From Photoshop to Programming

Yasmin Kafai, Kylie Peppler, Grace Chiu, John Maloney, Natalie Rusk, and Mitchel Resnick

Walk into any Computer Clubhouse and you are likely to see members creating and manipulating graphics, animations, videos, music, and often integrating multiple media. The professional image-processing tool, Adobe Photoshop, is particularly popular. Indeed, a "Photoshop culture" has emerged at many Clubhouses, with youth proudly displaying their visual creations on bulletin boards in the Clubhouse and in the Village, sharing Photoshop techniques and ideas with one another, and helping Clubhouse newcomers get started with the software. Clubhouse members, like many youth today, are familiar with a wide range of design tools they can use on computers for creative production; programming tools, however, are most often not considered part of this portfolio (Margolis, 2008).

As a case in point, we report on our observations in the Youth Opportunities Unlimited Inc. Computer Clubhouse that was opened in 2001 and thus is perhaps more typical of other Clubhouses in the network than the Flagship, with its long-standing history. Two full-time coordinators run the day-to-day operations and facilitate activities at the Clubhouse with the help of several mentors (see Chapter 8). From observations in 2003–2004, we knew that programming activities did not occur. Although Microworlds software, a visual Logo computer programming software, was available as part of a broad suite of software, neither adult coordinators nor members used it. While the most popular software titles enabled multiple media integration and manipulation, programming was considered a "stand alone" task and was therefore perceived as incompatible and irrelevant to existing popular design activities. How programming became part of this one Clubhouse's portfolio of design activities and what members learned and how they thought about programming will be the focus of this chapter.

THE DESIGN OF A NEW PROGRAMMING TOOL—SCRATCH

Through prior observations at the Clubhouses, we found that youth have an interest in video games, music videos, cartoon animations, and interactive, design-based art, all of which are a natural springboard into creating and programming.

Thus we started with addressing the overly narrow notion of programming by focusing on the cultural artifacts that it could produce. This led us to recognize the benefits of programming as creative media production, which included a broader range of digital artifacts, ranging from video games to "media mixes" of images, video, and texts. With that in mind, we set out to create a media-rich programming environment, called Scratch (see Chapter 4), which would provide youth with the ability to import and manipulate various media files that could be integrated with existing creative software. Arguably a full-fledged programming language, Scratch vastly differs from other novice-friendly visual programming environments in that it utilizes a user-friendly building block command structure, eliminating the risk of syntax errors. More information about the design of Scratch can be found in Resnick et al. (2003) and Maloney et al. (2004).

Our observations revealed that several pathways into the programming culture evolved over time at the Clubhouse. One of the coordinators introduced Scratch in late 2004. Although Scratch was loaded on several of the computers at this time, fewer than 10 members took advantage and created anything using the new software. Beginning in 2005 a steady stream of undergraduate mentors joined the Clubhouse and the first explosion of Scratch activity was seen starting in early January 2005. Youth were encouraging one another to try out the program, and mentors worked with youth to create their first Scratch projects. Commonly, mentors would engage youth who had never worked in Scratch before by suggesting that they import some of the pictures that they had stored in their folders on the Clubhouse server (see Chapter 8). At this point in time the server featured a predominance of graphics-only projects that lacked any computer programming, which was due in part to the high volume of youth opening the program without any official orientation. Printouts of projects quickly began to cover the walls, and Scratch became the leading design activity within a few months of its introduction.

In the beginning of 2006 there was an even greater interest in Scratch and some new things began happening within the Clubhouse culture. Scratch was used among the youth as a measure of membership in the local culture: New members wanting to establish clear membership in the community had to first create at least one Scratch project and store it for others to play on the central server. For the first time, more expert youth were seen mentoring other youth in Scratch. Scratch experts had a high-status position within the local culture, with some youth emerging as general experts, whom mentors, coordinators, and other youth consulted for help with Scratch, and other youth specializing in certain genres or tricks within Scratch. In addition, groups of youth had begun working collaboratively together to create projects with a group name, such as DGMM, for the Dang Good Money Makers. Youth also began to work independently of mentoring support, reflective of the high volume of projects beginning in June 2006, on complex projects and problems that they encountered in Scratch.

THE CLUBHOUSE DESIGN CULTURE

To further understand the impact of introducing new design software into the Clubhouse environment, we examined in 213 field notes the design focus of vari-

ous activities distinguishing between design, games, Web, homework, and social activities. A subset consisting of 64 field notes was coded by four graduate students and revealed a reliability of 85–92%. Design activities involved the use of programming, 3-D animation, and graphic software such as Kai's SuperGoo, Bryce5, Photoshop, KidPix, game design programs such as RPG Maker, and music production software. Game activities included games on the computer, such as Roller Coaster (Tycoon), School Tycoon, video and online games, such as Whyville.net, as well as board and card games, foosball, and air hockey. Web activities involved Web surfing with a Clubhouse member, while homework involved mentors helping youth with their homework. We also created a "Personal" category to include all social activities and interactions between the mentor and Clubhouse youth that establish and build upon the interpersonal relationship outside of the context of the other activities. We interpret these findings as being a proxy for Clubhouse activities, of which we would otherwise have no indication.

One general outcome is that programming activities increased as Scratch became embedded in the popular suite of design tools that Clubhouse members utilized on a daily basis (Kafai, Peppler, & Chiu, 2007). Initially, design activities comprised about 70% of all Clubhouse member activities followed by homework (14%), Web surfing (8%), and games (8%). After Scratch was introduced, it comprised 25% of all activities while other design activities accounted for 18%, followed by an increased interest in games (30%) and personal activities (17%), while homework and web Surfing decreased (5% each).

The design portfolio illustrates how programming had become part of the Clubhouse activities. Over the course of 2 years we tracked Scratch development and collected all projects created by Clubhouse members. There were several reasons for this approach, but important to the purposes of this chapter is that it allowed us to peek at the computing culture when even mentors and researchers were not present. The number of new Scratch projects is also a good indication of general interest in computer programming over time. We observed that the majority of interest in Scratch occurs from January through August, and there was less interest in the fall months between September and December. This is probably due to several reasons, but can be somewhat explained by the presence of mentors from January through March. There was also a high volume of projects being created over the summer months (especially in June and July of 2006) in the absence of extensive mentoring support. We interpret this as an indication of the extended and prolonged impact that mentoring support can have on a programming culture beyond (or at least temporarily beyond) the weekly visits of the mentors.

LOCAL DESIGN CULTURES: THE LOW RIDA PROJECTS

The total number of Scratch projects paints a picture of an active computing culture, but what exactly are youth creating in Scratch? Because Scratch was designed to flexibly promote self-expression, youth appropriated the software in a number of ways. Over the course of 18 months, we collected 536 programming projects created by members of the Clubhouse, some designed alone and others with mentors

(Kafai, Peppler, Alavez, & Ruvalcaba, 2006). We found that 44% of these projects fell into the category of animations with and without user manipulation, followed by 23% of graphics-only projects, and 15% of game projects focusing on fighting, sports, and adventure; 14% of projects escaped a clear categorization because they did not provide enough detail. The quantitative changes in design and mentoring activities were accompanied by qualitative changes in Scratch program genres and Clubhouse members' conceptions of programming.

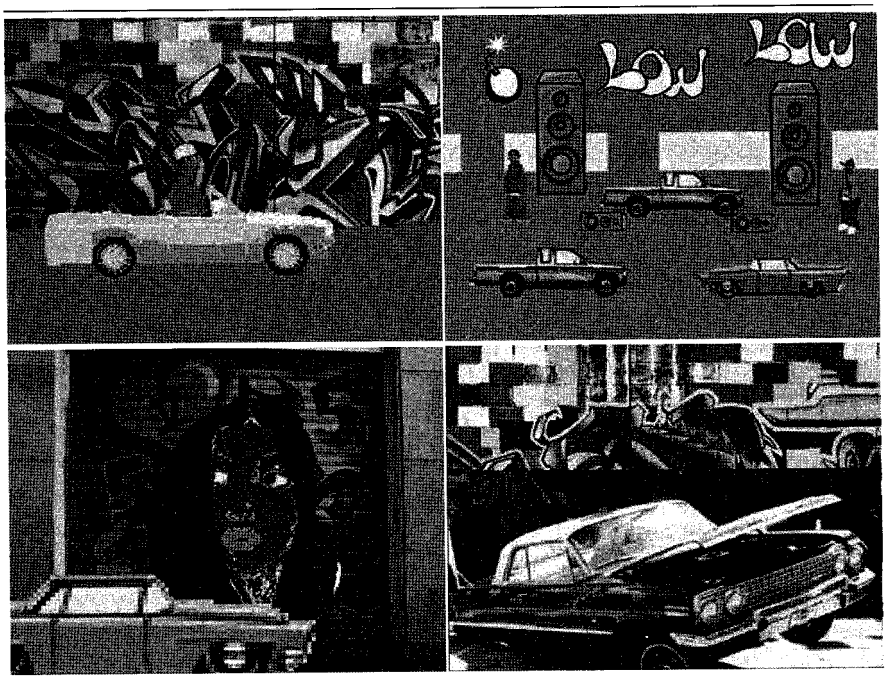
Some youth emerged who took on strong leadership roles. These leaders began to work with groups of 10–12 other youth to seemingly manufacture certain genres of projects; one example of this is the Low Rida movement that began in January 2006. Within urban youth cultures there is a lot of interest in customizing cars. Television shows, like MTV's *Pimp My Ride,* have popularized this trend within mainstream American culture. Previously in the Clubhouse, a popular activity was to manipulate digital pictures of expensive cars, inserting a picture of yourself next to "your" car. Made popular by a young biracial African American/Latino youth named Dwight, a culture of Low Rida interactive art projects has emerged. In one of Dwight's first projects, "Low Low," the viewer controls the hydraulics on two cars using arrow and letter keys. According to Dwight, the essential parts to his "Low Rida" project are the cars, the urban background, the graffiti-like lettering, and the speakers (see Figure 12.1). It is important to note that the Low Rida movement emerged in the absence of mentoring support. The Clubhouse members conceptualized the idea and executed the projects almost entirely by themselves.

Several new Low Rida projects have emerged based on Dwight's earlier work, resulting in a widespread use of Scratch. In these projects, the creators have used Photoshop, Painter7, Image editors, and computer programming for creative production. By participating in the Low Rida movement, youth gain access to skills, empowering them to become designers of digital media. This is an important aspect of participation in an informal learning culture where contribution is valued. Projects like these eliminate barriers between high and low pop cultures by taking an urban youth culture theme and reinventing it using high-status knowledge, such as software design.

SCRATCH PROGRAMMING PERFORMANCES

In a further step, we used the archive of Scratch projects to analyze their content for programming scripts. While most programming assessments focus on various aspects of individual or team programmers' performances in coding and debugging a piece of software, here we decided to evaluate the performance of a community by examining the content of the archive for essential programming concepts. The goal of this analysis was to study the extent to which youth touched upon these concepts as well as to gauge if the community as a whole increased their knowledge of computer programming over time. Of the 536 projects, 111 of them contained no scripts at all. These "pre-scripting" projects illustrate the use of Scratch simply as a media manipulation and composition tool. Beginning Scratch

FIGURE 12.1. Screenshots of Dwight's "Low Rida" projects are in the upper and lower right corners. Other members of the clubhouse created the two other "Low Rida" Scratch projects. In the upper left, Dwight's brother customized his ride by painting it gold and drawing in gold hubcaps. In the lower left, an 8-year-old girl created her own version of the "Low Rida" project, inserting a portrait that she created of herself using Painter7 software.



users often spend time importing or drawing images and recording sounds before moving on to scripting.

Of the remaining 427 projects, all of them created scripts and 374 of them had scripts run in parallel. These are core ideas in programming that confront every Scratch programmer when they begin writing scripts. About half of the projects provided keyboard or mouse input for users of their programs. On the other end of the spectrum, variables, random numbers, or Boolean logic were only used in few projects; these are concepts that are not easily discovered on one's own. In fact, one Clubhouse member had a desperate need for the variables in his project. When Mitchel Resnick, on a visit to the Clubhouse, showed him how to use variables, he immediately saw how they could be used to solve his problems and thanked Mitchel repeatedly for the advice.

When we compared the distributions from the first year to the second year of our data collection, we found that four of the seven programming concepts (loops, Boolean logic, variables, and random numbers) demonstrated significant gains in the number of projects utilizing the targeted concepts. One of the remaining concepts (i.e., conditional statements) had marginal gains, and one concept (communication/synchronization) demonstrated a significant reduction in the number

of projects utilizing this particular concept. For more detail see Maloney, Peppler, Kafai, Resnick, and Rusk (2008).

Given the popularity of games and animation, it is not surprising that projects showing user input were common. It was a pleasant surprise to find that the commands for synchronizing interactions between objects were also fairly heavily used; interobject communications is one of the most complex ideas in Scratch, but it answers a critical need when building more complex projects. On the other hand, Boolean operations, variables, and random numbers were less obvious objects. We also examined trends over time. In general, the number of projects produced in the second year doubled the number of projects produced during the same period the first year of the project. When we compared the percentage of projects containing the various programming concepts over time, we found that most of the concepts that we targeted for our analyses demonstrated significant gains during the second year.

PERCEPTIONS OF PROGRAMMING

What did the Clubhouse members in this study have to say about their Scratch programming experiences? We interviewed a large number of youth to better understand how they are making sense of and appropriating Scratch. Each interview lasted about 15–20 minutes and questions included the following: What is computer programming to you? Does Scratch remind you of anything that you do at school or at home? How does Scratch differ from other computer software programs? All of the interviews were transcribed in preparation for later analyses. Researchers coded for themes rather than individual statements because these were group interviews and participants often expressed agreements with statements voiced by others; thus we did not expect every participant to repeat impressions.

General conceptions of Scratch were overwhelmingly positive with Clubhouse members proclaiming that it's their "favoritest thing ever." According to youth, Scratch is extremely flexible and has no, or few, limitations. Having trouble defining what Scratch was exactly, most youth described it as "something that allows you to use your imagination" or as "a system that will allow you to do whatever you want." Most youth cited at least four to five different applications for which Scratch could be used, including making games, Low Ridas, comics, animations, music videos, short movies, and digital art. Although youth could recall a great deal about how to create projects in Scratch, citing specific commands and naming specific parts of the screen, most youth were unaware that creating in Scratch would be considered "computer programming." In fact, over half of the youth were unable to define what computer programming was.

If youth do not recognize that they are learning programming through Scratch, what do youth believe that they are gaining from their experience? Clubhouse members reported a wide range of connections to traditional subject areas such as math, reading, science, and foreign language learning, in addition to strong connections to the arts. The following excerpt is taken from an interview with Arnold, a 14-year-old African American boy with limited Scratch experience, as

he recounts his personal connection to Scratch through his experience as an actor. Notably, he cites how drama could be extended and reinforced in certain ways through Scratch.

> *Arnold:* Well let me see . . . . Well Scratch it really brings out my potential and it actually brings out my acting experience.
> *Kylie:* How so?
> *Arnold:* Well when you take the microphone, you can create your own voice for your character. Like I love Arnold Schwarzenegger. Yeah it just really brings out your potential . . . . Thinking of what you're doing with acting you can take it out of your mind and say like "in this picture we want to like do action stunts like flips and stuff", and if you're at school you're like doing Romeo and Juliet. You can make it more funny [in Scratch] by putting in some dragons. You can make a dragon go up to a castle and say "I came to rescue you". . . . Then you put them all in their places [in Scratch] and then once we do "Action!" We all come in with our parts.

Although we don't intend for youth to become hacker types as a result of their experience in Scratch, the involvement in the design process has awakened new possible career opportunities for some of the youth, notably the teenage boys. As one member puts it, "It teaches how to play games and make games and it helps us figure out our future." This particular youth would now like to be a professional video game designer, to attend college at MIT, and perhaps someday design a program like Scratch. He revels in his conversations with the professional programmers of Scratch and thoughtfully comes up with suggestions for how to further revise Scratch. It's clear that experiences like the ones at this Computer Clubhouse can have a considerable impact on the outlook and career aspirations of young people. Clearly, this is an area worthy of further exploration if we intend for youth to enter the computer science pipeline through informal avenues of education.

Most youth didn't identify scripting in Scratch as a form of programming. In general, when youth were asked, "What is computer programming to you?" they responded: "Computer programming? I do not have a clue [what that is]!" At first we were concerned that youth didn't make the connection between Scratch and programming. But on reflection, not seeing Scratch as "programming" may have helped Scratch catch on, allowing youth to see Scratch as being in line with their identities as kids, as something "cool," and as a central part of the Clubhouse culture. After all, the point of engaging youth in computer programming is not to turn them into hackers or programmers, but because being engaged in the full range of technology fluencies—including programming—is an educational right of the twenty-first century. This point becomes even more important when over 90% of the youth that come to the Clubhouse have never been in a computer class during their entire K–12 schooling experience. The Computer Clubhouse then becomes an important space for access to computer programming tools and skills.

IMPLICATIONS FOR CREATING DESIGN CULTURES

A simple story of our efforts to seed a programming culture in the Computer Clubhouse would focus on the Scratch software alone. Our results indicate that Scratch indeed was integrated into the portfolio of design activities in this particular Clubhouse, yet the true test of diffusion and integration will come as we are releasing the software to other Clubhouses within the network and beyond. We found that, on their own, Clubhouse youth discovered and used many commands while others hardly were used. These findings are especially surprising given the lack of formal instruction and the fact that the mentors had no prior programming experience. As part of our intervention, Scratch was never intended to be a shrink-wrapped package that was simply handed to members; rather, it was introduced in tandem with other changes at the Computer Clubhouse.

The introduction of both Scratch and undergraduate mentors would not have been possible without a change in relationships at the Clubhouse. A formal partnership was forged between the university and the Clubhouse's community host organization in order to gain support from the organization's infrastructure for these changes. By establishing goals, expectations, and communication protocols with the community organization, we were able to gain crucial buy-in on multiple levels, from the director to the coordinators. Through these various changes, a culture of programming began to emerge more in line with the initial vision of the technology fluency aspect of the Clubhouse learning model.

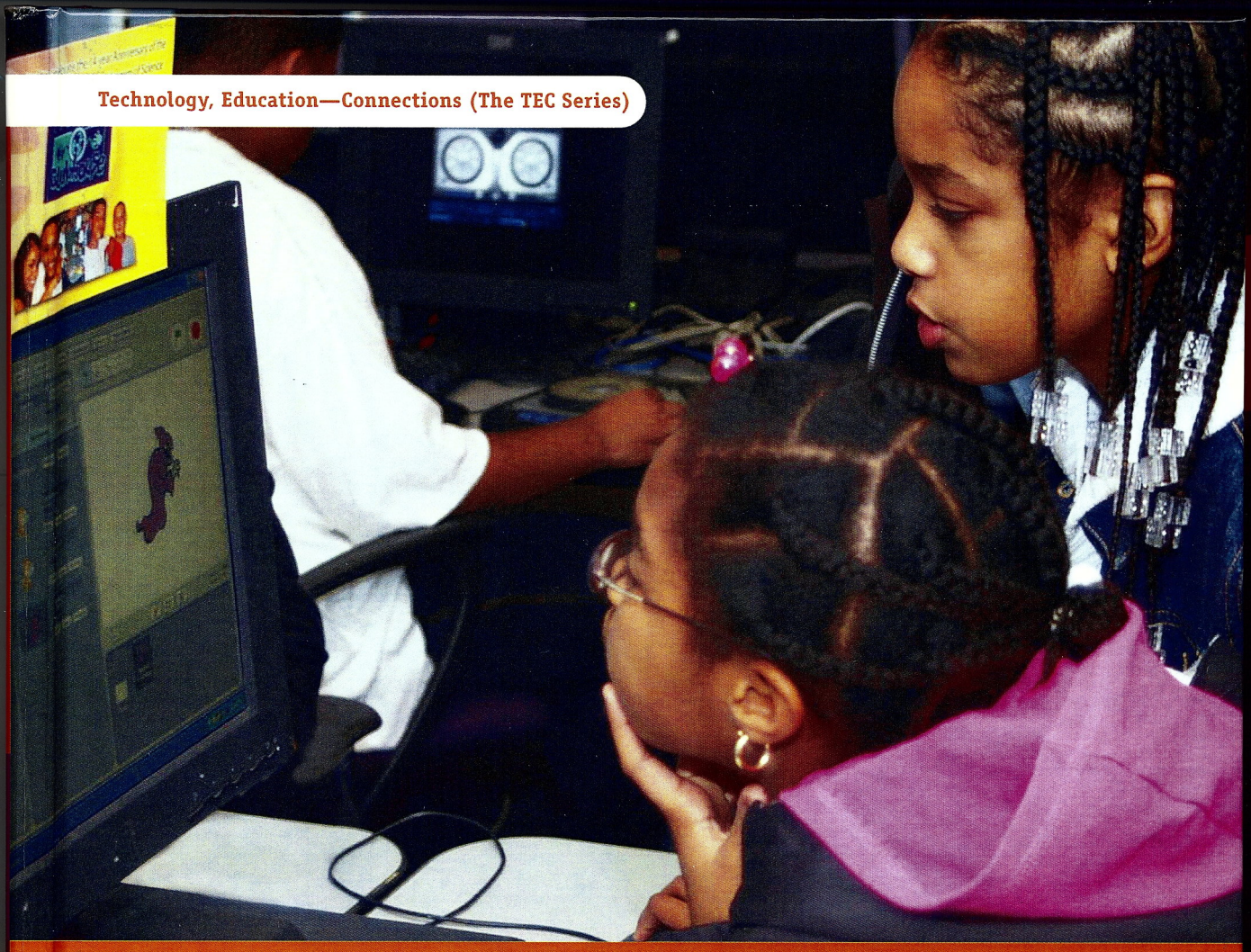
An equally pressing question is, of course, why did Clubhouse youth choose to get involved in Scratch programming given that they had many other design options? A good answer might have been provided by Caitlin Kelleher and Randy Pausch (2005), who noted how programming can become more accessible for novices "by simplifying the mechanics of programming, by providing support for learners, and by providing students with motivation to learn to program" (p. 131). We think that Scratch addresses all three of these areas. For one, the design of the Scratch blocks simplifies the mechanics of programming by eliminating syntax errors, providing feedback about placement of command blocks, and giving immediate feedback for experiments. Furthermore, we think that the social infrastructure of the Clubhouse is important in providing support for novice programmers. While the mentors did not have any prior programming experiences—all of them were liberal arts majors—they were willing to listen and encourage youth in pursuing their programming projects. Often we could observe youth recruiting mentors to be collaborators or sounding boards for their project ideas. At times, we saw Clubhouse youth teach mentors a few things they had learned about Scratch. While mentors are often associated with being more knowledgeable than their mentees, here we found a more equitable relationship that turned both members and mentors into learners. This need for an audience and resources may also explain the success of the recently opened Scratch Web site (http://www.scratch.mit.edu), which allows programmers to upload their projects and share them with others.

Finally, we think that the multimedia aspect of Scratch facilitated urban youth's engagement in programming. The project archive provided ample evidence that

Clubhouse members were savvy about various media genres and interested in not only using them but also producing their own versions. Many Scratch programs started with images pulled from the Web and centered on popular characters. In fact, we have evidence from other analyses that Scratch projects focused on generic characters were more often abandoned than those that used popular characters. Youth interest in technology starts with digital media and might thus serve as a more promising pathway into programming. The broad spectrum of media designs—from video games to music videos and greeting cards—is a true indicator of youth's interest in not only being users of digital media (as they do on a regular and personal basis) but in going beyond mere consumption to become content creators themselves, a role often denied to urban youth. We argue that youth require technological fluency of how to construct new media in order to become critical consumers and producers. We think that such directions in community technology developments are particularly important for urban youth, who are often seen as pushing new adaptations and transformations of media, but are also perceived as standing on the sidelines of technology development and production.

As illustrated in the examples of Clubhouse work, multiple aspects of media-rich production in informal settings provide youth access to technological fluency that empower them as designers in a setting where their contributions are valued. Our approach to technological fluency in the media-rich Scratch software and in the programming projects in the Clubhouse was grounded in youth practices. Previous discussions have cast this issue mostly in terms of access to digital equipment, talking about the Digital Divide when, in fact, the focus should be on the participation gap that exists in today's society. It is here that our work with Scratch production gathers particular relevance in light of the inequitable access and participation of minority and low-income youth in digital technologies. Technological fluency is not just about knowing how to code, but also involves the personal expression as illustrated in the previous examples. These projects emphasize graphic, music, and video—media that have been found to be at the core of technology interests for youth. Youth needs to be encouraged to become consumers, designers, and inventors with new technologies. Places like the Computer Clubhouse can provide access to creative, critical, and technical media-production skills such as programming in low-income communities and fill a gap not covered elsewhere.

Technology, Education—Connections (The TEC Series)



Edited by YASMIN B. KAFAI, KYLIE A. PEPPLER, AND ROBBIN N. CHAPMAN

THE COMPUTER CLUBHOUSE

CONSTRUCTIONISM AND CREATIVITY IN YOUTH COMMUNITIES

Forewords by BARTON J. HIRSCH AND ROSALIND HUDNELL

Contents

Foreword *by Barton J. Hirsch* ix

The Computer Clubhouse: The Intel Perspective *by Rosalind Hudnell* xi

Acknowledgments xiii

Introduction: The Computer Clubhouse: A Place for Youth 1
Yasmin Kafai, Kylie Peppler, and Robbin Chapman

PART I: THE COMPUTER CLUBHOUSE MODEL

1. **Origins and Guiding Principles of the Computer Clubhouse** 17
Natalie Rusk, Mitchel Resnick, and Stina Cooke

2. **Going Global: Clubhouse Ideas Travel Around the World** 26
Patricia Díaz

3. **Perspectives from the Field: It Takes a Village to Raise a Clubhouse** 35
Kylie Peppler, Robbin Chapman, and Yasmin Kafai

PART II: CREATIVE CONSTRUCTIONS

4. **Making Games, Art, and Animations with Scratch** 47
Kylie Peppler and Yasmin Kafai

5. **Interface Design with Hook-Ups** 58
Amon Millner

6. **Youth Video Productions of Dance Performances** 71
Kylie Peppler and Yasmin Kafai

PART III: COLLABORATIONS IN THE CLUBHOUSE COMMUNITY

7. Encouraging Peer Sharing: Learning Reflections in a Community of Designers 81
Robbin Chapman

8. The Multiple Roles of Mentors 90
Yasmin Kafai, Shiv Desai, Kylie Peppler, Grace Chiu, and Jesse Moya

9. The Computer Clubhouse Village: Sharing Ideas and Connecting Communities of Designers Across Borders 100
Elisabeth Sylvan

PART IV: SHOWCASES OF COMPUTER CLUBHOUSE SUCCESSES

10. Participation, Engagement, and Youth Impact in the Clubhouse Network 111
Gail Breslow

11. Hear Our Voices: Girls Developing Technology Fluency 125
Brenda Abanavas and Robbin Chapman

12. From Photoshop to Programming 136
Yasmin Kafai, Kylie Peppler, Grace Chiu, John Maloney, Natalie Rusk, and Mitchel Resnick

Epilogue: A Place for the Future 145
Yasmin Kafai, Kylie Peppler, and Robbin Chapman

References 151

About the Contributors 156

Index 159

This book is about the Computer Clubhouse—the idea and the place—that inspires youth to think about themselves as competent, creative, and critical learners. So much of the social life of young people has moved online and participation in the digital public has become an essential part of youth identities. The Computer Clubhouse makes an important contribution not just in local urban communities but also as a model for after-school learning environments globally. This model has been uniquely successful scaling up, with over 100 clubhouses thriving worldwide. Showcasing research by scholars and evaluators that have documented and analyzed the international Computer Clubhouse Network, this volume considers the implications of their findings in the context of what it means to prepare youth to meet the goals of the 21st century.

Book Features:

- A successful, scalable model for providing at-risk youth a rich array of media design and computing experiences.
- Diverse examples of media created in the Clubhouse, ranging from digital stories, video games, interface designs, and digital art projects.
- Color photos of life in the Clubhouse, including youth projects.
- Interviews with stakeholders in the Clubhouse Network, from the director to coordinators at various international clubhouses.

"It is difficult to conceive of an after-school setting that would have a greater emphasis on positive youth development.... Beyond learning computer programming, young people at the Clubhouses learn marketable skills in product design, project management, teamwork, marketing, and communication.... Read [these chapters], appreciate what has already been accomplished, and consider the exciting possibilities for the future."

—From the Foreword by **Barton J. Hirsch**, Northwestern University, author of *A Place to Call Home: After-School Programs for Urban Youth*

"As you will read in this book, the impact of the Computer Clubhouse on underserved youth around the world has been far-reaching, long-lasting, and life-changing."

—From the Foreword by **Rosalind Hudnell**, Intel Corporation

"Essential reading for anyone concerned with the development and education of contemporary youth.... Its lessons go far beyond the Clubhouse."

—**Michael Cole**, author of *The Fifth Dimension: An After-School Program Built on Diversity*

Contributors: **Brenda Abanavas, Gail Breslow, Grace Chiu, Stina Cooke, Shiv Desai, Patricia Díaz, John Maloney, Amon Millner, Jesse Moya, Mitchel Resnick, Natalie Rusk,** and **Elisabeth Sylvan**

Yasmin B. Kafai is a professor of Learning Sciences at the Graduate School of Education at the University of Pennsylvania. **Kylie A. Peppler** is an assistant professor of Learning Sciences at Indiana University, Bloomington. **Robbin N. Chapman** is a learning technologies consultant.

ISBN 978-0-8077-4990-6

TEACHERS COLLEGE PRESS

Teachers College
Columbia University
New York, NY 10027
www.tcpress.com

Cover photo by Jae Chung at the Flagship Clubhouse at the Museum of Science, Boston, which is also the global headquarters for the Intel Computer Clubhouse Network.