Peppler, K. (2010). The New Fundamentals: Introducing Computation into Arts Education. In E. P. Clapp & M. J. Bellino (Eds.) *20Under40: Reinventing the Arts and Arts Education for the 21st Century*.

The New Fundamentals: Introducing Computation into Arts Education

Kylie A. Peppler

Indiana University

Abstract. Given the advent of digital experimentation in the arts, this chapter conceptualizes the role that media arts can play in educational settings by looking to the ways that professional artists manipulate digital media. This chapter argues that learning to creatively code constitutes the new fundamentals of arts education in a digital world. The chapter presents a survey of contemporary projects that use computation as a way to manipulate the medium of the computer, outlines core-programming concepts for the novice reader, and showcases what inner-city youth are already creating through the use of computer programming.

The arts are incorporating new domains and artistic genres at impressive speeds, enabled largely by contemporary artists' use of new digital technologies as tools for communication, distribution, and creativity—constituting a field that might be broadly described as the new media arts or digital arts. With the advent of digital experimentation in the arts, new technologies are increasingly being introduced in higher education as well as K-12 arts classrooms. As a sign of the times, large schooling districts like Los Angeles Unified Schooling District (LAUSD) are already adopting media arts as a new official content area. However, current proposals for the media arts curriculum do little to teach youth the fundamentals of the new emerging field—instead focusing on introducing youth to video, sound, and image editing. While manipulating digital media is important, current proposals largely overlook the role of computation or computer programming in the K-12 media arts curriculum. By contrast, post-secondary media arts programs allot a great deal of time to learning about computer programming, considering it one of the core fundamentals of expression in digital media.

This chapter conceptualizes the role that media arts can play in educational settings, looking to the ways that professional artists utilize digital media and the methods of preparation that can be used to foster related skills in K-12 classrooms. Specifically, the practice of manipulating the computer through the act of programming deserves special attention. Often perceived as a narrowly technical skill with applications only to math and science, the ability to computer program is a fundamental proficiency of today's media artists, many of which use programming languages—artificial languages, like Java, C++, Processing, or Arduino, designed to control the medium of the computer—to creatively code unique and expressive work. Additionally, as evidenced by burgeoning online communities of tween/teen game designers, animators, and digital artists, learning to code creatively is becoming to today's generation what learning to read and write was to those growing up in the 20th Century. This type of creativity with technology is at the core of what media artists are able to do with new media and can, and should, be made accessible to even the youngest media artists.

This chapter argues that learning to creatively code constitutes one of the new fundamentals of arts education in a digital world. As evidence, a survey of contemporary tools and youth-designed projects is presented here that use computation as a way to manipulate the medium of the computer, followed by an outline of core programming concepts for the novice reader with examples of how a group of inner-city youth among others are already creating their own software and tools in much the same way that professional media artists go about this task—through the use of computer programming. This chapter further uses examples from youth engaged in one particular media arts software, Scratch, as a way of illustrating the vital role of computation in new media.[1] While several tools exist to help young artists learn to creatively code, including Alice, Logo, StarLogo TNG, and

others, this chapter focuses on the tool, Scratch, because of its widespread popularity and its use in and outside of the schooling system, making it a tool that is amenable to communities interested in learning to creatively code for the first time. Scratch is a media-rich programming environment that allows young designers to learn and use essential programming skills while also learning how to expressively manipulate the medium of the computer.[2] Furthermore, Scratch is one of the few tools that allows designers to pull in almost any image or sound file into a project, which presents strong opportunities to connect to youth's cultural interests as well as opens up their artistic palette for a greater degree of expression.

The New Media Arts: An Introduction

Art that makes use of electronic equipment, computation, and new communication technologies comprises the emergent field of "media arts" (also called digital arts or new media). Media arts, because it is a new field that is still being defined, can be situated as being distinct from traditional disciplines (such as visual art, music, dance, and theatre) but includes some overlap that arts educators may recognize, such as visual arts, animation, film, and, at times, electronic music. Because of its complex history, the field of media arts shares concepts and terminology with a number of other fields, including the sciences (e.g., gravity, mass, and acceleration), animation (e.g., tweening and motion paths), visual arts (e.g., color, perspective, and shape), and film (e.g., vocal intonation, visual style, and direction).[3, 4] In this sense, media arts could be described as a "meta-medium" that spans many different types of artistic practice. Central to most work in new media, however, is the role that computer programming plays in allowing the artist to have a fuller range of control over the artistic product.

Many educators may initially feel as if they need to choose between engaging youth in media arts or traditional arts as the central focus for their arts education programs. Arguably, this is not the case. Conceptually, this tension is important to recognize when designing a K-12 arts education program, suggesting that we don't need to think about replacing traditional arts education or even subsuming some of the goals of the arts to new media, rather to think about media arts and its techniques, skills, and concepts as building on and extending traditional concepts and ideas into a new medium—one that is already highly valued by youth, as evidenced by the proliferation of youths' digital expressions of creativity in online spaces. In fact, media arts at its core is still about pursuing aesthetic sensibilities in a digital medium but controlling other aspects of experience that can only be made possible through computer programming. However, as we begin to incorporate this new discipline into the schooling curriculum, it's important to rethink the K-12 arts education curriculum with particular attention to how we define the foundational practices of the discipline.

Creative Coding

In order to control various aspects of the interactive medium of the computer, artists have turned to applications that allow them to manipulate the medium of the computer. While a large number of programming tools have been around since the inception of the computer, few of these environments have traditionally appealed to the visual artist. This has been in part because prior languages were difficult for novices to learn and because they did not appeal to the visual nature of artists. As a result, media artists have introduced new tools into the landscape that have reshaped the fundamentals of artistic practices in new media. Computer programming, in this context, is another tool that has been added to the palette for artists. In the milieu of media arts, learning to computer program is often an important component of becoming "software literate" or having the ability to create novel user interfaces with the computer. Media artist, Casey Reas argues, "software is the medium that controls this flow of bits traversing the air and surface of our planet. Understanding software and its impact on culture is a basis for understanding and contributing to contemporary society."[5] And, yet, this capability is controlled by the few who understand and use programming languages to design software. Using the societal implications of widespread literacy (in the traditional form of reading and writing), Reas argues for the potential of technological literacy on a societal scale, and the reasons why programming should be a central component of media arts education today.[6] This overlaps with what researcher, Brian Smith would describe as "computational flexibility."[7] Being computationally flexible builds upon literate practices involving knowing how to use computationally rich software (e.g., word processors, spreadsheets, and presentation tools) as well as develop fluency (i.e., knowing how and why existing tools do not meet current needs), but extends this to include the ability to create the tools that one can otherwise only imagine. This type of creativity with technology is at the core of what media artists are able to do with new media.

Much of the practice of computer programming in the context of the arts begins with an intention (be it visual, audio, gestural, or interactional) and then seeking out commands or groups of commands that help to realize this idea, often letting the two co-evolve. This stands in stark contrast to the way many computer science classes are typically taught, where new programming commands are presented in the contexts of math or science problems that are introduced by the instructor. While most popular conceptions of computer programming involve users typing lines of equations and formulae onto an empty screen, numerous visual programming languages scaffold the programming environment with graphical interfaces that assist in choice-making and debugging. Especially considering the development of visual programming environments, youth don't need to gain an in-depth proficiency in computer programming before they can produce media art for the first time. The field has produced several

shortcut tools that allow youth (and adults alike) to use programming concepts such as loops, conditionals, data types, and numerical representations in a way that is more conducive to visual artists and novice programmers.[8] There are a host of new tools available in today's landscape to aid artists of all ages in producing media art. One such popular tool is Processing, created by Ben Fry and Casey Reas. Processing is an open-source programming language and environment for people who want to program images, animation, and interactions. It is used by students, artists, designers, researchers, and hobbyists for learning, prototyping, and production. Processing was created to teach the fundamentals of computer programming within a visual context and to serve as a software sketchbook and professional production tool. Other popular programming environments, like Arduino, enable users to create programs on a computer to facilitate interaction with an external object, like a robot or a machine, for those that wish to take digital media beyond the screen. While these tools target professional artists and the DIY crowd, other tools are available that are more amenable to the K-12 context, including Alice, LEGO Mindstorms, and StarLogoTNG, amongst others.[9]

The technology in focus here, Scratch, represents a particularly engaging medium for K-12 programmers. Due to its media-rich capabilities and novice-friendly design, Scratch has the potential to impact youths' arts learning in a number of ways. Additional incentive to examine the use of Scratch more closely concerns the program's immense popularity in both in-school and out-of-school settings. Translated into 50 languages, as of mid-2010 over a million Scratch projects have been uploaded to the www.scratch.mit.edu community by over 500,000 members. Scratch is the only programming language to utilize a building block command structure, saving users the burden of memorizing bits of code to program and instead providing several pages of commands that users drag to a central screen to control objects or characters (see Figure 1).[10] Being a media-rich environment, objects can include any graphic image either created in the program's paint editor or imported from a file. Designers can also incorporate existing sound or video files, as well as integrate other input/output devices into new design projects. Youths' projects run the gamut of incarnations, from art objects to animated stories. These projects can run uninterrupted in their entirety (e.g., like a music video) or can require the user to interact with the pieces through keystrokes (e.g., like a videogame).

Many of the widely available tools used to introduce programming concepts to youth tend to only allow creators a limited number of choices in their creations, placing restrictions on the amount of creativity and flexibility afforded by the software. This “platform approach” to media art making restricts youth unnecessarily. By contrast, this chapter's interest is in advocating software that allows for a great deal of flexibility, enabling users

to express themselves in a wide variety of modalities even when basic rules of coding apply across all modes.

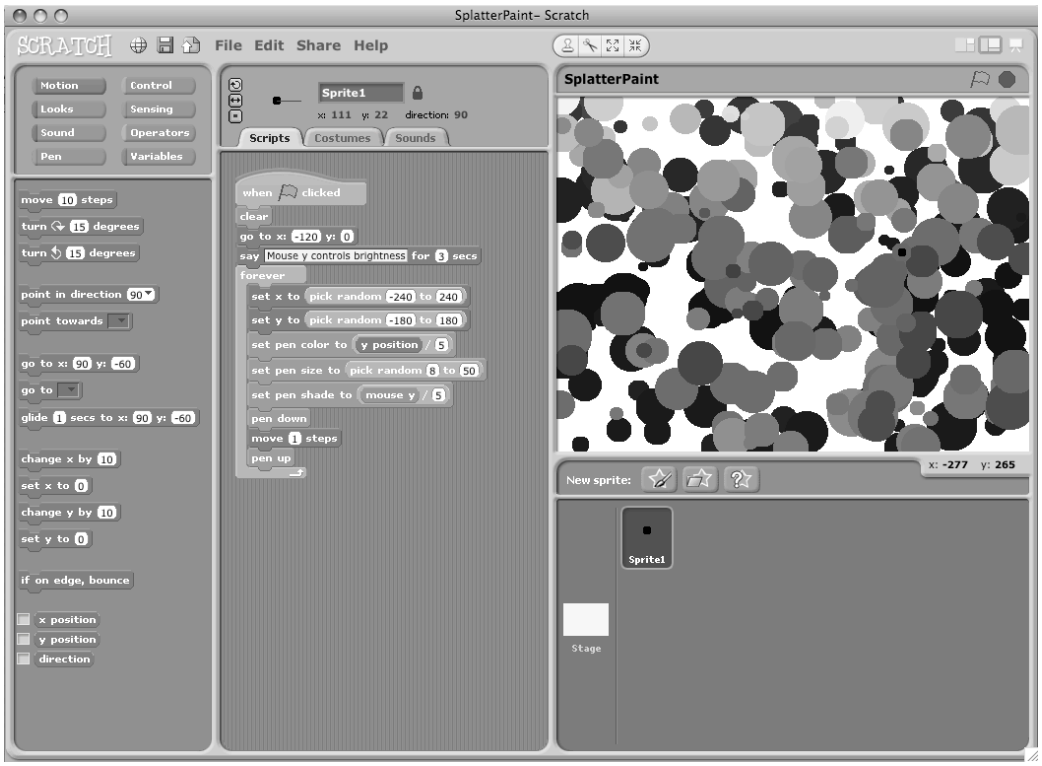

Figure 1. Screenshot of the Scratch user interface

To illustrate how coding can begin to take on various creative forms, a range of projects are presented below, including an interactive dance video, videogame, and animated art. This work was created within the context of a design studio found within the Computer Clubhouse community in South Los Angeles.[11] The Computer Clubhouse is a community technology center that moves beyond notions of a computer lab to engage youth in an after-school digital arts studio setting where youth can use applications that encourage skills beyond typing and web browsing, allowing participants to more deeply invest in the process of designing and critiquing their work. The data presented here was collected as part of a three-year ethnographic study researching literacy and learning in the media arts practices of urban youth.

Making Interactive Art, Dance Videos, and Lowriders Using Creative Code

To help dispel the myth that harnessing the medium of the computer using tools like those described above would be inaccessible to all but top-level students, the projects examined here were made by a mix of young artists

varying in age, gender, and ability. The range of expertise demonstrated in the coding underscores that users along the programming spectrum can not only create personally meaningful work using technology that accommodates varied ability levels, but that those rewards can even be realized early on in the process. Across these cases, however, all young designers utilized programming in service of realizing a higher aesthetic aim, demonstrating youths' envisioning of computation as a type of tool at an artist's disposal and not merely a computer science exercise.

A case in point can be found in a piece of media artwork created by an emergent reader and writer named Brandy who was 9 years-old.[12] Brandy created the piece, titled "Ranbow" [*sic*], over the course of a couple of days. Interestingly, Brandy, a special education student, learned to computer program before she learned to read and write, relying on a combination of mentor support and self-guided help resources to assist in her navigation of the programming environment.[13] "Ranbow" features a picture of an eight ball, a toaster, flowers, and a clip art image of berries all on a plain white backdrop (see Figure 2). At the click of a button, the images change colors at dizzying speeds. This piece was intended to be a birthday card for the artist's cousin and is particularly interesting because, though the designer was unable to read or write, she was able to successfully tie together several different modes of communication (images and animation) in order to create a personally meaningful and expressive project using a visual programming language. While she had pulled together several images that were a collection of memories and objects shared with her cousin, she extended the expressive capabilities of the art work beyond those of a static collage by using programming commands to animate the onscreen objects—this was in an effort, so said the artist, to make the piece seem like a "party."[14]

“Ranbow” illustrates what Mitchel Resnick would call the "low floors" of learning to creatively code.[15] Resnick argues that design tools that allow for novice users to create something for the first time have “low floors,” making it accessible to even the most novice user. He also argues that flexible software also allows for a wide variety of project genres (i.e., "wide walls") as well as "high ceilings" for youth wanting to push the upper boundaries of the medium. These are the type of attributes that we should look for in the 21st Century arts classroom to allow the greatest degree of expression for a wide spectrum of abilities—following much in the tradition of tools like paints, clay, and other media that have similar properties.

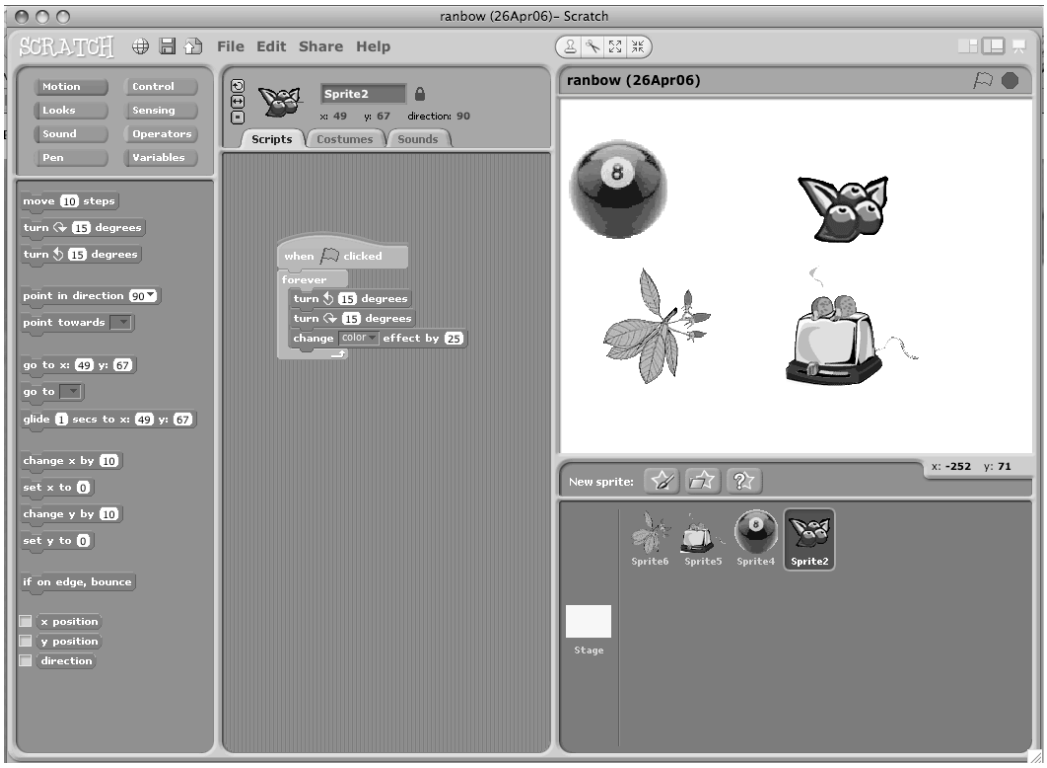

Figure 2. Screenshot of *Ranbow*

Figure three features a work of a 15 year-old Latina named Alejandra titled "M-Seventeen2." In the creation of this media art "dance video," Alejandra extended the mechanics of stop-action animation into a digital realm, as well as explores user interactivity in addition to synchronization between audio and image. Her piece features a highly stylized self-portrait that can either be controlled by the keystrokes of the viewer, or made to dance automatically in a pre-programmed series of gestures that move in time with an imported song, "Asereje" by Las Ketchup. "M-Seventeen2," with its vast collection of over 25 hand-drawn images created in the program's paint editor (as seen in the "costumes" column of figure 3), represents a more comprehensive combination of skills pulled from traditional arts (drawing), film (synchronization), as well as computer science (interactivity). Alejandra's project ties together bits of code with visual images so that the two seamlessly fit together. Additionally, she included two major branches in the logic flow, enabling the viewer to experience the art piece by either passively watching the pre-programmed animation sequence, or taking control of the keyboard to create their own sequence of "dance steps."

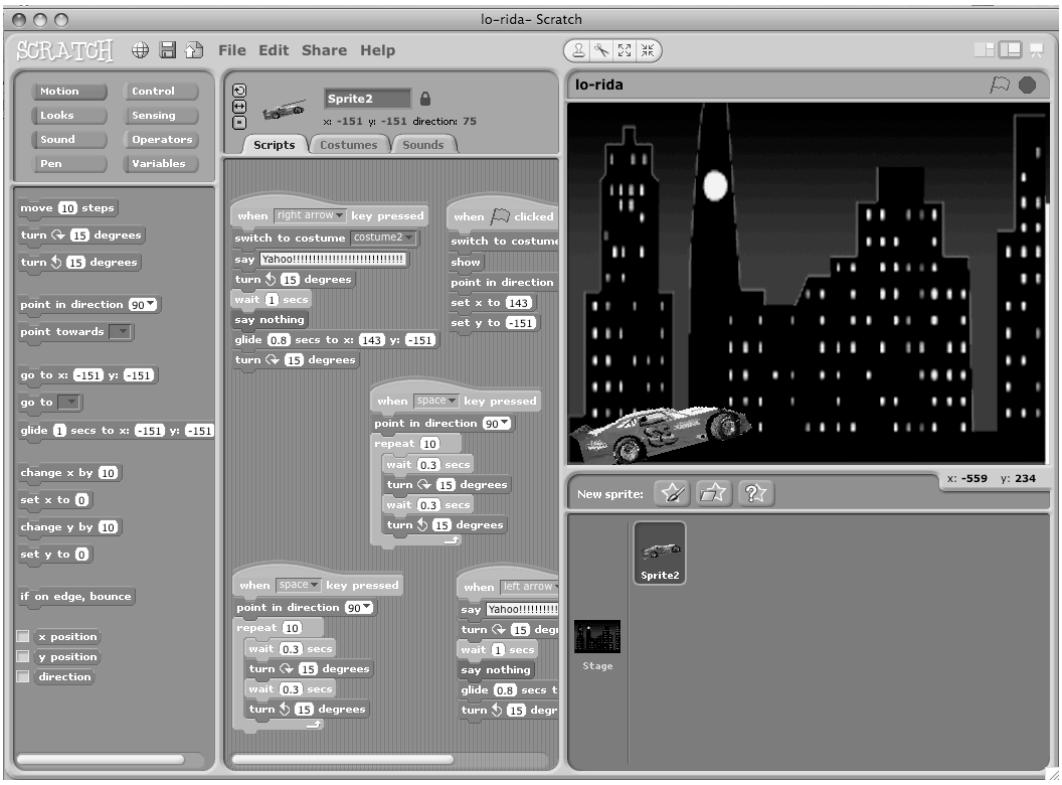

Figure 3. Screenshot of *lo-rida*

Beyond developing the ability to computer program, through the creation of this work, Alejandra learned about several aspects of digital media production, such as stop-action animation, in the aim of creating a realistic depiction of dance. The process of shaping this piece within a computer programming environment helped develop Alejandra's ability to sequence movement and separate this into individual steps. Learning to animate pushes youth to think about and abstractly model their perceptions of events, mostly with the goal of making them seem more realistic. Alejandra learned to render a single image as well as how to combine these images to simulate movement and gesture, each informing the other in a cyclical process of trial and error.

The next project, titled "lo-rida" by Alberto, a 12 year-old Latino male, is an example of a genre of the interactive art projects that originated at this site. Perhaps most importantly, the media-rich aspects of the Scratch technology fueled the cultural significance of the genre for this community of largely African-American and Latino youth. Inspired by the customized cars principally associated with the Mexican-American community (and made popular nationwide by media representations, such as MTV's *Pimp My Ride)*, designers would import their favorite images of lowrider cars into their projects, and customize them through paint editing and animation (coding).[16] Alberto's "lo-rida" project depicts a lowrider racecar in front of a city skyline.

In this project, the artist has created a degree of interactivity by allowing the hydraulics of the car to be controlled by the viewer's pressing of the arrow keys (see Figure 4).

Representing a mix of genres (perhaps of videogames, lowrider cars, and cartoons) that are interactive (by requiring the viewer to operate the hydraulics in most cases, similar to games or web-based art), performative (as showcasing your lowrider project within the community is particularly important), and artistic (as each member strives to customize their project), the lowrider arsenal of projects were seen within the youth community as meaningful, viable modes of expression that are inextricably linked to the designers' out-of-school identities. This type of mixing of the local context into new, interactive digital media creates new genres of work. Lowriders, pulling from the history of the car customization movement, represent a conscientious and literate practice that stands in opposition to the pressure to assimilate into the American mainstream culture. As educators plan to include a wider variety of genres in the arts education curriculum, it becomes important to question whose genres are being included and excluded in the curriculum. Rather than deciding on this *a priori*, creative production provides an exciting opportunity for youth to share out-of-school literacy practices with media educators so that these practices can be leveraged to meet the larger goals of education. Additionally, media-rich computer programming tools allow youth to invent their own genres of work rather than confine them to particular project types.

Youth Reflections on Their Media Art Practices

While it's clearly demonstrated above that youth exhibited a range of expressionate potential through their use of Scratch, we turn to a series of interviews with the Clubhouse youth to determine where they would situate their work in Scratch within the range of their prior academic and/or artistic experiences.[17] When asked whether creating media art in Scratch reminded the youth of anything at school, all of the youth said it was at least like one subject matter, and most cited more than one subject that they thought connected to their experiences. Overwhelmingly, youth associated their experiences in Scratch with their prior arts education experience, followed by language arts, and then far fewer with their experiences in mathematics, science, and computer class. When probed further about the connections to which art forms, youth cited a variety of answers, including drawing or sculpture, drama, dance, and music. The connections that youth made in all subject areas seemed to be dependent upon the extent of their experience in each subject area or art form. Further, when asked, “if Scratch had to be something not on the computer, what would it be?” the most common response was “paper” or “a sketchbook” because Scratch allows one to “do anything that you want with it, just like paper."[18] Based on these insights, it

appears that youth, while making connections to subject areas across the curriculum, see their work most in line with the arts—creating a natural home for this type of computer programming work. This is a thought-provoking finding given that, at face value, youth could be seen as merely learning to computer program and/or mixing existing media—areas that might be most well suited for computer science or media education courses. Instead, youth who engaged in computer programming saw themselves as artists, demonstrating the creative and communicative potential that work in a digital domain can have given the appropriate tools. Furthermore, most youth did not see their programming experiences as being connected at all to computer classes. This underscores the differences between media art making and narrowly technical experiences that the youth had in their computer classes, such as typing and direct instruction on word processing, digital presentation, and spreadsheet tools.

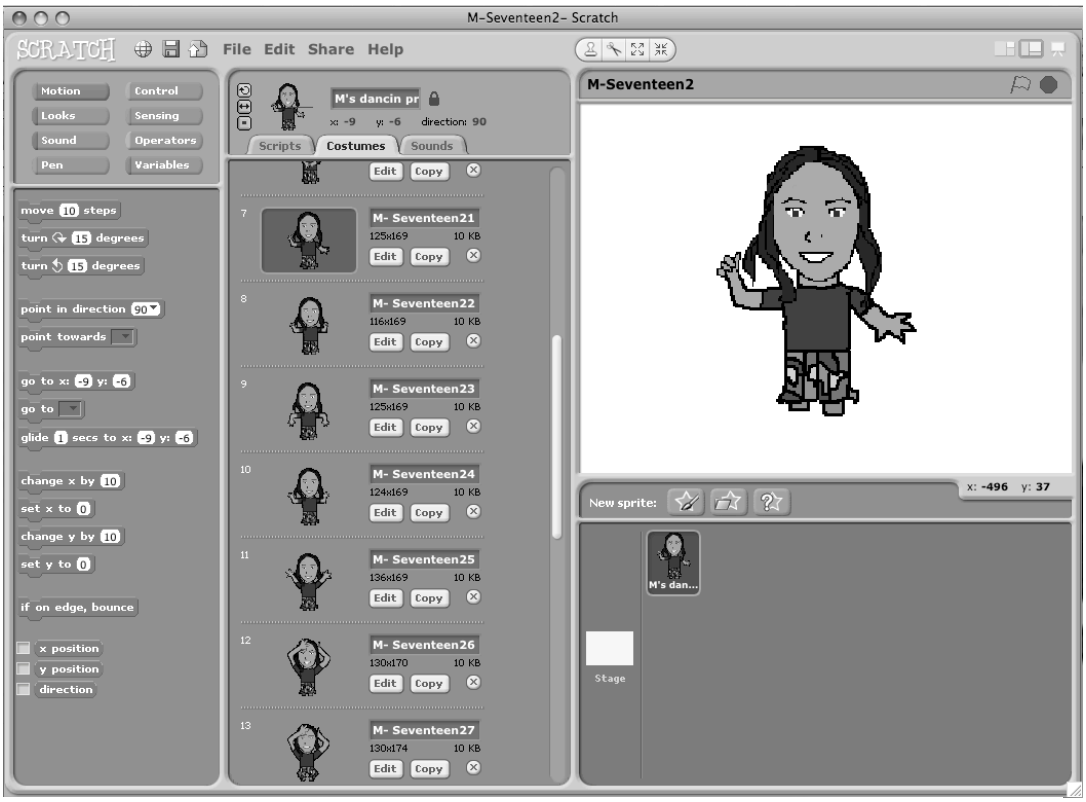

Figure 4. Screenshot of M-Seventeen

The New Fundamentals

When we step back and examine the curriculum of prestigious post-secondary programs in media arts, we begin to see that the courses spend approximately 1/3 of their time on computer programming, 1/3 on traditional visual aesthetics, and 1/3 on the intersection of programming of

aesthetic design. This ratio represents a guiding proportion for arts educators, one that emphasizes cultivating aesthetic awareness in addition to introducing a new fundamental of arts education, media arts—where programming (and not just digital media manipulation) plays a leading role. Learning to computer program in the context of the arts has a number of advantages, including moving away from the narrowly technical approach to computer programming to emphasizing the means or ends of programming to create a particular effect. Some, however, may question whether youth engaged in media art making will be learning the fundamentals of computer programming while engaging in tools like Scratch. Prior research, however, indicates that youth intuitively learn the "big ideas" of computer programming through media art making in Scratch over time and without explicit instruction.[19] The big ideas or fundamentals of programming concepts that confront every Scratch user when they begin writing scripts include: User Interaction (use of keyboard or mouse input), Loops, Conditional Statements, Communication and Synchronization (*broadcast* and *when-receive*), Boolean Logic (*and*, *or*, and *not*), Variables, and Random Numbers. For example, in the projects described above, youth utilized user interaction whenever they asked the viewer to press a key to initiate an action (e.g., key presses to control the hydraulics on the lowrider) and/or utilized loops when there were repeated actions (e.g., a quick sequence of colors to create a flashing effect in the Ranbow project). It's important to note that, in the current approach, youth utilized certain commands in service of the larger needs of their project and they gave their onscreen objects “powers” that they wouldn't otherwise have on paper. The youth became interested in meaningfully manipulating the code to create the desired effect—not to learn programming as an ends of itself.

To give the reader a better idea of how these fundamental programming concepts are manifested in Scratch, we can further analyze the use of command blocks in any particular project. For example, Figure 5 shows a script from a simple paddle game in which a ball falls from a random place along the top of the stage and must be caught by a paddle controlled by the mouse. This script uses the concepts of sequential control flow, a loop, conditional statements, variables, and random numbers. The overall game also uses the concepts of user interaction (the paddle tracks the x position of the mouse) and threads (the paddle has its own script that runs in parallel with the ball script). Over time, youth even in informal learning environments begin to use a larger number of programming commands as they seek to further control their on-screen objects. In the context of media arts, it's less important that youth can name such programming constructs but that they can use them in the context of their own work.

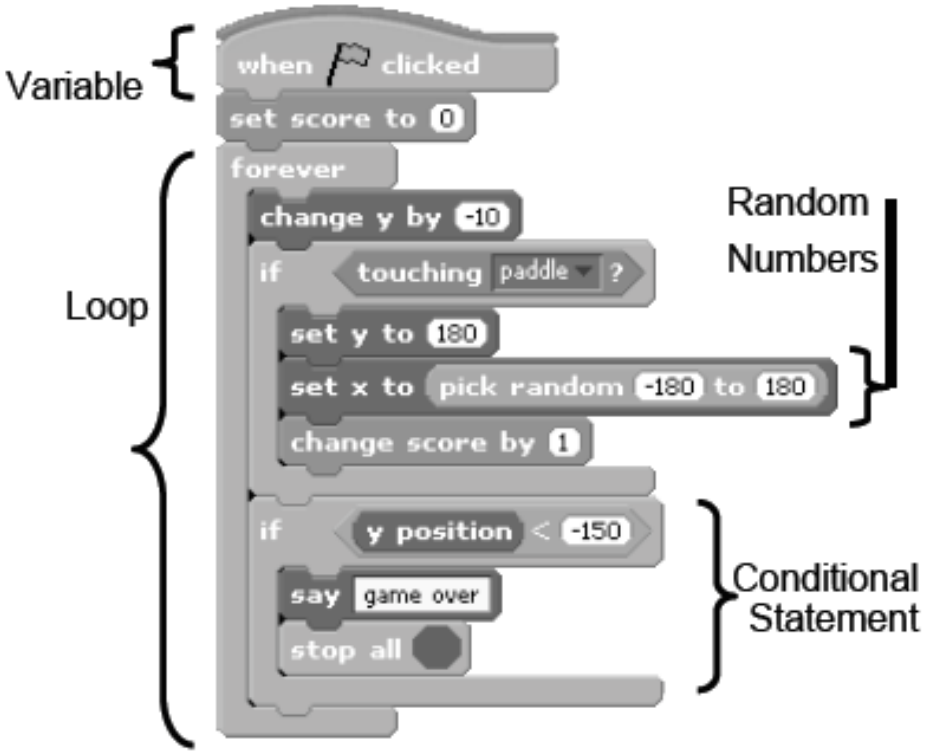

Figure 5. Stack of Sample Scratch Commands and the types of Programming Concepts

There are, additionally, instrumental and intrinsic benefits of creative coding. The instrumental benefits are the ways in which students transfer what they learn through using programs like Scratch into other domains or life skills. The intrinsic benefits of creative coding describe the benefits of learning to creatively code in and of itself. For the art educator wanting to incorporate creative coding into their arts curriculum, it is important to (a) have software like those mentioned above loaded on the classroom computers, (b) to become a member of one of these communities and to produce some work, themselves, even at a very novice level, (c) to introduce introductory projects into the curriculum that allow for a great deal of personalization (e.g., a "name" project, where students import letters of their name and use commands to animate each letter in unique ways); and (d) allow longer periods of time for youth to explore this media toward particular ends of their choosing. It has also been demonstrated in prior research that youth have access to popular media that can be incorporated in their projects, as it is key to prolonged engagement in projects across the spectrum of ages. This also connects art educators to the visual culture curriculum.

Conclusion

The advantages of this emphasis on the new fundamentals are at least

twofold: (1) computer programming is an essential component to artistic expression in a digital era—a tool that has an arguably increasing importance for youth and society at large; and (2) media art projects emphasize graphic, music, and video—media at the core of youths' technological interests—and thus could provide new opportunities to broaden participation of under-represented groups in the design and invention of new technologies.

Additionally, there are several reasons why arts educators, in particular, should be interested in the incorporation of the field of media arts into the schooling curriculum. Among others, current conceptions of schooling envision new technologies being integrated across the curriculum in all K-12 school subject areas in order to keep up with the demands of preparing youth for the 21st century. Rather, it is important to argue for an expansion of the curriculum to include teaching youth how to work with new technologies as an expressive medium, becoming software literate, and enabling youth to produce new tools and modes of communication. The arts perspective on new technologies is unique in the schooling landscape. Other curricula, such as technology or computer courses, have the tendency to focus on narrowly technical activities. Creative coding, on the other hand, offers an opportunity for youth to explore the full potential of the computer as an artistic medium and consider the implications of learning to communicate in a time when multimodal discourse is becoming increasingly important. Accordingly, the study holds implications for informing policy by placing central the need for arts in today's schools. In light of the increased mediation of our society, many feel that there is an urgent need for arts education to develop youths' ability to be critical about the messages that they receive and transmit. This study encourages and informs policy aimed at meeting technology fluency and creative thinking goals by emphasizing the critical role of producing one's own media texts in any arts education program.

Schools, then, can play an important role in this new landscape, as they are poised to address the limitations of ongoing free play in the after-school hours. Within schools there is an opportunity to systematically introduce core media arts concepts and go into greater depth that otherwise is not possible within after-school spaces. While youth make important discoveries through unstructured learning experiences like those fostered at the Computer Clubhouse, such environments are carefully constructed and are reliant on the availability of high-quality interactions with adults or more expert peers, which are oftentimes scarce resources. Arts educators, by contrast, can reliably provide access to high-quality dialogues during critiques with youth engaging in digital media, and potentially go into greater depth into a media arts curriculum. For educators fearing that the tools required to produce media art require too much training before any real projects can be realized, learning to computer program within the context of media arts does not necessarily involve an extensive and time-consuming

introduction and many helpful resources already exist that can be easily leveraged in the classroom. It's important to note that these efforts in introducing creative coding into the context of arts education are not geared towards replacing the traditional arts or turning all youth into programmers. Rather, learning the language of creative coding is essential to communicate in a digital age.

Kylie Peppler (Born 8-1-1979), Bloomington, IN

Kylie Peppler is an Assistant Professor in Learning Sciences at Indiana University, Bloomington. An artist by training, Kylie's research focuses on the intersection of arts education and new technologies. She has received grants from the National Science Foundation, the MacArthur Foundation, and the Spencer Foundation to support her work. Kylie's recent articles have or will appear in the *Cambridge Journal of Education*, *Teachers College Record*, and *Learning, Media, and Technology*. In 2009 Kylie's first book *The Computer Clubhouse: Creativity and Constructionism in Youth Communities* was published by Teachers College Press.

NOTES

1. Resnick, M., Maloney, J., Hernández, A. M., Rusk, N., Eastmond, E., Brennan, K., Millner, A. D., Rosenbaum, E., Silver, J., Silverman, B., & Kafai, Y. B. "Scratch: Programming for Everyone." *Communications of the ACM,* (forthcoming). The Scratch community was awarded the Eliot Pearson Award for Excellence in Children's Media in 2008, an Honorable Mention for the Scratch Community in the 2008 Prixars Electronica Competition, and the Kids@Play award for Best Informal Learning Experience in 2009.
2. Maloney, J., Peppler, K., Kafai, Y. B., Resnick, M., Rusk, N. *Programming by Choice: Urban Youth Learning Programming with Scratch*, (March, 2008). Proceedings published by the ACM Special Interest Group on Computer Science Education, Portland, OR.
3. Nalven, J., & Jarvis, J. *Going Digital: The Practice and Vision of Digital Artists.* Boston, MA: Thomson Course Technology, 2005; Paul, C. *Digital Art.* New York, NY: Thames & Hudson, 2003.
4. Ibid.
5. Reas, C. "Processing: Programming for the Media Arts." *AI & Society* 20, no. 4 (2006): 526-538.
6. Ibid.; Reas, C. "Media Literacy: Twenty-First Century Arts Education." *AI & Society* 20, no. 4 (2006): 444-445.
7. Smith, Brian K. "Design and Computational Flexibility." *Digital Creativity* 17, no. 2 (2006): 65-72.
8. Maeda, J. *Design By Numbers.* Cambridge: The MIT Press, 1999; Maeda, J. *Creative Code.* New York: Thames & Hudson Inc., 2004; Reas, C. "Processing: Programming for the Media Arts." *AI & Society* 20, no. 4 (2006): 526-538; Resnick, M., Maloney, J., Hernández, A. M., Rusk, N., Eastmond, E., Brennan, K., Millner, A. D., Rosenbaum, E., Silver, J., Silverman, B., & Kafai, Y. B. *Scratch: Programming for Everyone. Communications of the ACM,* (forthcoming).
9. For a complete history see Kafai, Y. B. "Constructionism," in K. Sawyer (Ed.), *Cambridge Handbook of the Learning Sciences.* Cambridge, MA: Cambridge University Press, 2006.
10. Guzdial, M. (2004). "Programming Environments for Novices," in Fincher, S. & Petre, M. (Eds.) *Computer Science Education Research*. London, UK: Taylor & Francis Group, plc., 2004: 127-154; Resnick, M., Maloney, J., Hernández, A. M., Rusk, N., Eastmond, E., Brennan, K., Millner, A. D., Rosenbaum, E., Silver, J., Silverman, B., & Kafai, Y. B. *Scratch: Programming for Everyone. Communications of the ACM,* (forthcoming).
11. Kafai, Y.B., Peppler, K., & Chapman, R. (Eds.) *The Computer Clubhouse: Creativity and Constructionism in Youth Communities.* New York, NY:

Teachers College Press, 2009.

12. Pseudonyms are used for all youth mentioned throughout this chapter.
13. Peppler, K. & Warschauer, M. "Lessons from Brandy: Creative Media Production by a Child with Cognitive (Dis)Abilities." Paper presentation at the American Educational Research Association (AERA) Annual Meeting, Denver, CO, April 30-May 4, 2010.
14. Ibid.
15. Resnick, M., Kafai, Y., & Maeda, J. "ITR: A Networked, Media-Rich Programming Environment to Enhance Technological Fluency at After-School Centers in Economically Disadvantaged Communities." Proposal submitted to National Science Foundation, 2003.
16. See Cowan 2004 for a fuller account of its emergence amongst migrant workers during World War II; Cowan, P. "Devils or Angels: Literacy and Discourse in Lowrider Culture," in Mahiri, J. (Ed.), *What They Don't Learn in School: Literacy in the Lives of Urban Youth.* Oxford & N.Y.: Peter Lang Publishing Company, 2004: 47-74.
17. Peppler, K. "Media Arts: Arts Education for a Digital Age." *Teachers College Record* 112, no. 8 (2010).
18. Maloney, J., Peppler, K., Kafai, Y. B., Resnick, M., Rusk, N. *Programming by Choice: Urban Youth Learning Programming with Scratch*, (March, 2008). Proceedings published by the ACM Special Interest Group on Computer Science Education, Portland, OR.
19. Ibid.