases, the following two questions were asked:

1. Do you feel differently about making mistakes than you did before the project?
2. Did making mistakes or errors in this project make you feel dumb?

[32] Rocky refused to complete the post survey.

Table 18

Students' Survey Responses to the Statement: "I Like Figuring Out How to Fix My Mistakes"

Student	Pre-Survey	Post-Survey	Change
Rocky	10	n/a	n/a
Jamal	1	4	+3
Tabitha	1	4	+3
Carlos	7	10	+3
Dino	5	3	-2
Tegan	6	8	+2

When asked whether they felt differently about making mistakes as a result of the project (see Table 19 for details), four students said no and two students said yes, in some way. Both Jamal and Tabitha alluded to some aspect of the idea that being able to fix your mistakes was in some way productive. Tabitha reported that being allowed to fix your mistakes was different from other domains. Jamal reported that fixing mistakes during the project might have helped his thinking skills in general. The remaining four students interviewed did not share the same sentiment, mostly answering in one word that their feelings about making mistakes did not change. The only student to elaborate was Rocky who reported that learning from your mistakes continued to be his view.

Table 19

Students' Interview Responses to the Mistake-Related Interview Questions

	Do you feel differently about making mistakes than you did before the project?	**Did making mistakes or errors in this project make you feel dumb?**
Tegan	No.	NnnMmm.
Rocky	Not really... **Learn from your mistakes.** The past is the past and move on pretty much. That's pretty much how I still feel.	MmmHmm. **Just cause the stupidest stuff that you just couldn't see** and then you show you and you're like, "yup".
Dino	NnnMmm.	Not really. Well, kind of because I hate computers because I never understand them, so. Like when I am around computers **I feel dumb because like you never get them to work the way you want them to** and if you do something, something else comes out wrong and you can't fix it. It's just it's like.
Carlos	No.	No. Because **I know I'm smart.**
Tabitha	**Yeah cause you have more opportunity to try to fix them on here (*the computer*) than in other things I guess**.	**Kind of because everyone else was getting it**. Like people who don't care about school were getting it.
Jamal	Well maybe a little. It didn't have a big impact. But maybe a little, I guess it would, **it might help maybe I guess like cognitive thinking skills** or something. **Being able to think back and figure out what step you went wrong** and give you a better idea of how to fix it.	No not really. Just cause like, **no not really cause I could like fix them and stuff**. And they were just like little mistakes, so.

Only two of six students reported in the interview that they felt that making mistakes in programming had to do with intelligence or academic success, suggesting that making mistakes in programming made them feel less intelligent. For example, Jamal did not feel less intelligent as a result of making mistakes during the project because he was able to fix the mistakes he made. The idea of being encouraged to fix mistakes without repercussion was a fundamental part of the project that I hoped students would come away with. Jamal's sentiment parallels his views about mistakes being somehow productive from the first question.

However, Tabitha, stated that making mistakes in the project made her feel less intelligent because she felt that she was not able to understand when everyone else gave the appearance that they were able to.

Facilitator	Did making mistakes during this project make you feel dumb?
Tabitha	Kind of because everyone else was getting it. Like people who don't care about school were getting it.
Facilitator	Do you feel like you are someone who cares about school so you should get it more easily?
Tabitha	Well yeah I care a lot more now so, yeah I guess.
Facilitator	Did you feel like you were going to be graded or judged based on what you would get done?
Tabitha	Kind of. Cause I thought that like there was going to be a bunch of people at the last show thing.
Facilitator	Yeah me too.

Tabitha	So I didn't want to bring something... Like ours looked just stupid. Everyone else's was way good.
Facilitator	Did you feel like someone would be like "oh you get a D" or something? Or did you feel like it would be more personal?
Tabitha	Well, like everyone is judgmental so I know they'd just be thinking like "what is that?"
Facilitator	Ok. And you didn't want people to think you were just goofing off during your class time.
Tabitha	Yeah.

In the excerpt Tabitha stated that she felt caring about school should somehow have been reflected in how successful she was at the project. Also, Tabitha reported that other people's potential judgments about her project and whether she had taken it seriously made her feel badly. Recall that Tabitha had become insurmountably frustrated with debugging her code and that her programming code had been inadvertently erased during workshop 11. Subsequently she struggled to finish her project. Despite this setback, Tabitha attended the design exhibit and displayed what Steph had managed to replicate of their work. Steph, her partner, did not attend due to a conflict with a scheduled counseling session. For Tabitha, the public component of the project seemed to make her feel uneasy because she wanted those who saw her project to understand that she cared about school more than she used to.

In summary, these students generally reported feeling more positively towards learning by fixing mistakes after the project. Students also reported feeling more confident about fixing errors in programming code after the project. This suggests that

elements of the activity design and experience with the project may have supported these students in becoming more positively inclined towards mistake making and more confident in debugging. A small number of students reported feeling differently about making mistakes after the project, feeling that mistakes were in some way more productive than previously. Many students did not report feeling differently about mistakes at all. The term "mistake" can take many contexts. The reporting suggests that after the project some students may not have felt differently about making mistakes in general, as Rocky said, "Learn from your mistakes. The past is the past." Also, it is possible students may have associated the interview question about making mistakes with making mistakes in life rather than school, as Jamal talked at great length in his interview about how he was raised by his parents to fix his own mistakes. That is not incompatible with how students felt when asked to respond directly to specific statements about learning by fixing mistakes in the surveys. From the survey reports, students answered after the project that they did feel like they learn better when given the opportunity to fix their mistakes and they more positively associate learning by fixing their mistakes.

Ideas About Computing

During interviews, I asked students directly, "Did the project change how you feel about computers?" After all, one of the goals of the project was to broaden participation in computing through an experience creating a personal project with an innovative hybrid technology. If the students gained a new appreciation for computing, then perhaps they would consider participating in academic or personal activities involving computer

programming again. On the contrary, if the experience with the project worked to strengthen or bring about new negative feelings about computing technologies and computer programming then continued broadened participation by these students would probably not occur. Knowing what students felt about computing after their involvement with the project could provide insight into students' future participation in computing. Students' reports were varied (see Table 20).

Four of six students, Tegan, Jamal, Carlos, and Tabitha, reported feeling positively or more positively toward computing than they had originally mainly because of the way the project opened them up to new opportunities and ideas they had not realized existed. One student, Jamal, even stated that he was now thinking about pursuing a career involving computing. And finally, two of six students, Dino and Rocky, reported disliking computers more than previously. In the excerpt below, Rocky stated that the project furthered his negative feelings because he believed computer programming would be easy given prior experience with the Internet and instead it was difficult.

> Rocky (I thought) That they were easy...Because the stuff I've done on them, just the internet and stuff is easy. But, the programs that they actually make, isn't...Well, I just don't like. I don't want to ever program anything again...Cause it just wasn't my thing. I didn't like it.

In general, most students reported that they feel either positively or more positively than previously towards computing after their experience with the DigiblePets Project. Some students reported that the experience with computing broadened their ideas in a positive way about what people could accomplish with computers.

Table 20

Students' Interview Responses to Feelings About Computers

	Did the project change how you feel about computers?
Tegan	"**Yeah. I didn't know it was all like.** Well, I knew but I never did anything like that. Just internet and stuff."
Rocky	"(I thought) That they were easy...Because the stuff I've done on them, just the internet and stuff is easy. But, the programs that they actually make, isn't...Well, I just don't like. **I don't want to ever program anything again**...Cause it just wasn't my thing. **I didn't like it**."
Dino	"I hate computers...No. **It made it worse**."
Carlos	"No it didn't change it. **I still like them**."
Tabitha	"Kind of. I mean this is one program and **you can do a lot more**. I don't know. Tegan, you know who that is? Her little like monkey thing. I don't know. **I think it's pretty cool that you can like do this and it's just a program and codes and stuff**."
Jamal	"A little bit yeah. **I thought about going into a field with computers**. I hadn't really ever thought about it before but after this, I'm thinking about it."

Overall Impressions of the Project

The final component involved the students' overall impression of the project. In general, the project was intended to help students learn about computing and the design process in a fun and interesting way. Many students expressed satisfaction throughout the

project, for example cheering, but I elicited this information through the final interviews. In the interview I asked all students "What was it like working on this project?" to get a sense for their overall impressions. When answering the question, many students expressed positive feelings but the enjoyment was sometimes augmented by frustration. Four of six students, Jamal, Tegan, Tabitha, and Carlos had positive things to say about the project, that it was fun, unique and interesting, even if sometimes frustrating (see Table 21).

In the following excerpt Carlos stated how he felt about the project overall.

Facilitator	What was it like working on this project?
Carlos	It was fun.
Facilitator	What was fun about it?
Carlos	I learned how to program it and I got to mess around with the computer.
Facilitator	Did you feel like that the whole time or did it change throughout the project?
Carlos	Well in the middle I was like frustrated cause I couldn't do some stuff but, yeah, I still liked it at the end.

Dino, Tabitha, who had both positive and negative comments, and Rocky all mentioned that participating in the project was at times frustrating and difficult. This result suggested that at least two students, Dino and Rocky, were driven away from hybrid design technology projects and the computer programming and design work involved with them. However, all students, excepting Dino who was not asked, reported

they would probably or definitely sign up for the project again given the chance. Even those students who never wanted to program again or found the project frustrating would choose programming in the context of designing a personally meaningful project over what was normally offered in school.

Conclusion

The sentiments of the students who participated appeared to be mixed. In general, most students did not report changing their ideas about making mistakes and learning. Yet, many students did report they felt more positively towards learning by fixing their mistakes and more confident about their ability to debug computer programs. Many students gained a greater appreciation of computers and computer programming, suggesting that many of these students may consider themselves part of a larger, and now indeed broader, computational community. But other students did not. Some students enjoyed their experience in the project; others found it difficult and frustrating. Despite this, all but one student reported they would participate in the project again if the opportunity arose.

Thus, while there was a mix of responses, I take the comments from students to be positive with some important design implications. The project was difficult for students and included times of frustration and confusion. Many things could have been done differently to make the negative experiences more positive in both aspects, making and fixing mistakes and computing in general.

Table 21

Students' Interview Responses to How They Felt About the Project Overall

Student	**What was the Project like?**
Tegan	"**Interesting. Different**. I've never done something like that before. I don't know. It's **frustrating** sometimes."
Rocky	"Um, **I just didn't like having to debug everything.** I don't like computers anymore. They're harder than I want them to be. So."
Dino	"Yeah **it was hard because I didn't get it.** But I didn't want to try to change it because if I ruined something it would just make it worse, you know? Then I wouldn't be able to fix it. "
Carlos	"**It was fun**. I learned how to program it and I got to mess around with the computer. Well in the middle I was like **frustrated** cause I couldn't do some stuff but, yeah, I still liked it at the end."
Tabitha	"I've never done something like this...Well, **it was cool** because we don't really do that many hands on projects. That was really cool...Um what's the word. I don't know. **I got really frustrated.** Just cause **I couldn't figure it out** and my partner is gone a lot. So. It was hard for me to do on my own."
Jamal	"**It was fun**. Like when I first started making my own I didn't really know where to start so that's why I kind of didn't do anything for a while. But then **once I figured out what to do and everything, it came together**. "

For example, designing activities to more saliently center on the productivity of fixing mistakes in learning may have supported students in becoming even more positively inclined towards finding and fixing mistakes in their work. Similarly, limiting

the amount of frustration students dealt with during their projects, through more concrete, strategic supports, could potentially have helped students have a more positive experience overall. Both of these changes could also have an effect on students' confidence in fixing programming errors and in students' enjoyment of and satisfaction with the project as a whole.

CHAPTER 8

CONCLUSION

This dissertation began with the hypothesis that students from an alternative high school could complete and even benefit from an open-ended learning experience with an innovative hybrid design technology. I designed the DigiblePets project and implemented it at Winder Alternative High School with nine students. For five weeks, students designed, developed, and crafted their own interactive pets by creating a tangible body out of physical materials embedded with a microprocessor and programming a corresponding virtual program.

The primary goal of this project was to engage students in computing by encouraging those who may not normally relate to computing or have had limited access to computing to engage in a hybrid design project. Hybrid design technologies were thought to be well suited to this goal because they combine elements of known interests and hobbies with computer programming (Eisenberg, 2003) and include a physical component that can be both engaging and familiar (Eisenberg, et al. 2002). The DigiblePets project was also designed around an interactive pet, much like popular children's toys, thus known to and liked by students. In addition, a goal of this research project was to use debugging and code modification/reuse as strategies within a Constructionist-influenced learning environment as a way for novice programmers to learn aspects of computer programming. I also used other relevant literature on designing learning environments with technology to frame a set of commitments for the design of

the project. In keeping with the design research tradition, an intent of this research was not only to gain practical understanding of how young people think and learn with hybrid technologies but also to further develop theoretical ideas in the areas of hybrid design technologies, computer science education, and the learning sciences.

Overall, students were successfully in creating their own interactive pet projects with at least some functionality that differed in ways from the given prototype project. However, there were some areas for improvement within the project. For example, most students did not participate in all facets of media design. The implications of students' limited participation across disciplines are relevant to the Maker community as well as designers, educators and researchers of hybrid media inside and outside of classrooms. The implications are further discussed in this chapter.

Also, the Constructionism community touts the benefits of Constructionist learning environments. The learning environment for this project was designed to remain true to the spirit of Constructionism. In particular, the tenet of sharing, seen as a vital component of Constructionism, was integrated into the project as an important part of the learning/making process. However, how, when, and where sharing occurred during the project differed from intended and designed sharing experiences, which has potential to shift perceptions of how to view and support sharing as a part of learning. Later in this chapter I discuss the notion of sharing and how this research provides different insight into how sharing should be thought of.

Part of the focus of the final chapter, beyond making theoretical contributions is to make practical contributions that highlight potential areas for improving classroom-

based approaches to learning with hybrid design technologies and iterating upon the overall project design for additional implementations. An iterative approach is an important part of design research, allowing researchers and designers to discover and improve their learning implementations through execution, evaluation, and enhancement in real scenarios (Edelson, 2002). I intend to highlight areas where results and observations countered assumptions I had based on previous research and literature to create a baseline for other designers of learning environments with hybrid technologies. Thus, as a community, we can use previous design approaches such as this one to think about how learning with hybrid technologies can be improved. Also, another intent was to use students' results from this study to refine the DigiblePets Project for another implementation. I wanted to reflect on the design as a whole and evaluate potential opportunities to refine the project. In the remainder of this chapter, I discuss how the discovered limitations of the project may actually have important implications for Constructionist-inspired learning environments. I describe ways to improve the overall design of the project to tackle the issues observed.

In the final sections of this chapter, I synthesize how the outcomes of the project connect back to the original research questions. This provides a summary of the research. I also discuss limitations of the research.

Theoretical Contributions

Relative Notions of Sharing

Constructionism reports and highlights an intuitively sensible set of conditions

and requirements for learning to happen including sharing as an important part of the learning process. Sharing is not only part of Constructionist learning but also motivating for youth designers. Based on relevant literature, the theoretical assumption at the beginning of the project was that sharing would be motivational, especially with a group of students who do not often have a chance to be recognized in a positive way for their academic work. As a result, I designed several aspects of the activity structure to promote sharing with the community, for example daily round table discussions and the culminating design show event. The notion of sharing was consistent with Constructionism in some ways, for example, at certain times, students wanted to show off to friends, favored teachers and administrators. However, sharing most often occurred spontaneously with students dictating the parameters and players of the sharing. In observed cases where students shared on their own terms, students exhibited pride and excitement in their work. Contrary to expectation, students were oftentimes reluctant to share in designed and deliberately sanctioned ways. For example, Chapter 4 highlighted how round table discussions were often cursory, with students more interested in continuing work on their individual projects than contributing. Later in this chapter I also discuss how students were very reluctant to attend the culminating design show, where sharing was intended to be the impetus, focus and reward of the event. Of note is that with this population of students there were a lot of histories and structures in place that also discouraged sharing, for instance the sharing paradigm discussed in Chapter 5 when students were quick to claim ownership over ideas and discourage one another from helping others with ideas. A theoretical take away is that the notion of sharing a public

artifact has an overlooked relative dimension. With whom you want to share, when you share, and how you share are important issues to explore. Sharing has a vulnerability aspect and students chose with whom to share. For example, in Chapter 5, I observed that Steph chose to share her physical design with the school secretary, not the classroom teacher, me or another student. It may be that the models of great sharing are not only supportive but they are also consistent with a broader history students bring with them to the learning environment. Sharing is conditional, situational, and sensitive to the sharer. In the Constructionist literature, Samba schools exemplify the sharing ideal (Papert, 1980). Samba school participants know all about carnival, they know what's desired, they know that it's not a proprietary environment or space, they know there is some sense of collective ownership. Proponents of Constructionism should think about the notion and nuances of sharing in learning spaces and explore how these notions and nuances can potentially contribute to the learning process.

The Hybridity Continuum

One reason hybrid design media are exciting and garnering interest is because of their potential to engage young people in multiple disciplines of design, for example computer programming and crafting. This is especially important because these media are designed to appeal to young people who may not otherwise be interested in or willing to attempt a discipline like computer programming. Researchers and designers of hybrid design media intend for the technologies to act as a bridge between the known, familiar interests and computing or vice versa in ways that are relatable, natural, and motivating (Eisenberg, 2003). A theoretical assumption underlying this project was that students

would naturally navigate this bridge between tangible and virtual, craft design and computer programming, taking part in exploring both facets of design. I hypothesized students would become engrossed in an area of interest, like crafting, and through the development of their physical pets would feel compelled to participate in bringing their efforts to life by programming the pet to interact. Similarly, I believed students who joined the project because of an interest in computer programming would eventually desire to help bring their virtual creations into the physical world by crafting a physical pet. Rather than adding credence to these ideas, in the project, students strictly divided roles, demonstrating a desire to pull apart and isolate the computational and physical elements of the multi-modal design project (see Chapter 5). Although students did connect to computing in different ways and many students who might not have otherwise participated in computing were compelled to participate in the project, students did not always participate in all aspects of the project. Rather than providing multiple means of entry into different disciplines of design, in most observed cases, the fact that there were multiple modalities to the design of students' pets allowed for division of labor, which segregated and solidified instead of integrating the different elements. For example, as discussed in Chapter 4, Tegan was drawn away from computing by a compulsion to take ownership over the physical pet design. Also in Chapter 4, Carlos was never persuaded to participate in crafting the physical pet or helping develop how users would interact with the alien that he had essentially created and developed independently in the virtual space. Carlos was in charge of the computer programming, but stopped participating in the design process when the group shifted focus to their physical pet, maintaining that he was

not artistic through to the end of the project. Steph and Tabitha relegated themselves to their own design spaces, physical and virtual, and only when Tabitha gave up on the project did Steph attempt to computer program with disastrous effect (Steph deleted all the girls' code as discussed in Chapter 5).

These examples are disconcerting because they contradict assumptions made by the hybrid media community. The observations from this project provide evidence that even when the intent of both the designer of the media technology and the designer of the activity structure of a project using hybrid media technology are to integrate multiple disparate modalities into one design project, like art and computer programming, hybrid media can be and may likely be dichotomized by students. The desired fusion of tangible and digital media in an innovative way was not enough to incite all students to participate in multiple domains. Hybridity is complex and has ramifications for the opportunity for developing new interests. Aspects of this complexity became visible when collaboration broke down and students strictly divided labor. This result is particularly pertinent to the burgeoning idea of integrating hybrid deign media into classroom learning environments where exposing students to multiple disciplines and supporting students in developing new interests might well be essential outcomes, critical to key learning goals.

Students' segregation of the design media in this study suggests the need for developing models that consider hybridity as part of a continuum that encompasses the technologies and the structures of the learning environment. For example, as discussed in Chapter 5, group design projects have the potential to be collaborative but are not necessarily so, even when the learning environment, like in this project, was designed to

foster shared ideas and effort. In an effort to guide collaboration and learning, some designers dictate exactly how students should interact, for example the FCL jigsaw model provides students with specific tasks to master and then bring back to their groups (Brown & Campione, 1996). However, supports that would make more stringent mandates on how students work together on complex tasks have both inherent costs and benefits. The costs include counteracting the open-ended, exploratory learning environment designers and proponents of these media strive for. Having the ability to organically renegotiate and reconsider ones role in a collaborative learning effort has been shown to be important for students' development of new ideas and success in complex problem solving activities (White & Pea, 2011). Providing instances where students have the opportunity and motivation to reestablish their roles may be the type of support that could broaden students' participation in multiple design domains. The results of this study provide a word of caution to educators and researchers excited about the potential of tangible/digital design media. How youth engage in design projects with these media may not always align with the goals and notions of designers. Rather than assuming that hybrid media are promising simply because they offer a way to bridge disciplines, we need to further research effects of these media on learning and interest development and use the findings to develop a framework for understanding the hybridity of multi-modal design media as a function of the technologies and how they are integrated into learning environments.

Reflections on the Results

In this section I summarize my findings, linking them back to the original research questions. In doing so, I describe what was learned during the project, and how that affects research and design of hybrid design technologies in education.

The first result of the research is that this population can effectively complete a hybrid design technology project given the constraints associated with their experiences, histories and school. This addresses research question 1. Can these students successfully complete a hybrid technology design project? The students were willing, equipped and capable of completing a complex academic task in a Constructionist-inspired environment, with the designed supports. Students used the experience of debugging prototype code to use programming concepts and structures. Then, students adapted what they had learned to create their own computer programming projects by reusing, modifying, and writing original code. Students constructed physical pets from craft, art, and found materials and then programmed the pets to interact via embedded microprocessing boards. The pet projects were imaginative, functional, and unique. Students continued to attend class despite absences. They participated in all aspects of the designed activities, and showed signs of sustained engagement. On occasion students arrived early and others stayed late to work on their projects, such as Jamal described in Chapter 4. Despite the challenges associated with an extended design and programming project, these students did not give up. Recall that the staff at Winder were supportive about the project overall but believed perseverance and with it productive and prolonged participation would be one of the biggest challenges. I consider the completion of the

project, as represented by several students in Chapter 4, to be an important story about real possibilities for students to do meaningful work.

Second, students participated in aspects of programming, debugging, crafting, designing interactions, and design thinking, although all students did not participate in all of those areas. This addresses research question 2. How do alternative high school students engage in the environment? What is the nature of their participation? When provided a more structured environment, students took part in collaborative problem solving, as described in Chapter 5. With less structure, students worked in a cooperative rather than collaborative way by distributing tasks, roles, and goals during the project. Without the explicit structuring of the task, groups still completed the work. However, the groups functioned cooperatively – they each worked on separate pieces important to the resultant whole – rather than collaborative – in which they would have talked with and engaged one another with ideas and suggestions based on what other group members had contributed immediately before. All students enrolled in the project succeeded in earning course credit for their efforts.

Third, I address research question 3. In what ways do students engage with bugs? What is the nature of students' debugging strategies? Students willingly participated in debugging activities. Students employed many strategies for facing bugs including tinkering and were most often able to resolve most of the bugs in their projects. For instance, tinkering represented 57% of Jamal's debugging activities and 40% of Tegan, Rocky, and Ted's debugging activities. Students resolved 80% of the bugs they encountered in their independent projects. In many cases students identified and fixed

bugs without or with minimal support. Some students engaged in fixing a single bug for long stretches of time. Given their experiences debugging, the majority of students were then successfully able to debug parts of a new program after the project was completed. Some students were able to debug all parts of the program given. This too is a very encouraging result.

Fourth, there were some ways in which it appeared some of the students felt empowered by this project. This addresses research question 4. Do students exhibit elements of empowerment with respect to computing? More specifically, how do students think about their experience with the project, especially with respect to their making mistakes and their feelings toward participating in computing in general? On the whole, the majority of students reported more positively associating learning by making and fixing mistakes after the project. All students reported they felt more confident, with some feeling very confident, with finding and fixing errors in computer code after the project. Many of the students reported that they felt more positively about computing because of their experience. Some students mentioned having learned more about the possibilities of computing. Many students reflected positively on their experience, but there was definitely frustration at times in the project. Nearly all students reported they would participate in the project again if given the opportunity.[33] One student, Jamal, reported interest in pursuing a career in computing as a result of his experience with the DigiblePets project.

---

[33] Recall Dino was inadvertently not asked the question.

Finally, because of the target population, this research project had a set of unique design challenges including questionable content appropriateness, absenteeism, and wariness towards one another. It was far from Papert's (1980) model of learning in a collaborative, harmonious samba school, in which groups of people with different levels of expertise work together to create something meaningful. Yet, some important ideas from Constructionism held true. Students were engrossed for a long stretch of time in creating projects that seemed to have personal meaning, even when that meaning was not sanctioned in school. Students successfully worked in an open-ended, independent, entirely new kind of learning environment. Students explored aspects of computer programming and programmatic thinking.

Limitations and Steps for Improvement

First, there are a series of questions that could inform the project that deserve to be addressed. For instance, I noted in Chapter 6 that there was a lull in programming effort when the craft materials arrived during workshop 5. Should there have been different kinds of supports or structure to avoid this type of lull or is this to be expected when a new set of materials arrives? Students debugged and resolved many bugs, but many times needed facilitator support in doing so. Were there things that could have or should have been done differently to get better debugging? In Chapter 5, I discussed the way students segregated work. Did the separation affect the quality of what the students could be accomplished, or was it an appropriate way for the students to proceed given the newness of the entire project to them? These questions are substantial and could all be

part of a larger discussion on using hybrid technologies for learning in similar learning environments. Many more iterations of the project with different populations in different settings would have to be undertaken before headway could be made regarding them. My goals for this project were more modest. At the most fundamental level, I was trying to design and implement an intervention with a set of underlying theoretical assumptions and characterize the nature of what happened. Yet I still recognize there are important ways the implemented project could be refined to increase student engagement and potential for learning.

As acknowledged previously, the design and implementation of the DigiblePets project was a single iteration. Design research is most often considered several cycles of design, implementation, analysis, and refinement (Edelson, 2002). However for the purposes of this study, only one instantiation of the project cycle was completed. Given the results and what I have learned, I hope to enact further implementations in the future, taking into consideration that I intend to continue my work in this domain.

Also, this research was conducted with a small population. Because I wanted to realize my research in an authentic classroom environment with struggling students, I did not have the ability to choose who and how many students participated. The small sample size was an unintended consequence of the setting, constraints, and demands of the school and students within it. Perhaps it would be possible to recruit more students at a larger alternative school or even by implementing a second iteration at Winder where word of mouth from other students may encourage more students to sign up.

As with any research project, there are things that for any number of reasons could have been done differently. Every aspect of the design and implementation of the new kind of technology and the activities of the project were carefully thought through based on prior research and grounded in literature. However, had aspects of the design or research strategy been modified, the outcomes could have been made more salient. Recall that this instantiation was meant to be a first endeavor at a project of this kind. This is why design research can be seen as so important. Different groups differ in their needs and interests and design research has the potential to keep us accountable to those differences and accountable to developing theories and approaches that answer to these changing variables.

During the DigiblePets project, I oftentimes had to make modifications and in-the-moment decisions to meet student needs and react to unanticipated events. For instance, I was not prepared to deal with a student being taken to jail in handcuffs like Ted was, or another student confiding in me that he had never done anything important academically before receiving a paper certificate for passing a class, like Jamal. I witnessed students struggle with frustration and tried to use my background as a teacher to continue to push them to develop important skills to deal with the errors and problems they were encountering instead of telling them the answer. I sometimes was faced with problems I did not know the answer to, for example Maya and Dino encountered a programming bug during workshop 12 that I had to honestly explain I could not identify. Despite this I tried to help the group the best that I could. I also had to put honest effort into simply trying some things out that ultimately were not successful, like the

culminating event and having students share and improve on one another's work. I hoped the project would begin to change how students viewed learning and computing, but realistically, five weeks was not enough time, in most cases to achieve that change.

Even when students worked in deeply committed ways, situations arose pinpointing limitations in the overall project realization. The pet theme was successful at promoting initial student involvement, for example, recall from Chapter 4, Tegan reported in an interview that she chose to sign up for the project specifically because we were making pets and not just computer projects. The pet theme also provided some students an affective means to relate to their projects, for example Tegan and her monkey and Jamal with his zebra and accessories. However, the overarching design goal of the pet project was not well defined nor well inculcated for the students. The intended reasons for designing pets include: to create an interesting, integrated, sensible, interactive toy for young children and to show the product to and get reactions from a group of outside individuals. Students were informed of the intended audience, but the audience was not well explored or explained to students nor, as a result, internalized by students. Students needed more intimate and specific knowledge of their audience in order to make their projects appropriate for and interesting to the intended users.

Resulting student projects were interesting and unique but lacked purpose or overall integration and were not always audience appropriate. Students' projects were oftentimes inappropriate for a general audience. For example all student projects, at one point or another in the design process, made use of vulgar language, gestures and/or illegal activity. Students delighted in the fact that they could make their pets say curse

words or derogatory statements by either speaking out loud or in a speech bubble. Students made their pets "party" with alcohol and illicit drugs. As a result, students lost sight of the purpose for making the pets in that they were not designing for the intended audience of young children.

The evening design show at the end of the project was intended to be the culminating activity where students demonstrated their pets to friends, family, invited school personnel and community members. However, the design show was poorly attended by both students and invited guests. Of the nine students in the project, only four were cajoled into attending, even when attendance was compulsory for course credit. Students reported that course credit was a major motivating factor in their participation in the first place.[34] Although disappointing, the result is consistent with what one might anticipate. Most of these students have limited, if any, relationships with their parents, teachers or any other adults. The students have little academic identity, making it awkward at best to encourage friends to attend. They have many binding outside of school commitments that trump academic involvement atypical of students from a conventional high school. Expecting the students to relish a consequential task that did not mesh with their circumstances and conflicted with their outside of school lives did not make sense.

One final thought on the purpose of the Digible Pet projects must include the limited overall sensibility and lack of integration of student designs. Students created lots

---

[34] After the poor showing at the event, Anna, the classroom teacher, allowed those students with legitimate excuses write an essay about their experience with the project in order to earn course credit.

of unique functionality for their pets but did not really encompass the overall idea behind creating an interactive pet, versus an interactive object. The eventual responses to button presses and sensor inputs were not consistent with what one might expect of a pet. Students did not see the pet theme as a driver for innovative, imaginative but related ideas having to do with real pets. Students were not concerned with their pets being virtual companions that need care like water, food, exercise, and a place to sleep or play. They also did not see user input as a way for individuals to interact with their pets as beings. My hope, as mentioned in Chapter 5, was that students would have intimate knowledge of interactive pets, like ZhuZhu pets and Webkins. Their knowledge of or interest in these toys was either limited or not accessed. Therefore, in future project iterations, so long as the pet metaphor continues, more attention must be paid to familiarizing students with what interactive pets actually are and why they are popular. Giving students an opportunity to access and increase their domain expertise before they design their own pets could promote student projects to be much more integrated, pet-like and improved overall. Studies of computational crafts show similar findings about the benefits of allowing students to experiment and get to know how the media can function before embarking on a long-term product (Buechley et al., 2007). This would also have encouraged students to think about their projects conceptually before embarking on a host of functionality changes that may not have been well integrated. In addition, attending to the issues above would help students be more focused by providing an incentive for purposeful, directed design decisions and allow for better DigiblePet Projects overall.

Next Steps and Final Thoughts

On a broad level, this project presses on various theoretical constructs, impelling researchers and designers in computer science education, learning sciences, and hybrid design media to further explore issues related to debugging as a motivational learning tool, the nuances of sharing in learning, and implications of prospective hybridity of tangible/digital media. On a more local scale, possible future iterations of this project are exciting as there are many ways new opportunities could be explored based on the results of this first project. In this section I highlight a few potential next steps regarding future research projects. For example, in a next iteration of this project students could work on individual projects, but more time could be dedicated to exploring the realms of interactive pets and audience/users and include much more discussion of ideas together as a group. This could potentially encourage students to participate in more aspects of the project, as they would be responsible for developing every part of their pet. I would like to investigate the differences in students' interactions when working on individual projects in a supportive learning community. Also, the original project participants could be included in a new instantiation to work alongside novices as mentors and "expert pet developers". The experts could potentially help foster more peer interaction and peer learning as well as limit frustration the novice students reported. I would like to examine whether students interact more in this type of environment and whether feelings of proprietariness and wariness are as abundant. Likewise, the next iteration could be necessarily lengthened and include more students. Even if the workshop times could not be extended, the overall duration of the project could be increased to give students more

time to try out ideas, plan their projects, and become experts in all types of interactive pets. I would like to explore the types of projects students could develop given more supports, more peers, more time, and more experience. A combination of these ideas could work to improve the overall quality of the projects students create as well as in some ways limit frustration, increase the productive exchange of ideas, and provide more direction for students' pets. By attempting to extend the project in some these ways, I would hope to provide a richer environment for learning and more influential experience with computing overall.

In summary, the DigiblePets project successfully compelled a group of struggling students to engage in computing, crafting, and interactive toy design. It is indeed possible, and I hope to have shown also fruitful, to work with and explore processes of technology design and engagement with students who have been formally moved out of mainstream educational systems. For these students, the project was a radical departure from school learning and the students persevered within the new learning environment. The project gave students an opportunity to feel a sense of pride in their work and for many, to enjoy learning in school, perhaps for the first time in a long time. Like Tegan said to the principal of Winder during the final workshop while holding up her monkey and smiling, "I did it, all by myself!"

REFERENCES

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

Abrahamson, D., & Wilensky, U. 2005. *The stratified learning zone: Examination of the pros and woes of collaborative-learning design in demographically-diverse mathematics classrooms.* Paper presented at the annual meeting of the American Educational Research Association. Montreal, Canada.

Ackerman, E. (1996). Perspective-taking and object construction: Two keys to learning. In Y. Kafai & M. Resnick (Eds.) *Constructionism in practice: Designing, thinking and learning in a digital world (pp.*25-35). Mahwah, NJ: Erlbaum.

Alper, M., Hourcade, J. P., & Gilutz, S. (2012). Adding reinforced corners: designing interactive technologies for children with disabilities. *Interactions*, *19*(6), 72-75.

Aron, L. (2003). Towards a typology of alternative education programs: A compilation of elements from the literature. Washington, DC: The Urban Institute.

Barrows, H. (1996). Problem-based learning in medicine and beyond: A brief overview. *New Directions for Teaching and Learning*, 68, 3-12.

Berland, L. (2011). Explaining variation in how classroom communities adapt the practice of scientific argumentation. *Journal of the Learning Sciences*, 20(4), 625-664.

Berland, M., & Lee, V. (2011). Collaborative strategic board games as a site for distributed computational thinking, *International Journal of Game-Based Learning*, *1*(2), 65-81.

Bielaczyc K. (2006). Designing social infrastructure: Critical issues in creating learning environments with technology. *Journal of the Learning Sciences*, 15(3), 301-329.

Blikstein, P. (2011). One Fabrication Lab per School: the FabLab@School project. Tedx Talk, Manhattan Beach, CA. http://tedxmanhattanbeach.com/past-events/conference-october-2011/speakers/paulo-blikstein-2/

Blikstein, P. (2013). Digital fabrication and 'making' in education: The democratization of invention. In J. Walter-Herrmann & C. Büching (Eds.), FabLabs: Of machines, makers and inventors. Bielefeld: Transcript Publishers.

Brown, A. (1992). Design experiments: Theoretical and methodological challenges in creating complex interventions. *Journal of the Learning Sciences*, 2, 141-178.

Brown, A., & Campione, J. (1996). Psychological theory and the design of innovative learning environments: On procedures, principles, and systems. In L. Schauble & R. Glaser (Eds.), Innovations in learning: New environments for education (pp. 289-325). Mahwah, NJ:Erlbaum.

Brown, J., Collins, A., & Duguid, P. (1989). Situated cognition and the culture of learning. *Educational Researcher*, *18*(1), 32-42.

Bruckman, A. (1998). Community Support for Constructionist Learning. *The Journal of Collaborative Computing*, 7, 47-86.

Bruckman, A. (2000) Situated support for learning: Storm's weekend with Rachel. *Journal of the Learning Sciences*, 9(3), 329-372.

Buechley, L., Eisenberg, M., Catchen, J., & Crockett, A. (2008). The LilyPad Arduino: using computational textiles to investigate engagement, aesthetics, and diversity in computer science education. In Proceedings of the Twenty-Sixth Annual SIGCHI Conference on Human Factors in Computing Systems (pp. 423-432). Florence, Italy, April 05 - 10, 2008. New York, NY: ACM.

Buechley, L., Eisenberg, M., & Elumeze, N. (2007). Towards a curriculum for electronic textiles in the high school classroom. *ACM SIGCSE Bulletin*, 39(3), 28-32.

Catrambone, R., Stasko, J., & Xiao, J. (2002). Anthropomorphic Agents as a UI Paradigm: Experimental Findings and a Framework for Research. Technical Report GIT-GVU-02-10. Georgia Institute of Technology, Atlanta.

Chmiel, R., & Loui, M. C. (2004). Debugging: from novice to expert. In *ACM SIGCSE Bulletin,* 36(1), 17-21.

Cobb, P., Confrey, J., diSessa, A., Lehrer, R., & Schauble, L. (2003). Design experiments in educational research. *Educational Researcher,* 32(1), 9-13.

Collins, A. (1990). Toward a design science of education. In E. Scanlon & T. O'Shea (Eds.), *New directions in educational technology* (pp. 15-20). Berlin: Springer-Verlag.

Collins, A., Joseph, D., & Bielaczyc, K. (2004). Design research: Theoretical and methodological issues. *Journal of the Learning Sciences*, 13(1), 15-42.

Cooper, S., Dann, W., & Pausch, R. (2000). Alice: a 3-D tool for introductory programming concepts. *Journal of Computing Sciences in Colleges,* 15(5), 107-116.

Csikszentmihalyi, M., & Wolfe, R. (2001). New conceptions and research approaches tocreativity: Implications of a systems perspective for creativity in education. In K. A. Heller, F. J. Manks, R. Subotnik, & R. J. Sterberg (Eds.), *International handbook of giftedness and talent* (pp. 81–93). Oxford, UK: Elsevier.

Cunniff, N., Taylor, R. P., & Black, J. B. (1986). Does programming language affect the type of conceptual bugs in beginners' programs? A comparison of FPL and Pascal. *ACM SIGCHI Bulletin,* 17(4), 175-182.

Del Marie Rysavy, S., & Sales, G. C. (1991). Cooperative learning in computer-based instruction. *Educational Technology Research and Development*, *39*(2), 70-79.

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

DuMont, M., & Fields, D. (2013). Hybrid shmybrid: Using collaborative structure to understand the relationship between virtual and tangible elements of a computational craft. Madison, Wisconsin: International Society of the Learning Sciences.

DuMont, M., & Lee, V. R. (2012). A Technologically Enhanced Construction Kit as a Support for Children's Collaborative Computational Thinking. Paper presented at the 2012 annual meeting of the American Educational Research Association, Vancouver, BC.

Edelson, D. C. (2002). Design research: What we learn when we engage in design. *Journal of the Learning Sciences,* 11(1), 105-121.

Edelson, D., Pea, R., & Gomez, L. (1996). Constructivism in the collaboratory. In B.G. Wilson (Ed.), *Constructivist learning environments: Case studies in instructional design* (pp. 151-164). Englewood Cliffs, NJ: Educational Technology.

Eisenberg, M. (2003). Mindstuff: Educational technology beyond the computer. *Convergence*, 9(29), 29-53.

Eisenberg, M., Eisenberg, A., Gross, M., Kaowthumrong, K., Lee, N., & Lovett, W. (2002). Computationally enhanced construction kits for children: Prototype and principles. Proceedings of the International Conference of the Learning Sciences, (pp. 79-85).

Eisenberg, M., Elumeze, N., MacFerrin, M., & Buechley, L. (2009). Children's programming, reconsidered: Settings, stuff, and surfaces. Paper presented at the 8th annual conference on Interaction Design and Children. Como, Italy.

Evard, M. (1996). A community of designers: Learning through exchanging questions and answers. In Y. Kafai & M. Resnick (Eds.), *Constructionism in practice: Designing, thinking and learning in a digital world* (pp. 223-239). Mahwah, NJ: Erlbaum.

Falcão, T. P., & Price, S. (2010). Informing design for tangible interaction: a case for children with learning difficulties. In *Proceedings of the 9th International Conference on Interaction Design and Children* (pp. 190-193). Barcelona, Spain.

Fernández-Cárdenas, J. M. (2008). The situated aspect of creativity in communicative events: How do children design web pages together? *Thinking Skills and Creativity, 3*(3), 203-216.

Fields, D., Kafai, Y., & Searle, K. (2012). Functional aesthetics for learning: Creative tensions in youth e-textiles designs. In van Aalst, J., K. Thompson, M. Jacobson, & P. Reimann (Eds.), *The Future of Learning: Proceedings of the 10th International Conference of the Learning Sciences (ICLS 2012,* pp. 196-203). Sydney, Australia: International Society of the Learning Sciences.

Fields, D., Searle, K., Kafai, Y., & Min, H. (2012). Debuggems to assess student learning in e-textiles. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education* (pp. 699-699). Raleigh, NC.

Finn, J., & Rock, D. (1997). Academic success among students at risk for school failure. *Journal of Applied Psychology*, 82(2), 221-234.

Fredricks, J., Blumenfeld, P., & Paris, A. (2004). School engagement: Potential of the concept, state of the evidence. *Review of Educational Research*, 74(1), 59-109.

Furst, G. (2006). Prison-based animal programs a national survey. *The Prison Journal*, *86*(4), 407-430.

Griffin, J., Kaplan, E., & Burke, Q. (2012). Debug'ems and other deconstruction kits for STEM learning. *Integrated STEM Education Conference (ISEC), 2012 IEEE. Vol. 2*, 1-4.

Grover, S., & Pea, R. (2013). Computational thinking in K–12 A review of the state of the field. *Educational Researcher*, *42*(1), 38-43.

Guzdial, M. (2003). Programming environments for novices. *Computer Science Education Research*, *2004*, 127-154.

Harel, I., & Papert, S. (1991). Software design as a learning environment. In I. Harel & S. Papert (Eds.), *Constructionism: Research reports and essays* (pp. 41-84). Norwood, NJ: Ablex.

Huang, Y., & Eisenberg, M. (2011). Plushbot: an application for the design of programmable, interactive stuffed toys. In *Proceedings of the fifth international conference on Tangible, embedded, and embodied interaction* (pp. 257-260). Funchal, Portugal.

Hussain, S., Lindh, J., & Shukur, G. (2006). The effect of LEGO training on pupils' school performance in mathematics, problem solving ability and attitude: Swedish data. *Educational Technology & Society*, 9(3), 182-194.

Kafai, Y. (1996). Learning design by making games: Children's development of design strategies in the creation of a complex computational artifact. In Y. Kafai & M. Resnick (Eds.) *Constructionism in practice: Designing, thinking and learning in a digital world* (pp. 71-96). Mahwah, NJ: Erlbaum.

Kafai, Y., Fields, D., & Burke, Q. (2011). Collaborative agency in youth online creative production in Scratch. In *Proceedings of the 19th International Conference on Computers in Education, Chiang Mai, Thailand*.

Kafai, Y., Fields, D., & Searle, K. (2012). Making learning visible: Connecting crafts, circuitry & coding in e-textile designs. In J. van Aalst., K. Thompson, M. Jacobson, & P. Reimann, (Eds.), *The Future of Learning: Proceedings of the 10th International Conference of the Learning Sciences (ICLS 2012;* pp.188-195*)*. International Society of the Learning Sciences: Sydney, Australia.

Kafai, Y., & Harel, I. (1991). Learning through design and teaching: Exploring social and collaborative aspects of constructionism. In I. Harel & S. Papert (Eds.), *Constructionism: Research reports and essays* (pp. 85-110). Norwood, New Jersey: Ablex.

Kafai, Y., & Peppler, K. (2011). Youth, technology, and DIY: Developing participatory competencies in creative media production. In V. L. Gadsden, S. Wortham, & R. Lukose (Eds.), *Youth cultures, language and literacy*. *Review of Research in Education*, 34.

Kafai, Y., Peppler, K., & Chapman, R. (2009). *The computer clubhouse*. New York, NY: Teachers College Press.

Kafai, Y., Peppler, K., Chiu, G., Maloney, J., Rusk, N., & Resnick, M. (2009). From photoshop to programming. In Y. Kafai, K. Peppler, & R. Chapman (Eds.), *The computer clubhouse* (pp. 136-144). New York, NY: Teachers College Press.

Kafai, Y., & Resnick, M. (1996). *Constructionism in practice: Designing, thinking and learning in a digital world.* Mahwah, NJ: Erlbaum.

Kahn, K. (2004). Toontalk-steps towards ideal computer-based learning environments. In L. Steele, & M. Tokoro (Eds.), *A Learning Zone of One's Own: Sharing Representations and Flow in Collaborative Learning Environments* (pp. 253-270). IOS Press.

Kelleher, C., & Pausch, R. (2005). Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programming. *ACM Computing Surveys,* 37(2), 83 -137.

Klahr, D., & McCoy Carver, S. (1988). Cognitive objectives in a LOGO debugging curriculum: Instruction, learning and transfer. *Cognitive Psychology*, 20, 362-404.

Koschmann, T., Myers, A., Feltovich, P., & Barrows, H. (1993). Using technology to assist in realizing effective learning and instruction: A principled approach to the use of computers in collaborative learning. *Journal of Learning Sciences*, 3(3). 227-264.

Lampert, M. (2001). *Teaching problems and the problems in teaching* . New Haven, CT: Yale University Press.

Lange, C. & Sletten, S. (2002). *Alternative education: A brief history and research synthesis*. Project FORUM, National Association of State Directors of Special Education, Alexandria Virginia, February 1, 2002.

Lapidot, T., & Hazzan. (2005). Song debugging: Merging content and pedagogy in computer science education. SIGCSE Bulletin, 37(4), 79-83.

Lee, M. K., Kiesler, S., & Forlizzi, J. (2010). Receptionist or information kiosk: how do people talk with a robot?. In *Proceedings of the 2010 ACM conference on Computer supported cooperative work* (pp. 31-40). Savannah, GA.

Littlefield, J., Delclos, V., Bransford, J., Clayton, K., & Franks, J. (1989). Some prerequisites for teaching thinking: Methodological issues in the study of LOGO programming. *Cognition and Instruction*, 6(4), 331-366.

Lu, J., & Fletcher, G. (2009). Thinking about computational thinking. Chattanooga, TN. ACM. Academic Press.

Malan, D., & Leitner, H. (2007). Scratch for budding computer scientists. *ACM SIGCSE Bulletin* 39(1), 223-227.

Maloney, J., Burd, L., Kafai, Y., Rusk, N., Silverman, B., & Resnick, M. (2004). Scratch: A sneak preview. Second International Conference on Creating, Connecting, and Collaborating through Computing. Kyoto, Japan (pp. 104-109).

Marti, P., Pollini, A., Rullo, A., & Shibata, T. (2005). Engaging with artificial pets. In *Proceedings of the 2005 annual conference on European association of cognitive ergonomics* (pp. 99-106). Athens, Greece.

McCauley, R., Fitzgerald, S., Lewandowski, G., Murphy, L., Simon, B., Thomas, L., & Zander, C. (2008). Debugging: A review of the literature from an educational perspective. *Computer Science Education*, 18(2), 67-92.

McCrae, R. (1987). Creativity, divergent thinking, and openness to experience. *Journal of Personality and Social Psychology*, 52(6), 1258-1265.

Meyers, J., LaMarche, J., & Eisenberg, M. (2010). Craftopolis: Blending tangible, informal construction into virtual multiuser communities. Paper presented at the 9th Annual Conference on Interaction Design and Children. Barcelona, Spain.

Miles, M., & Huberman, A. (1994). *Qualitative data analysis* (2nd ed.). Thousand Oaks, CA: Sage.

Millner, A. (2009). Interface designs with hook-ups. In Y. Kafai, K. Peppler, and R. Chapman (Eds.) *The Computer Clubhouse: Constructionism and creativity in youth communities* (pp. 58-70). Teachers College Press. New York, NY.

Minsky, M. (1985). *Society of mind*. New York, NY: Simon and Schuster.

Monroy-Hernández, A., & Resnick, M. (2008). FEATURE Empowering kids to create and share programmable media. *interactions*, *15*(2), 50-53.

Murphy, L., Lewandowski, G., McCauley, R., Simon, B., Thomas, L., & Zander, C. (2008) Debugging: The good, the bad, and the quirky - a quantitative analysis of novice's strategies. SIGCSE Bulletin, 40(1), 163-167.

National Advisory Committee on Creative and Cultural Education. (1999). *All our futures: Creativity, culture and education*. *www.cypni.org.uk/downloads/**all**out**futures**.pdf*

National Research Council. (1999). *Being fluent with information technology*. Washington, DC: National Academy Press.

National Research Council. (2010). *Report on a workshop on the scope and nature of computational thinking*. Washington, DC: National Academy Press.

Norman, D. (1991). Cognitive artifacts. In J. M. Carroll (Ed.), Designing interaction: Psychology at the human-computer interface (pp. 17-38). New York, NY: Cambridge University Press.

Pang, L. S., & Foley, R. M. (2006). Alternative education programs: Program and student characteristics. *The High School Journal*, *89*(3), 10-21.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. New York, NY: Basic Books.

Papert, S. (1991). Situating constructionism. In I. Harel & S. Papert (Eds.), *Constructionism: Research reports and essays* (pp. 1-12). Norwood, NJ: Ablex.

Pea, R. (1983). Logo programming and problem solving. Paper presented at an American Educational Research Association Symposium (Montreal, Canada, April 1983) as 'Chameleon in the Classroom: Developing Roles for Computers'.

Pea, R. (1986). Language-independent conceptual "bugs" in novice programming. *Journal of Educational Computing Research*, 2(1), 25-36.

Pea, R., Kurland, D., & Hawkins, J. (1985). *Logo and the development of thinking skills*. in M. Chen & W. Paisley (Eds.), Children and microcomputers: Research on the newest medium (pp. 193-212). Beverly Hills, CA: Sage.

Peppler, K., & Kafai, Y. (2001). From SuperGoo to Scratch: Exploring Creative Media Production in Informal Learning. *Journal on Learning, Media, and Technology*, 32, 7, 149-166.

Raffle, H. (2006). Kinesthetic media: Touch, toys & interactive materials. In *ACM SIGGRAPH 2006 Educators program* (p. 8). Boston, MA: ACM.

Raffle, H., Parkes, A., & Ishii, H. (2004). Topobo: A constructive assembly system with kinetic memory. *Proceedings of the Conference on Human Factors in Computing Systems* (pp. 869-877). Vienna, Austria.

Resnick, M. (1996). New paradigms for computing, new paradigms for thinking. In Y. Kafai & M. Resnick (Eds.), *Constructionism in practice: Designing, thinking and learning in a digital world* (pp. 255-267). Mahwah, NJ: Erlbaum.

Resnick, M. (1998). Technologies for lifelong kindergarten. *Educational Technology Research and Development*, 46(4), 43-55.

Resnick, M. (2006). Computer as paintbrush: Technology, play, and the creative society. In D. Singer, R. Golikoff, & K. Hirsh-Pasek (Eds.), *Play = Learning: How play motivates and enhances children's cognitive and social-emotional growth*. Oxford University Press.

Resnick, M., Bruckman, A., & Martin, F. (1996). Pianos not stereos: Creating computational construction kits. *interactions*, 3(6), 41-50.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Milner, A., Silver, J., Silverman, B., & Kafai, Y. (2009). Scratch: Programming for all. *Communications of the ACM*, *52*(11), 60-67.

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

Resnick, M., & Silverman, B. (2005). Some reflections on designing construction kits for kids. In *Proceedings of the 2005 conference on Interaction Design and Children* (pp. 117-122). Boulder, CO.

Rusk, N., Resnick, M., Berg, R., & Pezalla-Granlund, M. (2008). New pathways into robotics: Strategies for broadening participation. *Journal of Science Education and Technology*, 17(1), 59-69.

Sawyer, R. (2006a). Educating for innovation. *Thinking Skills & Creativity*, 1, 41-48.

Sawyer, R. (2006b). Analyzing collaborative discourse. In *The Cambridge handbook of the learning sciences* (pp.187-204). New York, NY: Cambridge University Press.

Sawyer, R. (2012). Learning how to create: Toward a learning sciences of art and design. In *Proceedings of the International Conference of the Learning Sciences (ICLS 2012),* Sydney, Australia, 2012.

Sawyer, R., & DeZutter, S. (2009). Distributed creativity: How collective creations emerge from collaboration. *Psychology of Aesthetics, Creativity, and the Arts, 3*(2), 81-92.

Schank, R., Fano, A., Bell, B., & Jona, M. (1993). The design of goal-based scenarios. *The Journal of the Learning Sciences*, 3(4), 305-345.

Shelton, B. E., Stowell, T., Scoresby, J., Alvarez, M., Capell, M. & Coates, C. (2010). A Frankenstein approach to open-source: The construction of a 3D game engine as

meaningful educational process. *IEEE Transactions on Learning Technologies,* 3(2), 85-90.

Sipitakiat, A., Blikstein, P., & Cavallo, D. (2004**).** GoGo board: Augmenting programmable bricks for economically challenged audiences. In *Proceedings of the International Conference of the Learning Sciences (ICLS 2004)*, Los Angeles, CA.

Sonnenburg, S. (2004). Creativity in communication: A theoretical framework for collaborative product creation. *Creativity and Innovation Management, 13(4),* 254-262.

Stahl, G., Koschmann, T., & Suthers, D. (2006). Computer-supported collaborative learning: An historical perspective. In R. K. Sawyer (Ed.), Cambridge handbook of the learning sciences (pp. 409-426). Cambridge, UK: Cambridge University Press.

Strohecker, C. (1991). Elucidating styles of thinking about topology through thinking about knots. In I. Harel & S. Papert (Eds.), *Constructionism: Research reports and essays* (pp. 215-234). Norwood, NJ: Ablex.

Sullivan, F. R. (2008). Robotics and science literacy: Thinking skills, science process skills, and systems understanding. Journal of Research in Science Teaching, 45(3), 373-394.

Turkle, S. (2005). Computer games as evocative objects: From projective screens to relational artifacts. In J. Raessens & J. Goldstein (Eds.), *Handbook of computer game studies* (pp. 267-279). Cambridge, MA: MIT Press.

Turkle, S., & Papert, S. (1991). Epistemological pluralism and the revaluation of the concrete. In I. Harel & S. Papert (Eds.), *Constructionism: Research reports and essays* (pp. 161-192). Norwood, NJ: Ablex.

Vass, E., Littleton, K., Miell, D., & Jones, A. (2008). The discourse of collaborative creative writing: Peer collaboration as a context for mutual inspiration. *Thinking Skills and Creativity, 3,* 192-202.

Webb, N. M., Ender, P., & Lewis, S. (1986). Problem-solving strategies and group processes in small groups learning computer programming. *American Educational Research Journal*, *23*(2), 243-261.

Weiss, A., Wurhofer, D., & Tscheligi, M. (2009). "I Love This Dog"—Children's emotional attachment to the robotic dog AIBO. *International Journal of Social Robotics*, *1*(3), 243-248.

White, T., & Pea, R. (2011). Distributed by design: On the promises and pitfalls of collaborative learning with multiple representations. *The Journal of the Learning Sciences*, 20(3), 489-547.

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), 171-209.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3),33-35.

Winslow, L. E. (1996). Programming pedagogy—a psychological overview. *ACM SIGCSE Bulletin*, *28*(3), 17-22.

Zuckerman, O., Arida, S., & Resnick, M. (2005). Extending tangible interfaces for education: Digital montesory-inspired manipulatives. *Proceedings of the Conference on Human Factors in Computing Systems* (pp. 859-868). Portland, Oregon.

APPENDICES

Appendix A. Sample Lesson Plan

Lesson Title Tangible Digital Pets I	**Date** 11.1.11 - 11.2.11	**Lesson** # 3

Objectives

1. Students begin to design their own digital pet from "scratch".
2. Students begin to remix/reuse code from prototype pets to create their own pet.
3. Students learn to use round table sharing to get advice and learn from others.

Opener *10 min What do we know about Digital Pets?* *Brainstorm! (on post-it notes)* • *5 min - What is a digital pet? Types of digital pets.* • *5 min - Brainstorm ideas for types of interactions between pet and human or pet and computer.*	**Materials Needed** **3 cameras** **3 mics** **3 tripods** pack of post it notes 3 prototype pets 3 laptops 3 PicoBoards
Workshop *30 min + 30 min Digital Pet Designing* • *Introduce students to PicoBoards and how to get them working. Introduce students to craft materials. Say, "Your job is to design and develop a tangible digital pet of your own. Your pet should be designed for a 2nd or 3rd grader. We are going to work on these for the rest of the project. At the end, we are going to host a design exhibit, which we need to schedule, where you will each get a chance to show off your pet to family, friends, teachers and anyone interested .It will be just like an art exhibit with cheese and crackers and little napkins. The people who come will each get to give you feedback on your design, telling you what parts are good and what doesn't work so well. So you have a real reason to make your pet great. Any questions?"* • *Anticipate questions about "we don't know how to program" and "how do we even start?"* • *Explain how to save new program on desktop!! (Blah-blah-blah_intitials_version.scr)* • *Give students debug reports and tell them to fill them out as they work.*	**Materials Needed** daily debug reports

Roundtable *10 min Share*	**Materials Needed**
• *Collect debug reports* *Use prompts like:* • *Each pair should share the name of their pet and give some ideas about how and what their pet will do.* • *Any pair that would like to share something they got their pet to do should.*	Board

Appendix B. Pre and Post Survey

Digital Pets Project Appraisal Inventory

Name: Pre / Post

Date: Grade Level: 10 11 12

I am: Female Male

Read aloud to students:

This questionnaire is designed to help me get a better understanding of how you think about solving problems, computer programming and design. Please rate how much you agree with each statement by circling the number from 1 to 10. For this questionnaire, 1 means you don't agree at all, 5 means you more or less agree and 10 means you completely agree. Don't worry if there are some things you don't know about on this questionnaire. You can leave questions blank that you don't know about. I am just trying to assess what everyone knows about already and what you haven't studied yet. You won't be graded. Do you have any questions? (pause for questions).

Rate how much you agree with each statement by circling a number from 0 to 10 using the scale:

0	1	2	3	4	5	6	7	8	9	10
Don't agree at all					More or less agree					Completely agree

read: Let's start with the first question. Answer the first question now.

1. I know some computer programming

0 1 2 3 4 5 6 7 8 9 10

read: If you don't know any computer programming skip to question 8.

2. I know how to fix bugs/errors in computer code

0 1 2 3 4 5 6 7 8 9 10

3. I am confident I can fix a bug/error when my computer code isn't working right

0 1 2 3 4 5 6 7 8 9 10

4. When my computer code isn't working right, I'd rather just change my idea than have to try to figure out how to fix a problem in the code

0 1 2 3 4 5 6 7 8 9 10

5. I sometimes delete computer code and start over instead of trying to figure out how to fix it

0 1 2 3 4 5 6 7 8 9 10

6. I think debugging/fixing errors is a valuable skill

0 1 2 3 4 5 6 7 8 9 10

7. I do not like debugging/fixing my errors

0 1 2 3 4 5 6 7 8 9 10

read: Is everyone up to question 8? This next section is about making mistakes and learning. Answer the questions as honestly as you can.

8. Good computer programmers probably never need to do debugging (fix their errors)

0 1 2 3 4 5 6 7 8 9 10

9. I learn best when I have to figure out how to fix my mistakes

0 1 2 3 4 5 6 7 8 9 10

10. I like figuring out how to fix my mistakes

0 1 2 3 4 5 6 7 8 9 10

read: Questions 11 through 17 are about math. Think about your math classes at XXX High School and then answer the questions. Any questions about this section? Ok. You can start

11. Making mistakes is an important part of learning in math

0 1 2 3 4 5 6 7 8 9 10

12. Math class makes me feel like I am really good at solving problems

0 1 2 3 4 5 6 7 8 9 10

13. In math, it is ok to come up with your own way to solve problems even if it's different than what the teacher told you.

0 1 2 3 4 5 6 7 8 9 10

14. In math, there is usually only one right way to solve a problem

0 1 2 3 4 5 6 7 8 9 10

15. I don't really care if I am wrong in math

0 1 2 3 4 5 6 7 8 9 10

16. Being wrong in math means you are not smart

0 1 2 3 4 5 6 7 8 9 10

17. Math is frustrating

0 1 2 3 4 5 6 7 8 9 10

read: Is everyone up to question 18 ? You have an opportunity to take part in a project where you will use new technologies to design and

program your own tangible digital pets. When I say design, I mean create using art supplies and your imagination and when I say program, I mean computer programming. You do not need to know any computer programming beforehand. You will not be picked based on how you answer these questions, so you can be honest. For these next questions I want you to imagine what it might be like to be part of that project. So for the questions, you can predict what you think you it would be like to create your own digital pet using new technology, computer programming and art.

18. Making mistakes is an important part of learning when you are doing your own computer design project.

0 1 2 3 4 5 6 7 8 9 10

19. In design, there is usually only on right way to solve a problem

0 1 2 3 4 5 6 7 8 9 10

20. Having to debug/fix my errors during my computer design project makes me feel like I am really good at solving problems

0 1 2 3 4 5 6 7 8 9 10

21. Solving problems your own way when you are doing your own computer design project is ok, even if it's not the way the teacher does it.

0 1 2 3 4 5 6 7 8 9 10

22. Computer design projects are frustrating

0 1 2 3 4 5 6 7 8 9 10

23. Creating my own computer design project makes me want to fix my errors

0 1 2 3 4 5 6 7 8 9 10

read: For the next questions, just think about what you imagine the digital pets computer design project might be like compared to your math class in high school.

24. Math and computer design projects are really similar

0 1 2 3 4 5 6 7 8 9 10

25. The kinds of thinking you have to do in math and in computer programming design are really similar

0 1 2 3 4 5 6 7 8 9 10

~~ Tell me a little about yourself.

What kinds of things do you like to do for fun?

Why did you decide to participate in this project?

Do you have an after school job? How many hours do you usually work per week?

Do you plan to go to college?

Appendix C. Field Notes Sample

11.3
Lesson 3, Day 2

Compared to my expectations, today was an 8/10.

Having a lot of trouble staying on my lesson plans. The time is so short and having the students do anything not related to programming (when the computers and circuit boards are out in front of them when they walk in and still there when they walk out - because of time I have to have them set up before hand and we don't have time to put things away before the end of the period) just doesn't fit in that well.

Today was day 2 of the kids programming their own digital pets.

Jamal is working on a wildthing in the forest. He has been working diligently on getting his wild thing to walk across the screen. He asked me to help figure out how to get him to walk and then how to make it so he would turn around so he could walk back. He is working on understanding the repeat clause and how wait works. Also how to import new sprites and make them versions of your original sprite. I suggested that he might want to paint his own tree and have the wild thing turn around when he touched it. "Like the car in my program." He is currently doing that. He doesn't ask for help, but when I walk around and ask how he is doing, he will ask questions. He said, "I tried to make him turn around using the rotate 180 and instead it flipped upside down, how can I make him turn around?" This shows me he is working on figuring things out for himself with the skills he has learned already but isn't always getting the results he wants.

Anna is working on her dinosaur in the desert. She has programmed him to roar and to party. I think the kids have noticed that she is working alongside them and trying to figure things out, which is very cool.

Carlos and Dino and Maya are an interesting team. Dino does not like to sit down and rarely adds input or touches the computer unless C is gone. He said he hates computers today. He says they never do what you want them to. He asked several times when they were going to get to make their creature. This is the part he is looking forward to. M also does not touch the computer. She does a lot of texting. When asked, she said she didn't know what was going on. I asked C to explain the problem to her and see if she could help figure it out. I am not sure if she did that. They have an alien who rides a magic carpet and jumps on a trampoline to get to another background. C is getting the programming thing and is capable of doing much on his own, but he likes to be coached and calls me over often with his annoying meowing button. He presses it incessantly until I show up because it is so annoying.

Tegan worked alone today. She never called me over, but asked questions when I went to her. She spent much of today working on finding a background she liked for her monkey. After about half the period doing this, she started working on making him move differently based on the slider. This is the first I have seen someone try to alter how the tangible tool interacts in a significant way. She wants the monkey to have more range of motion. I told her she could use the same idea as the car but make a ladder or something

so that when the monkey touches it he can climb up onto the bed. She said that sounded so cool and asked how to do it. I said she should start by making a character that was a ladder and then I would help her. At the end of the period she showed Anna the idea and Anna said it was cool. Then Tegan said, "Don't steal it!". She has a sense of ownership over the idea and doesn't want to share it. This is interesting.

Dwayne, Tabitha and Steph worked together. Anna asked them several times to put their phones away and decide whether they were actually working on something. I am afraid I have lost them. Steph hasn't been here in ages and has no idea what's going on. Dwayne is a distraction and was asking the girls to text people for him. Tabitha seems still somewhat invested. When Steph was out of the room, she asked me what she was supposed to be doing. I said that she should make her own pet. To start she should decide what she wanted her pet to look like and then either import or paint him. Then I said she could start with a blank slate and make it do whatever she wanted. She seemed to think this made sense, but I am not sure if she followed that advice. By the end, they had a lion that they had figure out how to make roar. They seemed relatively pleased by that and a little more invested than previously. They also recorded Anna when she came over to reprimand them. Anna brushed it off, but it was the first instance of blatant insubordination that I have seen.

For the last 5 minutes we shared what we were working on and what the character could do. I should watch this tape because it seemed to inspire and motivate the kids to hear

what others were doing. Not until the last few minutes of class did anyone leave their spot to check out what someone else was doing. C &D went to see Tegan's monkey and gave her a hard time about it. I should watch that part as well.

Forgot to give out debug reports! Arg! Must remember next time.

I am not sure how much debugging experience they are getting or if they realize they are doing any debugging. I wonder if the code reuse stuff is actually more interesting here. I definitely have concrete code structure to point to to help guide them to trying new things. I have mentioned the car several times as a way to make things happen on the screen.

Similarly, I am not sure how much the tangible technology is influencing how they work or what their eventual projects will entail. I don't see them using it a lot. I see them more interested in the on screen stuff for the most part. It makes me wonder whether the tangible part of the digital pet is important or whether having a general theme, of a pet, is providing the motivation and the structure that is helping them stay on task and come up with ideas. This is like the robots article that claimed having a theme helps guide ideas (Resnick etc.).

I think Tegan is the most into her project right now. She spent a lot of time creating different features for her monkey including freckles and a mouth. The pet idea seems to

resonate with her. And since she is female, I am excited that she is into it. I don't know if another theme would have had the same motivation. I should ask this in the interview.

Appendix D. Debug Journals

Digital Pets Project Design Journal

Date: _____ Name: _____

My/Our Project is going:

4	3	2	1
Really well	Pretty Good	Not so great	Terribly

Today I/we (check all that apply ~~ you can check boxes multiple times if you wish):

☐	Tried to fix a programming error (bug)	☐	We fixed it & now things are working correctly
		☐	We are still trying to fix it
		☐	We couldn't fix it
		☐	We decided to work on other parts of our project instead of fixing it
		☐	We decided to delete the code and start over instead of fixing it
		☐	Because we couldn't figure out how to fix it, we changed our original idea to make the coding easier
☐	Did not try to fix a programming error (bug)		

How much of today's workshop did you spend Debugging? (circle one)

None A little (less than half) Some (half) Most (more than half) All

Did you seek help from other students in the project today? (circle one)

Yes No

Did you seek help from the teacher/facilitator today? (circle one)

Yes No

VITA

Maneksha Katrine DuMont

Education

2009 – present	Candidate, Instructional Technology and Learning Science UTAH STATE UNIVERSITY Logan, UT *Thesis: Engaging alternative high school students through the design and development of tangible digital pets* Committee: Victor Lee (chair), Brett Shelton, Mimi Recker, Deborah Fields, Andrew Walker, David Smellie
2007 – 2009	MS, Learning Science NORTHWESTERN UNIVERSITY Evanston, IL *Masters Project: Starting From Scratch: Creating Meaning through Creativity, Community and Computer Programming* Project Committee: Edd Taylor (chair), Andrew Ortony
2002 – 2003	MSED, Secondary Mathematics Education NORTHWESTERN UNIVERSITY Evanston, IL *Masters Thesis: Student-Centered Learning in Mathematics*
1995 – 1999	BA, Mathematics (with distinction) Concentration: Operations Research CORNELL UNIVERSITY Ithaca, NY

Honors

Utah State University Conference Fellowship (2010)
Utah State University Vice President for Research Fellowship Recipient (2009-2010)
Northwestern University Fellow (2007-2008)
BYU Literacy Institute Summer (2006)
Utah Principles' Conference Attendee (2006)
Praxis Recognition of Excellence Award in Mathematics (2006)
Suave Performance Plus Teaching Award Nomination (2004, 2005)

Professional Memberships

International Society of the Learning Sciences
American Educational Research Association
National Council of Teachers of Mathematics (2003 – 2004)
Metropolitan Math Club of Chicago (2003-2004)

Professional Experience

Research Assistant 2011-2012
Utah State University, Logan, UT
Physical Activity Data Project, Funded by the National Science Foundation

Secondary Mathematics Teacher
Treasure Mountain International School
Park City, UT 2006 - 2007
XXXXX High School
XXXXXXX, UT 2005 - 2006
Walter Payton College Preparatory High School
Chicago, IL 2003 - 2005

Software Engineer
Styleclick Inc. (formerly MVP.com and BigEdge.com)
Chicago, IL 1999 - 2002

Refereed Publications

Journals

Lee, V. R., & DuMont, M. (2010). An exploration into how physical activity data-recording devices could be used in computer-supported data investigations. *International Journal of Computers for Mathematical Learning*. 15(3), 167-189.

Conference Proceedings/Conferences

DuMont, M. & Fields, D. (2013). Hybrid Shmybrid: Using Collaborative Structure to Understand the Relationship Between Virtual and Tangible Elements of a Computational Craft. *CSCL13* Madison, Wisconsin: International Society of the Learning Sciences.

DuMont, M. (2012). Empowerment through design: engaging alternative high school students through the design,

development and crafting of digitally-enhanced pets. In *Proceedings of the 11th International Conference on Interaction Design and Children* (IDC '12). ACM, New York, NY, USA, 343-346.

DuMont, M. & Lee, V. R. (2012). Material pets, virtual spaces, isolated designers: how collaboration may be unintentionally constrained in the design of tangible computational crafts. In *Proceedings of the 11th International Conference on Interaction Design and Children* (IDC '12). ACM, New York, NY, USA, 244-247.

DuMont, M. & Lee, V. R. (2012). A Technologically Enhanced Construction Kit as a Support for Children's Collaborative Computational Thinking. Paper presented at the 2012 annual meeting of the American Educational Research Association, Vancouver, B.C.

Lee, V. R., & DuMont, M. (2010). Students' Investigations with Physical Activity Data Devices. In K. Gomez, L. Lyons & J. Radinsky (Eds.), *Learning in the Disciplines: Proceedings of the 9th International Conference of the Learning Sciences (ICLS 2010)* (Vol. 2, pp. 344-345). Chicago, IL: International Society of the Learning Sciences.

Berland, M., Lee, V. R., & DuMont, M. (2010). Small Groups, Big Mistakes: The Emergence of Faulty Rules During a Collaborative Board Game. *Proceedings of the Ninth International Conference of the Learning Sciences (ICLS 2010).* Chicago, IL: International Society of the Learning Sciences.

Teaching Experience

Teaching Assistant, Northwestern University Fall 2008
Macrocognition: Intelligence in Context (LOC 301)

Teacher, Secondary Mathematics 2003 - 2007

Volunteer Positions

Technology Assessment Committee & STE(Art)M Committee
2011-present