ISSN: 2321-2152

IJMECE

International Journal of modern electronics and communication engineering

E-Mail
editor.ijmece@gmail.com
editor@ijmece.com

www.ijmece.com

ISSN2321-2152www.ijmece .com
Vol 6, Issue 3Aug 2018

Programming by Choice: Urban Youth Learning Programming with Scratch

Imtiyaz Khan,MohdIrshad, Gayatri,

ABSTRACT

This article introduces Scratch, a block-based programming language with a visual interface that was created to make it easier for non-expert programmers to manipulate media. We present an 18-month long study of Scratch programming by urban kids (ages 8-18) in an after school organization called a Computer Clubhouse. We analyzed 536 Scratch projects created during this period and found that students were able to acquire fundamental programming principles without the aid of teachers or mentors. We examine the consequences for teaching programming at after-school settings in impoverished neighborhoods, as well as the reasons why urban adolescents pick Scratch over the numerous other software packages accessible to them.

Categoriesand Subject Descriptors

First Grade [Computer and Information Science Education]: Computer Science Instruction

Keywords

Scratch, accessibility, and beginner-friendly programming environments.

INTRODUCTION

Mentoring, altered curriculum, tool creation, outreach initiatives, and programming courses for non-majors have all been considered as potential solutions to the problem of low computer science enrollment in K-12 and higher education. Learning to code at community tech centers that provide free daily access is one area that has garnered surprisingly little attention. Summer camps and after-school activities have been more well-liked but have had shorter lifespans [1; 15]. Here, the student has more control over the pace, content, and length of their coding sessions than they would in a traditional classroom setting. With the inclusion of this notice and the complete citation on the first page, you have permission to make digital or physical copies of all or part of this work for personal or classroom use without payment. Other forms of duplication, publication, hosting, and distribution to lists are prohibited without express permission and/or payment. as opposed to being part of a mandatory school curriculum. While there has been much study of computer science education at the university level, little has been done to investigate or record extracurricular options for pre-college students to learn computer programming.

More chances to study computer programming may be made available via summer camps, after-school programs, and community technology centers. Most schools rely on technology to provide curriculum, but few provide low-income pupils with access to programming education [3]. The fact that students only spend 9% of their formative years in school is another factor that prompts one to think about other contexts that impact education [14]. Finally, adolescents who struggle in a classroom setting might find success in extracurricular activities.

This study explores the use of Scratch, a block-based programming language that helps new programmers manipulate media [11], in an urban after-school technology club called Computer Clubhouse. We compiled a database of 536 Scratch projects made by kids from a single Computer Clubhouse and studied how they evolved in terms of the commands and ideas they were using. We also conducted interviews with Clubhouse regulars to get their take on programming and Scratch. We cover the reasons why young people select programming over other applications and what newbie programmers may learn outside of a classroom setting.

Department of cse
imtiyaz.khan.7@gmail.com, mohdirshadislclg@gmail.com, gayatri12islclg@gmail.com,
ISL Engineering College.
International Airport Road, Bandlaguda, Chandrayangutta Hyderabad - 500005 Telangana, India.

1. SCRATCH

The MIT Media Lab's Lifelong Kindergarten Group and YasminKafai's group at UCLA developed Scratch together. Although it is geared towards beginners, Scratch is not the first environment or language of its kind. In fact, Kelleher and Pausch [7] and Guzdial [4] provide exhaustive overviews of the vast historical background of various advances. Scratch expands upon Logo's [8] concepts with a drag-and-drop interface reminiscent of LogoBlocks [2] and EToys [13], rather than the coding technique used by Logo itself. Scratch is a programming environment designed with kids in mind, with an emphasis on media tinkering and the encouragement of projects like animated storyboarding, game design, and interactive presentation making. The two main components of a Scratch project are the stage (background) and the sprites (characters). Images, audio, variables, and scripts are all stored separately in each individual object. Spreading and trading sprites is made simple by this group.

Stacks of blocks are created by dragging and dropping command blocks from a palette into the scripting pane. This is similar to putting together a puzzle. To activate a block or a group of blocks, just double-click on the desired block or stack. When a program starts up, a certain key is pushed, or the mouse is clicked on the sprite, the hat block on top of the stack is activated. Due to the fact that numerous stacks may operate simultaneously, most Scratch users are likely making use of multiple threads without even recognizing it.

There are four distinct regions on the Scratch screen (shown in Figure 1). The performance area is on the right. A full-screen presentation of the stage may be accessed by a button on the bar below the stage. There's a section down there with preview images of all of the project's sprites in it, just below the stage. If you click on a preview, the appropriate sprite will be selected. The sprite's script, costumes (graphics), and noises may all be seen and edited in the center window. To the left of the programming area is a palette containing several command blocks. There are eight different types of colors to choose from.

The inspiration for this user interface design came from a need to concretize the core ideas behind Scratch. The command palette's persistent visibility encourages tinkering. If you find a command in the palette that seems intriguing, just double-click it to learn more about it. As the scene progresses, the user may see relevant stacks in the scripting area becoming highlighted. The user is given a process model of how their scripts are read by the computer, which is aided by the simultaneous visibility of the palette, writing area, and stage.

Figure1:ScreenshotofScratchInterface

2. There are about 90 different commands available in Scratch, and they cover a wide range of topics, from relative movement (a la the Logo turtle) to absolute positioning with Cartesian coordinates, from image transformations (rotation, scaling, and effects like Fisheye), to cell animation (the switching between images), from recorded-sound playback to musical note and drum sounds, and even a programmable pen. For the Scratch user, this provides a rich framework in which to develop a deeper grasp of numbers, since many of the instructions need them as input. If you use a negative parameter with the move command, for instance, the sprite will travel in the opposite direction. Currently, only basic arithmetic, comparison, and Boolean operations are implemented; however, more complex scientific operations (such as sine) are on the roadmap. A sprite's proximity to an edge, another sprite, or a certain color may be detected with the use of special sensor blocks.

3. relay information about the position of the mouse or the pressing or releasing of a key.

4. Scratch's control structures include if/else statements, while loops (with varying iteration lengths and a variety of conditions), and event triggers (when-clicked, when-key-pressed). Named broadcasts are used for communication. If one sprite yells "you won!" another one can arrive on stage and start singing a triumphant tune. Multiple scripts may be activated by a single transmission. An easy method of synchronization is provided by a variation of the broadcast command that waits for all activated scripts to finish before continuing. Additionally, Scratch allows for two distinct kind of variables. While global variables are accessible to all objects, sprite variables are only exposed to scripts running inside a specific sprite. Sometimes, broadcast is used in tandem with global variables to transfer information across sprites.

5. SCRATCHINTHECLUBHOUSE

6. In January of 2005, we brought Scratch to a Computer Clubhouse in a South Central Los Angeles shop. One of the city's poorest neighborhoods sends its young people (8-18) to the Clubhouse, which provides services in many languages. After school, kids who sign up to be members of the Computer Clubhouse don't have to worry about paying

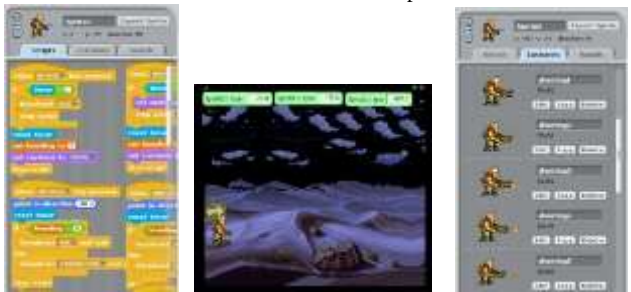
anything out of pocket [12]. Activities include playing Microsoft Xbox, downloading photographs, producing music at a studio, playing board games, editing photos in Adobe Photoshop, creating roller coaster games in RPG Maker, or developing 3D environments in Bryce 5 are all examples.

7. Despite the widespread availability of several kinds of programming tools prior to the introduction of Scratch, programming activities were not part of the Clubhouse array of activities [6]. Scratch became the most popular design tool at the Clubhouse during the first two years of the project, and local programming gurus developed during this time. There wasn't a lot of formal "teaching" of programming ideas; rather, kids worked on topics they were interested in and asked for help from adults only when they needed it. Every two or three months, we had a Scratch-a-thon when everyone in the clubhouse spent three to four hours using Scratch and then presented their creations to the rest of the group. Clubhouse participants used Scratch to create anything from video games and music videos to greeting cards and cartoons [9]. For instance, Kaylee, a female software designer of the ripe old age of 13, based her dance video "k2b" on a clip from a Gwen Stefani music video titled "Hollaback Girl" (see Figure 2).

Figure2:"k2b"Scratch program

The video game "Metal Slug: Hell Zone X," developed by Jorge (see Figure 3). In the center is a snapshot of an in-game character's avatar. On the

To the left is a screenshot of some of the code that determines how one of the avatars behaves. An example of a brief series of



still frames used to animate a shooting scene is shown on the right, which is a cropped screenshot of the outfits section.

To measure how deeply programming ideas were permeating the Clubhouse culture over time, we gathered young people's Scratch projects on a weekly basis for the first 18 months after their introduction. For our studies, we relied on three distinct data sets: (2) weekly participant field notes written by a team of Undergraduate and Graduate field researchers who visited the Computer Clubhouse and supported Scratch activities at the fiftieth-grade level. The exported project summary files included textual information such as the date, file name, author of the project, and the number and types of commands used, as well as the total number of stacks, sounds, and costumes used. Be aware that the Undergraduate and Graduate support staff were not computer scientists; rather, they were tasked with modeling the best practices of learning. The volunteers were Scratch newbies with little programming experience [5]. We saw this as a way for the mentees to gain confidence and independence, since they were able to take on the position of teacher on occasion.

Figure3:"MetalSlugHellZoneX"Scratchprogram.

8. PROGRAMMINGCONCEPTS

We were able to gather and analyze data from 536 projects, or 34% of the total number of projects completed at the Computer Clubhouse throughout the time period of this research. Scratch was used more often than Microsoft Word or any other program designed for creating multimedia. Over eighty kids, roughly split between boys and girls, utilized Scratch to make their own programs, with many of them working on a single project for months or even a year. These results show how Scratch has been widely used in the local Computer Clubhouse. Moreover, it is one of the few programming efforts that has effectively engaged young men and women of color.

By analyzing the gathered projects for common Scratch instructions, we were able to gain a sense of the programming ideas that these young people had acquired. We saw certain building elements as emblematic of the presence of a particular idea in a specific undertaking. Figure 4 depicts the script for a basic paddle game, in which the player uses the mouse to move a paddle and try to catch a ball that drops from the top of the screen. This script employs sequential control flow, a loop, conditional statements, variables, and random numbers. User input (the paddle) and control are also crucial to the game's design.follows the x coordinate of the mouse pointer) and needles (the paddle has its own script that runs in parallel with the ball script).

Only 113 of the 536 projects have any scripts at all. These "pre-scripting" examples show how Scratch may be used for straightforward media

production and modification. When first starting out with Scratch, many beginners spend time importing or sketching graphics and recording noises before going on to coding. Each of the remaining 425 projects uses sequential execution (i.e. a stack with several blocks) and the vast majority (374 projects, 88%) exhibits thread use (i.e. multiple scripts running in parallel). These are fundamental ideas in computer programming that any Scratch user must learn before they can begin developing their first programs. In addition to the above mentioned topics, we covered Boolean Logic (and, or, and not), Variables, Random Numbers, User Interaction (keyboard and mouse input), Loops, Conditional Statements, Communication and Synchronization (broadcast and when-receive), and more. These ideas are not as universally applicable as sequential processing. Without a loop or conditional, it is possible to create a basic application in which the user navigates a sprite using the arrow keys.

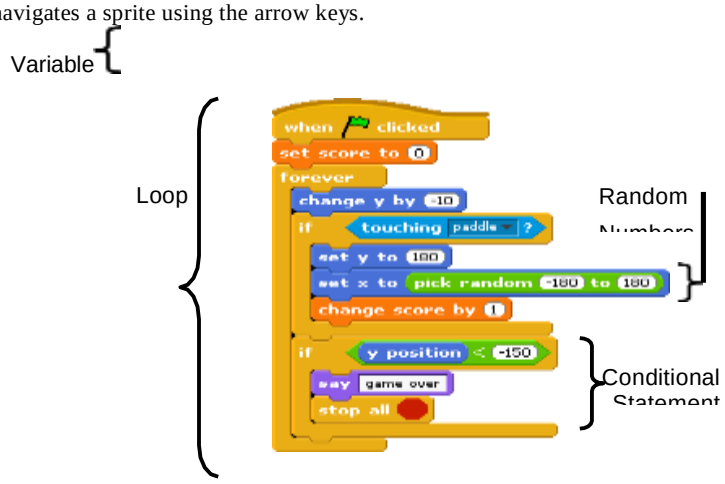


Figure4:ScratchScriptforBallinaSimplePaddleGame

The purpose of this research was to examine how often these topics were discussed by young people and to see whether there was an overall growth in the community's understanding of computer programming over time. The main ideas of programming are summed up in Table 1. It's hardly surprising that games and animations inspired so many projects to include user interaction and looping. To my delight, I also found that the Communication and Synchronization commands were used often; although being one of Scratch's more advanced concepts, inter-object communications is essential for constructing programs with several interdependent parts. However, the ideas of Boolean logic, variables, and random numbers are not simple to learn on one's own. The variables were essential to one user's work. While visiting the Clubhouse, he asked Mitchel Resnick for guidance, and Mitchel patiently explained how to utilize variables to solve his difficulties.

There was also a look at long-term tendencies. Overall, the number of completed projects during the second school year was twice that of the first school year over the same time period. (We wrapped up this phase of data gathering in the middle of the second academic year.) According to our research, when we compared the

When we looked at the proportion of projects from each year that included each programming idea, we saw that, on average, students were using more advanced concepts by the end of the second year (p .05). Variables, Boolean logic, and random numbers were some of the less intuitive ideas mentioned. In order to compare the proportions of projects from Year 1 and Year 2 that included the intended programming techniques, Chi-Square tests were conducted (see Figure 5). A majority of students' projects used the four targeted programming principles (Loops, Boolean Logic, Variables, and Random Numbers) after being exposed to the lessons (p .001). Among the remaining ideas, Conditional Statements showed minor improvement (p =.051), but the use of the idea of communication and synchronization dropped significantly.

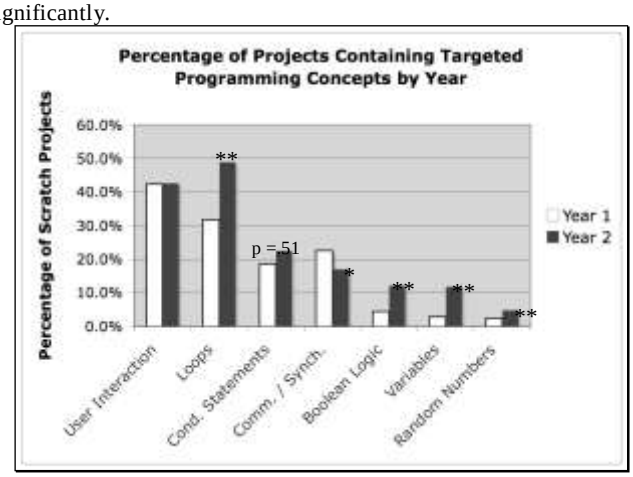


Figure5.Graph demonstratingthechangeinthepercentageofprojectsthatused variousprogrammingconceptsovertime

****p <.001*p <.05**

9. YOUTHIDEASOFPROGRAMMING

How did the young people in this research describe their own lives? Thirty members of the Clubhouse were surveyed to get their perspectives on programming and Scratch. When asked, "If Scratch had to be anything not on the computer, what would it be?" they were forced to come up with an answer. Most people (n = 8) said they would compare Scratch to a piece of paper or a sketchbook because you can "make whatever that you want with it, exactly like paper." There were also responses from people who compared Scratch to

in Flash, where he longed for a timeline to animate.

We asked kids a bunch of free-form questions to figure out where Scratch fit into their toolkit at home, at school, and at the Clubhouse. When asked whether their experiences in Scratch reminded them of anything from school, all of the kids claimed that it was similar to at least one topic, and most of them said that it was similar to numerous. The arts were the most popular choice overall (n = 20), followed by language arts (especially reading) (n = 10), then mathematics (n = 8), science (n = 5), history/social studies (n = 3), and finally computer science (n = 2). When asked how Scratch is related to other forms of art, the kids listed visual arts (n = 11), performing arts (n = 6), musical instruments (n = 4), and dancing (n = 3). From these comments, we were able to deduce that students associated Scratch most closely with classes that encouraged them to express themselves creatively and personally, such as art and language classes.

In our survey, most kids and teens didn't even realize that Scratch scripting was programming. In a survey of teenagers, the most common response to the question, "What is computer

programming to you? " was "Computer programming?" I have absolutely no idea what it is! At first, we were worried that kids wouldn't see the relationship between Scratch and computer programming. Scratch's success may be partially attributed to the fact that it is not seen as "programming," which allowed youngsters to perceive it as fitting in with their identities as kids, as something "cool," and as a fundamental element of the Computer Clubhouse culture. After all, teaching young people to code is not about turning them all become hackers or programmers; rather, it is about providing them with the 21st-century educational right to be exposed to the whole spectrum of technological fluencies. This is especially crucial considering that more over 90% of Clubhouse youngsters have never taken a single computer lesson throughout their time in elementary, middle, or high school. As a result, the Clubhouse becomes a pivotal location for easy access to software development resources.

10. DISCUSSION

Our research indicates that young people living in urban areas who attend a Computer Clubhouse are committed to learning computer programming. Children in the Clubhouse learned and applied ideas of user interaction, loops, conditionals, communication, and synchronization via their own initiative and exploration. Variables, Boolean logic, and random numbers, which are harder to grasp, saw gradual increases in use throughout time. Given the absence of any formal training or the mentors' lack of expertise in the field, these results come as a bit of a shock.

Absolute value and square root were never used in any of the projects. This makes sense, given that the sorts of projects for which such calculations are necessary are quite uncommon. Other notions, though, like as variables and random numbers, are very helpful, but they were slower to catch on. We assume these are not notions that can be figured out on one's own. The idea may have been lifted wholesale from one of the preloaded Scratch examples. Variables, however, are a notion that is present in both example projects and the real world.

that the community would not start using the concept unless a mentor came at the right moment and provided the necessary guidance.

Clubhouse kids could have used any number of other programming languages, so it's natural to wonder what drew them to Scratch. By "simplifying the mechanics of programming," "offering assistance for learners," and "giving students with desire to learn to program," Kelleher and Pausch [7] may have given the greatest response to this question (p. 131). We believe Scratch caters to these three needs. Scratch's block architecture, for one, simplifies the mechanics of programming by removing the need to worry about syntax problems, offering guidance on where to insert command blocks, and presenting instantaneous results from experimentation.

We also believe that the Computer Clubhouse's social infrastructure plays a significant role in helping newcomers to the programming world. All of the mentors were liberal arts majors with no programming expertise, but they were eager to give the young people advice and support them in their efforts. In many cases, we saw young people asking their mentors to work on their projects with them or at least provide feedback. Clubhouse kids would sometimes show their mentors the ropes of Scratch by sharing what they had learnt. Despite the common perception that mentors know more than their mentees, we found the reverse to be true in this case, with both parties benefiting from the learning experience [5]. This need for feedback and support may also account for the rapid growth of the Scratch website (scratch.mit.edu), which lets programmers post their work and collaborate with others.

Finally, we believe that the multimedia capabilities of Scratch contributed to the increased interest in programming among urban kids. The project archive showed that Clubhouse regulars were well-versed in a number of different types of media and eager to both consume and experiment with those forms. Numerous examples of Scratch projects use well-known characters whose photos were taken directly from the internet. Scratch projects centered on generic characters were more likely to be abandoned than those centered on popular characters, according to our analysis of data from other sources [10]. Digital media is often the entry point for young people who are curious about technology and might thus provide a more viable route for them to pursue careers in programming. The variety of media designs, from video games to music videos and greeting cards, shows that young people are interested in more than only consuming digital media (which they do on a daily and personal basis), but also in producing it, a role that is sometimes denied to urban youth.

.

11. REFERENCES

[1] Based on the work of [1] Adams, J. C. (2007). Alice, middle schoolers, and the fantasy literature debate. The 38th Annual Proceedings of the International Conference on Software Engineering

[2] Workshop on Computer Science Instructional Technology (pp. 307-311). To be published by ACM Press in New York.

[3] Begel, A. [2] (1996). An Introduction to LogoBlocks, a Visual Language for Creating Interactive Applications. The MIT Media Lab advanced student project report was never released to the public.

[4] According to [3] Goode, J., R. Estrella, and J. Margolis (2006). The gender gap in introductory computer science courses at the high school level. In J.

[5] Women in Information Technology: Perspectives on Underrepresentation, edited by M. Cohoon and W. Aspray (pp. 89-114). MIT Press, Cambridge, MA.

[6]

[7] In [4] Guzdial, M. (2004). The first steps in programming environments. Research on Computer Science Education, Edited by S. Fincher and M. Petre (pp. 127-154). Taylor & Francis, Libros, Lisse, Nederlands.

[8]

[9] Y. B. Kafai; S. Desai; K. Peppler; G. Chiu; J. Moya; and J. (in press). Fostering Equitable Service-Learning via Mentoring Partnerships in a Community Technology Center. Helping others learn by example and receiving individual instruction.

[10]

[11] Based on the work of Kafai, Peppler, and Chiu ([6]), (2007). Community Technology Centers and Their Role in Bringing High-Tech Programmers to Low-Wealth Neighborhoods. Proceedings of Communities and Technologies 2007, edited by C. Steinfield, B. Pentland, M. Ackerman, and N. Contractor (pp. 545-564). Springer, New York.

[12]

[13] According to [7] Kelleher and Pausch (2005). Reducing the learning curve for new programmers: a classification of introductory programming environments and languages. Publication information: ACM Computing Surveys, 37(2), 88-137.

[14]

[15] Papert, S. [8] (1980). 'Mindstorms,' by Nicholas Carr, Basic Books, New York.

[16]

[17] K. Peppler and Y. B. Kafai (9) (2007). Digital media creation in informal education: a look at tools from SuperGoo to Scratch. 32(2), pages 149-166 in Learning, Media, and Technology.

[18]

[19] As cited in [10] Peppler, K., and Kafai, Y. B. (under review). Media arts production as an intersection of technical, artistic, and critical processes; the creative bytes. The Learning Sciences Journal.

[20]

[21] Resnick, Michael; Kafai, Yasu; and Maeda, Jun. (2003). ITR is an after-school program that uses a media-rich, networked programming environment to increase students' comfort with technology. Submittal to the National Science Foundation, Washington, DC (which was funded).

[22]

[28] .

[23] A. Steinmetz, J. (2001). Learning Environments in Computers and Squeak. Squeak: Open Personal Computing and Multimedia, edited by K. Rose and M. Guzdial, pages 453482. Located in New York, Prentice Hall.

[24]

[25] Source: [13] Resnick, M., N. Rusk, and S. Cooke (1998). Urban youth get computer literacy at a clubhouse. Published in High technology and low-income communities, edited by D. Schon, B. Sanyal, and W. Mitchell. To be published by the MIT Press in Cambridge, Massachusetts.

[26] In [14] Sosniak, L. (2001). The Nine Percent Problem: Public and Private Education. 103 of the Teachers College Record.

[27] As cited in [15] Werner LL, Campe S, and Denner J. (2005). Girls in middle school + games programming = IT savvy. Sixth International Conference on Education Through ICT (SIGITE '05) Proceedings (pp. 301-305). To be published by ACM Press in New York