BILL BABBITT, DAN LYLES, & RON EGLASH

9. FROM ETHNOMATHEMATICS TO ETHNOCOMPUTING

Indigenous Algorithms in Traditional Context & Contemporary Simulation

Ethnomathematics faces two challenges: first, it must investigate the mathematical ideas in cultural practices that are often assumed to be unrelated to mathematics. Second, even if we are successful in finding this previously unrecognized mathematics, applying this to children's education may be difficult. In this essay, we will describe the use of computational media to help address both of these challenges. We refer to this approach as "ethnocomputing." Modeling is indeed an essential tool for ethnomathematics (Rosa & Orey, 2010). But when we create a model for a cultural artifact or practice, it is hard to know if we are capturing the right aspects; whether the model is accurately reflecting the mathematical ideas or practices of the artisan who made it, or imposing mathematical content external to the indigenous cognitive repertoire. If I find a village in which there is a chain hanging from posts, I can model that chain as a catenary curve. But I cannot attribute the knowledge of the catenary equation to the people who live in the village, just on the basis of that chain. Computational models are useful not only because they can simulate patterns, but also because they can provide insight into this crucial question of epistemological status.

Take, for example, our recent investigation of Navajo weaving. Figure 9.1 shows a common re-occurring angle in the weaving patterns, of about 30 degrees. When asked about this particular angle, the weaver said it is created by an "up one over one" pattern (up one weft over one warp). We could see why that would result in a 30-degree angle: because the height to width ratio of each successive stitch was about 1/3. But she then said she could do lots of other patterns (up one over two, up two over three, and so on). So we asked why this one is used so commonly. She explained that other angles gave a more jagged edge. In other words they were concerned about the aliasing problem (Fig. 9.2), a common feature in early computer graphics (and still a concern in certain situations such as bitmap images).

This implies a host of new questions: Do some weavers use anti-aliasing techniques similar to those strategies used in computer graphics, such as using an in-between color at the edges? What are the iterative algorithms for more complex shapes? Thus the ethnocomputing approach offers two advantages. First, although

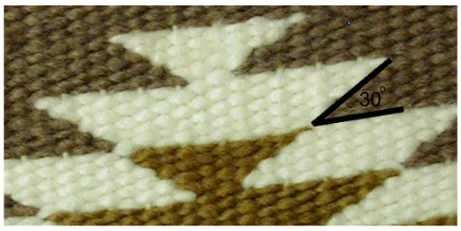


Fig. 9.1 Navajo blanket

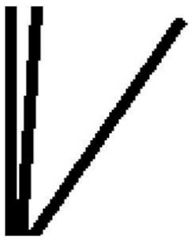


Fig. 9.2 Aliasing in computer graphics

we like to think of mathematics as being comprehensive in its ability to model patterns, some pattern generation systems are better conceptualized through the disciplinary idioms of computer science. Second, the conceptual framework of computing – the idea of information processing, algorithms, graphical user interface, etc. – allows new insight into the artisans' own perspectives in cases in which there is an analogous process. As noted above, there is a second challenge to ethnomathematics, and that is the challenge of converting these results into a pedagogical form suitable for pre-college classrooms; in particular for under-represented ethnic groups. Of course there are other applications for ethnomath, but education has been the most important. That is because of the possibility that cultural connections to math may improve the low performance and interest in mathematics that we tend to see in African American, Native American, Latino, and Pacific Islander children in the US. Several studies suggest that one factor for this low performance is that these children's identities are formed in opposition to the mainstream, such that doing well in math is considered "acting white" (Fordham, 1991; Fryer & Torelli, 2005). Another factor may be the myth of genetic determinism, in which children assume that their race lacks "the math gene" (Mathematical Sciences Education Board, 1989). Ethnomathematics could potentially offer a powerful counter to both the "acting white" myth and the genetic determinism myth. But merely pointing to a photograph of some intricate basket or monumental pyramid is not sufficient for engaging children or developing their mathematics skills. Here, too, computing can contribute.

In 2005, Anthony Lizardi, the chair of mathematics at Rough Rock, a school run by the Navajo Nation, invited us to develop a simulation for Navajo weaving. We

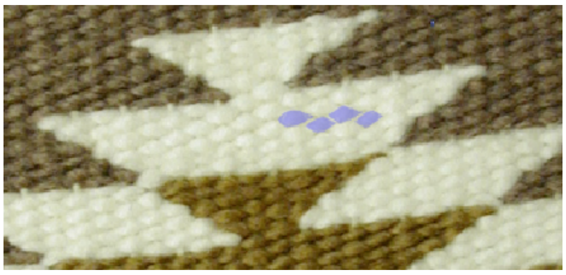


Fig. 9.3 Alternations in weft strands

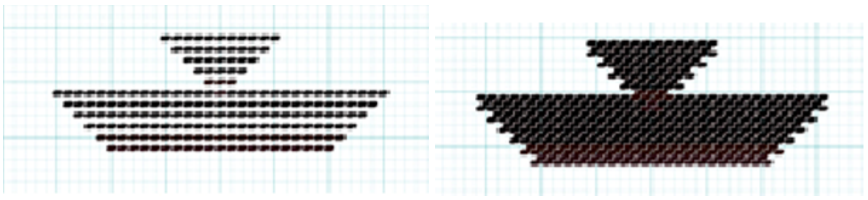


Fig. 9.4 Two steps in Navajo rug simulation

found that the weavers we spoke to were enthusiastic about this use of their work; there was some concern that future generations would lose interest in traditional weaving, and they saw this as an opportunity to maintain its relevance. They were also happy to discuss the various algorithms used to create particular shapes, which is how we arrived at the "up one over one" discussion above. However we ran into a problem with the details of the weaving process.

Upon examining the rugs we realized that would not be possible: the weft alternates up and down (see highlights in Fig. 9.3), because every other strand goes in back of the warp, and in the row above it the alternation is the opposite, with every other strand going in front of the warp. Since an important part of the weaving process compresses the rows together (the weaver pushes down with "comb"), the gaps (where the weft goes behind) from each row above and below are filled, thus producing an up and down alternation. Thus we could not simply map each individual weave to integer intersections on a Cartesian grid, which would be optimal for teaching purposes. One alternative was to simply map the weave into non-integer spaces between the grid intersections, but that would destroy much of its utility for the math teacher. Our solution was to simulate using two steps: first the user maps out a weave pattern using only the grid intersections (Fig. 9.4a), and then the user presses a comb icon, and the applet fills in the gaps (Fig. 9.4b), thus creating a completed weave simulation. Not only did this satisfy both the math teacher and our own interest in a good visual simulation, but it also better represented the actual process of weaving. Figure 9.5 shows some of the designs created by a group of Navajo high school students at the Diné College Pathways Summer Program in 2009.

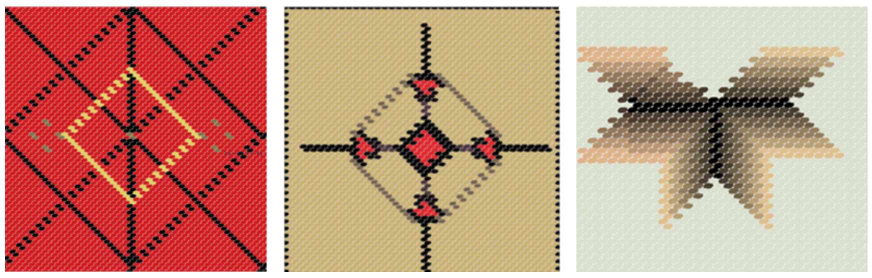


Fig. 9.5 Simulations by Navajo students

These "Culturally Situated Design Tools" (CSDTs) show statistically significant improvement in student performance, using controlled tests (Eglash, Bennett, O'Donnell, Jennings, & Cintorino, 2006; Eglash & Bennett, 2009). These web applets are available online (www.csdt.rpi.edu); the website for each "design tool" (allowing students to simulate cornrow hairstyles, Native American beadwork and weaving, Latino percussion rhythms, Mayan temples, urban graffiti and breakdance, etc.) includes some cultural background on its topic, as well as pages of lesson plans, evaluation instruments, etc.

In summary, we find the following three domains and interaction for ethnocomputing in the classroom:

– Simulations must "translate" from the particular indigenous or vernacular knowledge under investigation into the analogous knowledge forms contextualized for students in classrooms. For example, Native American traditions use the "four winds" or "four directions" as an organizing principle across many different knowledge systems: cosmology, religion, health, architecture, weaving, etc. (Eglash, 2009). This makes it an excellent candidate for teaching the Cartesian coordinate system using simulations of these artifacts such as bead work; we can clearly justify the four-quadrant coordinate system as an indigenous invention, and not merely a Western idea that is imposed on these artifacts (Barta & Eglash, 2009). At the same time, a pedagogy that introduces these artifacts should do so with the social context in which they arise. The cultural background pages for the "Virtual Beadloom" CSDT, for example, includes the use of Iroquois bead work in their US treaties and the influence of the Iroquois confederacy on the creation of the US constitution. In such instances we have attempted to steer a path between the Scylla of "white-washing" history (such that the horrors of exploitation and oppression are completely erased), and the Charybdis of a story of "victimhood" (which could demoralize students).
– These math and computing analogies are rarely exact; there is typically some negotiation between the "fidelity" of the simulation as an exact replica of the indigenous concept, and the utility of the simulation as a fit to the classroom curriculum. For example, weavers have to worry about fitting horizontal weft threads into the vertical warp without creating slack, hiding the ends when a

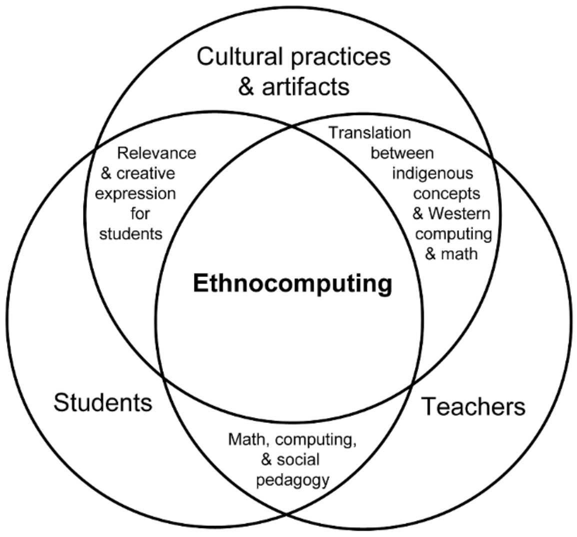


Fig. 9.6 Three domains in ethnocomputing

new color is started, etc. All of these activities might be modeled: for example, you could model the relations between two adjacent horizontal weft strands as 180 degrees out of phase. But that would complicate its use at lower grade levels where the concept of "phase" is not taught, and even at upper levels, being forced to think about phase while simultaneously using an iterative algorithm would make use potentially frustrating. In contrast to critics who complain that ethnomath adds too much external math to artifacts, the challenge in developing these simulations is to leave out much of the "high fidelity" modeling that would potentially be possible, in order to create a lower fidelity model that is both optimal for use and offers a clear translation of indigenous knowledge.

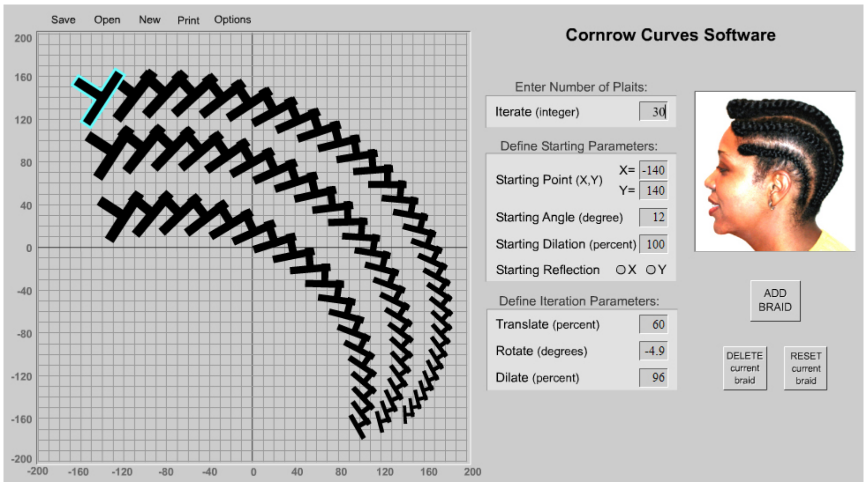
– In addition to attempting to negotiate the tension between fidelity to the indigenous conception and utility to the curriculum, a third tension exists when trying to satisfy student needs for relevance and creative initiative. For example, in our initial attempts to use fractal models of African artifacts (Eglash, 1999), we found that African American students occasionally expressed some hesitation over what were, for them, dusty museum objects. For this reason, our first simulation focused on cornrow hairstyles, which offered a compromise between African heritage and objects and practices familiar to them as part of contemporary African American culture. However, as our websites have developed, we have found that even for familiar practices (bead work in native American communities, graffiti among the urban "underclass," etc.) there is a need to teach these histories (where else are they going to learn about the history of graffiti?). Similarly, the ability to make creative use of these tools, and generate their own designs (some of which bear no resemblance to traditional examples) is critical for engaging these students, and encourages a sense of ownership over the mathematics. Moving from consumption to production, taking pride in self-efficacy and designs, learning to use math and computing as a means of self-expression rather than the disciplinary regime of "you got the wrong answer" – these are all critical components of ethnocomputing pedagogy. Figure 9.6 summarizes these three domains and their interactions.

FROM CSDTS TO PCSDTS

As noted above, our work with CSDTs made it clear that there is a component of ethnomathematics that has received little attention, because such "computational thinking" (Wing, 2006) is outside the purview of the standard math curriculum. Computing education is a key to the high-status skills and knowledge that allows a student to tap into the grid of twenty-first-century opportunities; one which under-represented students are often left out of. Would it be possible to use these CSDTs to teach computer science in primary or secondary school? In a recent publication (Eglash et al., 2011) we report on the use of our "African Fractals" CSDT (csdt.rpi.edu/african/African_Fractals/index.html) in a controlled study of two high school computer science classes with the same instructor. The control group received the same amount of instruction with a comparable fractal education website (it also used java-based applets) without any emphasis on cultural design. The results showed statistically significant advantages for the class use the African Fractals site in both performance and attitudes towards computing careers.

It was unclear, however, whether this effort to teach computing (as well as mathematics) could be applied to all CSDTs. Fractal geometry is a special case in that it is inherently mixing computing and mathematics. What about teaching conditionals, data structures, and algorithms? Such concepts were present in the CSDTs, but too deeply embedded in the tools. Take, for example, the "Cornrow Curves" simulation. Figure 9.7 shows the CSDT control panel and resulting simulation for three braids. The photo at right is one of many "goal images" that students can attempt to simulate. At left is the simulation. The left-most plait of the top braid is high-lighted to indicate that the numbers in the control panel refer to that braid. The simulation uses a recursive loop in which the original plait image is duplicated, and then geometric transformations are applied to the duplicated plait. This cycle is repeated, duplicating the previous duplication, until the desired number of plaits have been generated. However this algorithm remains invisible to the students; they only see input boxes for the parameters.

To make that algorithm visible, we would have to create a "programmable" Culturally Situated Design Tool, or pCSDT. Projects at CMU ("Alice") and MIT ("scratch") have developed programming interfaces that allow students to generate algorithms by dragging and dropping snippets of programming language ("codelets") into a "script" – thus eliminating the frustrating experience of having a program fail because you were missing a comma on line 137. But would students



Cornrow Curves Software

Enter Number of Plaits:

Iterate (integer) 30

Define Starting Parameters:

Starting Point (X,Y) X= -140 Y= 140

Starting Angle (degree) 12

Starting Dilation (percent) 100

Starting Reflection ○X ○Y

Define Iteration Parameters:

Translate (percent) 60

Rotate (degrees) -4.9

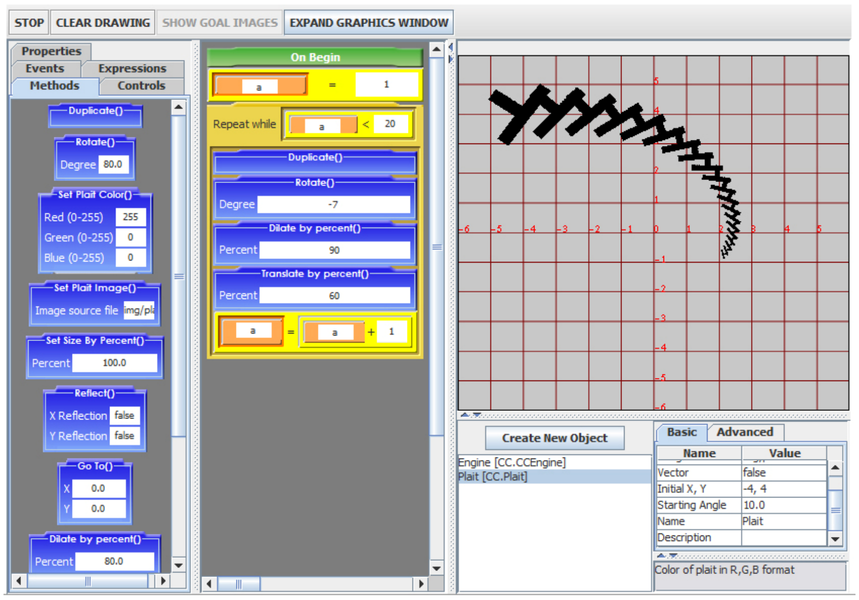
Dilate (percent) 96

Fig. 9.7 CSDT for cornrow hairstyles

who had experienced the ease of the older CSDTs, with a purely parametric interface, be willing to create these scripts? Would we have to hide the older versions from them? Finally, we also needed to create an interface that would be easily extensible for the creation of additional pCSDT applications – we did not want to build a unique interface for each tool. And of course all this needed to happen while keeping true to the cultural connections that motivated the project in the first place.

Our design efforts crystallized around a Java applet that could be easily deployed on the web, but also brought in on physical media (CD or flash drive) in case we were in a situation with low bandwidth (or no bandwidth) internet access. To meet the requirement of being easily extensible, the program is constructed in layers. The Core layer contains the interface that is used for every programmable CSDT. The application layer contains all the code relevant to each specific tool. For example, in the case of the Cornrow Curves applet, the application class says that we want codelets for transformational geometry such as "Rotate," a Cartesian grid for the background, a plait image for the default object, etc. Some codelets such as "Repeat While" loops are common to all pCSDTs, so they lie in the core class. Figure 9.8 shows the resulting pCSDT for cornrows.

The panel at the left contains the list of codelets, the center panel is the script created by dragging and dropping codelets, and the right-most panel is the simulation window. When users are finished with a script they can expand the simulation window to full screen before activating the script. At the top of the script, the user has declared a counting variable (called *a*) and initialized it with a value of 1. The next codelet is a control loop, to the effect of: "While $a < 20$, do the following." Inside the loop are codelets for duplicating the plait image, and applying geometric transformations (rotation, scaling, and translation). At the end



```
Duplicate()
Rotate()  Degree 80.0
Set Plait Color()  Red (0-255) 255  Green (0-255) 0  Blue (0-255) 0
Set Plait Image()  Image source file img/pl
Set Size By Percent()  Percent 100.0
Reflect()  X Reflection false  Y Reflection false
Go To()  X 0.0  Y 0.0
Dilate by percent()  Percent 80.0
```

```
On Begin
a = 1
Repeat while a < 20
    Duplicate()
    Rotate()  Degree -7
    Dilate by percent()  Percent 90
    Translate by percent()  Percent 60
    a = a + 1
```

Engine [CC.CCEngine]
Plait [CC.Plait]

Name	Value
Vector	false
Initial X, Y	-4, 4
Starting Angle	10.0
Name	Plait
Description	

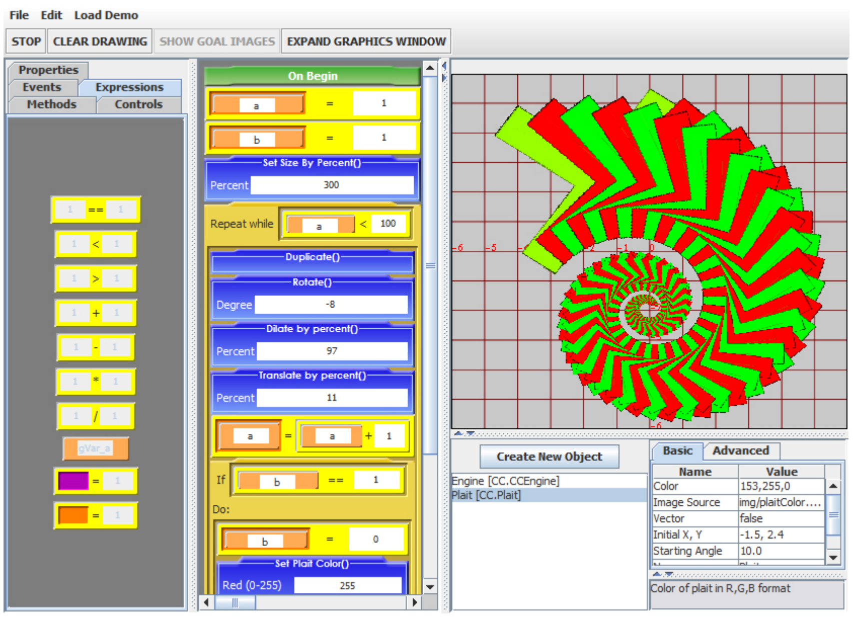
Color of plait in R,G,B format

Fig. 9.8 pCSDT for cornrows

of the loop, the variable *a* is incremented by 1. Thus the script makes visible the algorithm that was invisible to users of the original CSDT. We hypothesize that a non-numeric version of something like this algorithm is also cognitively available to the stylists who create these braids.

One of the most interesting aspects of ethnomathematics simulations is that the results that they produce can surprise the software developers who create them; that is, we were not completely certain what visual patterns we would be able to produce until we actually created the simulation and began to experiment with it. We found that the new pCSDT allows many patterns that were very difficult to make with the old version. For example, in Figure 9.7 you can see three braids created on the old version: Each of those braids required a separate series of trial and error experiments. In the new pCSDT, nesting one control loop inside another allows the user to automate the process of generating a series of braids. Another problem is that real cornrow braids sometimes have rotation values that switch back and forth, like a sinusoidal waveform. On the old version, the user would have to create that effect by piecing together separate braids. The new pCSDT version allows users to introduce conditional codelets (“if-then” or “if-then-else”) so that values such as rotation can be altered at any point in the braid.

The pCSDT version also allows some patterns that are impossible to make with the older version. For example, it is simple to introduce color by placing the color codelet into the script and entering R-G-B values (0-255). By inserting the counting variable (*a*) rather than a value (and this must be the value *a* multiplied by



```
On Begin
a = 1
b = 1
Set Size By Percent()
    Percent 300
Repeat while a < 100
    Duplicate()
    Rotate()
        Degree -8
    Dilate by percent()
        Percent 97
    Translate by percent()
        Percent 11
    a = a + 1
    If b == 1
    Do:
        b = 0
        Set Plait Color()
            Red (0-255) 255
```

Engine [CC.CCEngine]
Plait [CC.Plait]

Name	Value
Color	153,255,0
Image Source	img/plaitColor....
Vector	false
Initial X, Y	-1.5, 2.4
Starting Angle	10.0

Color of plait in R,G,B format

Fig. 9.9 Simulated braid with alternating colors

a constant, for which there are codelets), the color can increment with each plait, such that a braid can begin with blue and end with red, with corresponding gradients of purple in-between. By introducing a second variable (*b*) we can keep track of odd or even duplications, and thus alternate colors in the simulated braid (Fig. 9.9). Interestingly, we later realized that alternating colors are often used in physical braids – the ethnocomputing approach allows us to model aspects that were previously considered irrelevant.

OBSERVATIONS OF STUDENTS USING PCSDTS

The programmable version of Cornrow Curves is currently under investigation. Although we do not have a complete study we present here some initial observations. In the following descriptions we use the pronoun "I" since there was a singular observer.

Observer 1

The school I was assigned was an Alternative School serving students who were expelled from the regular system due to chronic disciplinary problems. The group I worked with was 100% African American, and included only one female student. The variation in attendance was so extreme – essentially a different group of students each time – that it precluded any comprehensive analysis, but it was still possible to make some general observations. First, the software's strength is in its ability to engage students by getting to apply their own experiences to mathematics in a way that showed math that underlies their own understanding. Their enthusiasm was great: they were jumping in to answer rather than needing me to call on them, as I had seen in other lessons. They were talking over each other trying to get my attention; at one point one of the students was on her feet.

One conversation was particularly striking to me: in the course of explaining the concept of ethnomathematics, I posed the following statement to the students: "One plus one equals two, except where it equals three." Initially they pointed out that they didn't think it was true, but as we reviewed possible counter examples (such as an added fee for performing a transaction) the students gained a greater appreciation for how you can think of math as a symbol system that was invented for modeling the world, and that the symbols might be developed differently as long as they were used consistently. One of the students then came up with his own example: he pointed out that you could think of 1 + 1 = 3 as a model for buying "loosies," or individual cigarettes (where it was common to offer three for the price of two as a bargain). The discussion about the use of math continued after the bell; more than anything, it showed that engagement with the material can be enhanced when entering an area where they can contribute their own knowledge and creativity.

The CSDTs are not by any means a panacea; but even in one of the worst possible educational circumstances, they can still be an opportunity for students to begin a mathematical or computational conversation about the world they actually share: in this case, one of cornrows, loose cigarettes, and alternate interpretations of the dominate discourse.

Observer 2

Work at the second school is, at the time of this writing, an on-going study. This middle school is classified as a "high needs" public school and is located in an urban area, with about 650 students in grades 6–8. The student population is about 55% African American, 17% Hispanic, and 9% multiracial, with 72% eligible for free lunch. In addition, the school also faces the daily challenges of educating students that range from cooperative to completely disruptive. As a first trial for the software, I chose a subset of students that would be considered co-operative and willing but who also ranged in academic ability.

The software was used by members of the seventh grade science club, composed of students that had chosen to participate in science enrichment activities out of an interest in science. These students worked with the program for about an hour, during which I recorded observations concerning their use of the program. I paid particular attention to how many objects (in the form of curves) they created and the complexity of the pattern they were able to produce in that period of time. In addition, I recorded my perception of their reactions as they made scripts with the program building blocks, called codelets, to accomplish drawing tasks that interested them.

At the beginning of the work session, I provided them with a brief demo of how to simulate a braid using a script, followed by an overview of how the software functioned. I started with how objects were created (in this pCSDT, there is only one type of object, the plait, but the user can create multiple instantiations of that object) and how the scripting panel worked, and the expectation that each script should start with an "On Begin" event. I explained that once an object is created, the codelet panel fills with all the available codelets for the object.

> First place an "On Begin" Event codelet in the scripting panel, then click on the Methods panel and add method codelets to define the plait pattern you want to create. For example, to create a new curve, click on "Create New Object," choose "Plait" and then begin selecting method codelets to complete the curve definition.

I then demonstrated by adding some codelets and clicking "Begin," so the students would know how to run their scripts and see the results they produced. Finally, I demonstrated deleting an object from the Object Panel by right clicking the object to be deleted and choosing "Delete." After this brief summary on object creation, deletion, and script building I encouraged the students to give it a try.

The students worked individually on netbook computers and each began working with the software. For the remainder of the trial time, I did my absolute best to not interfere unless a student had forgotten to use an "On Begin" Event codelet at the top of their script, and only if they seemed unable to resolve an issue themselves. In wandering from student to student, I did occasionally ask "What were you trying to do?" if something apparently unexpected had occurred, and otherwise just simply praised them on the work that they were doing as general encouragement.

During the course of my observations, I observed that student ability in working with the software ranged from having great difficulty with the programming process to working with relative ease. I will focus on two students at opposite ends of this spectrum; their pseudonyms are Tomas (male Latino) and Zahira (female African American). Tomas demonstrated a fair amount of proficiency in working with the programming aspects of the software, and later mentioned some previous experience with programming. Zahira was at the other end of the spectrum.

Tomas quickly created the scripts necessary to generate a curve on the screen. He had little difficulty in navigating the interface to find the event, control, and method codelets that were necessary to accomplish the task and only once, when I happened to be near him did he ask a question concerning loop creation. Once I reviewed with him how to insert the variables in the control structure for a "do while" loop, he proceeded to complete his script and clicked "Begin." I heard an audible gasp from Tomas, and upon returning to him I found that the results he was expecting was not what was displayed on the screen. I asked him "What were you trying to do?" and he explained how he wanted the loop to function and how he wanted the plait to be drawn across the screen. Upon closer inspection, I realized he had placed the "Duplicate" codelet before the Repeat-While loop, but I did not give him the solution. Rather I suggested that he go back through his script step by step and see if he could figure out how to fix it.

Zahira had taken a different approach to drawing a curve of plaits on the screen, as she was creating new objects for each plait in the curve. Although Zahira was not getting the program results that had originally been demonstrated at the beginning of the session, she was still very engaged in creating her curve on the screen in the manner in which she was able to do so. In addition to creating new plait objects and placing them on the screen using the initial (*x,y*) value in the properties panel, she was also making use of the rotate and dilate codelets resulting in an approximation of the results of the initial program demonstration. While I was observing her working she looked up and asked "how do you make it do the curve on its own?" There is nothing more gratifying in a high needs school than to have a student say "help me."

I took Zahira back through the original example and demonstrated the "Do While" codelet. I also reviewed the different panels containing the Controls, Methods, and Events. Having completed the review, I did not offer any more suggestions unless Zahira asked additional questions. I paid closer attention to Zahira as she worked for the rest of the session because I really wanted to know if she succeeded at climbing the learning curve. She continued to work steadily and did succeed at assembling a loop before the session ended.

At some point, I heard what I thought was an "Ah-hah!" from Tomas which immediately drew me back over to where he was working. He had successfully de-bugged his script and discovered that he had put the "Duplicate" codelet in the wrong place. Having moved "Duplicate" to the correct place the script functioned according to his expectations, which resulted in a very happy Tomas.

In addition to my observations of Tomas debugging a script and Zahira grappling with the beginnings of programming, there were other interesting indications of learning taking place. After hearing a groan from one student, I noticed a hand go up to the screen and trace along the Cartesian coordinate lines of the grid, followed by an "Oh!" and what seemed to be an adjustment of the starting (*x,y*) values, terminating in a "Yay!" Another student spent a significant amount of time experimenting with the starting angle of the plait – it seemed as though every time I passed by where this student was working, the plait was being rotated yet again, quite probably through most of the 360 degrees that are available for rotation!

ANALYSIS OF PRELIMINARY RESULTS.

Inquiry-based learning, in which students either invent a question themselves, or have their inquiry assigned to them, is increasingly supported by innovations in pedagogy (cf. Minstrell & van Zee, 2000). A crucial component of the education theory supporting inquiry learning is that of scaffolding, in which some temporary conceptual aid allows a student to advance their understanding, such that with a firmer grasp on new concepts, they can then climb to higher levels. Brush and Saye

(2002) introduce the terminology of "hard" and "soft" scaffolding. They refer to teachers as providing "soft" scaffolding, by which they mean it is contingent and adapted to circumstances. In contrast, they suggest that multimedia systems, of the type they introduce (which consists primarily of hyperlinked media to support high school social studies inquiry), can be labeled "hard scaffolding" because the designer must pre-plan whatever learning aids will be available.

In our case, neither category fits well: the scaffolding is contingent, not pre-planned. We never anticipated that a student would generate a braid simply by creating each plait as a separate object. But it is not a contingency generated by a teacher; rather it is a contingency generated by the interaction between a student and a digital medium that is sufficiently flexible and powerful to allow creative explorations. Rather than call this hard or soft inquiry, a better category might be "mangled inquiry." Both Tomas and Zahira's struggle with writing a script can be described using the model of "The Mangle" (Pickering, (1995) in which he detailed how scientific discovery occurs as a "dance of agency." Pickering describes the failure to accomplish a particular goal as "resistance" (in the language of Pickering, nature resisting a "capture" of its agency by some model or machine). The scientist then responds by seeking a new strategy to overcome that failure – changing models or machines or procedures until she finds one that works ("accommodation"). In the case of Tomas, the resistance came in the form of a logic error that placed the "Duplicate" codelet outside the control loop in his script. The resulting struggle to find a solution eventually led to accommodation when he moved the codelet inside the loop. However it is critical to understand that in Pickering's view, there is not simply one "correct" model or machine. Multiple different accommodations are possible, including a change of goal. And in fact, there are multiple locations within the script that would have allowed Tomas to successfully generate a braid, although the behavior might have varied slightly (e.g., there would be one less plait if you duplicated after the variable is incremented). Zahira's case consisted of two stages: she attempted an initial strategy that was temporarily successful, creating the braid with individual objects, which allowed her to proceed to the point where she was able to set a higher goal for herself (from the goal of merely making a braid by any means necessary, to the goal of having the script automatically generate the entire braid sequence).

Indeed, we can view our own attempts through this same lens of "mangled inquiry". Our planned evaluation system at the Alternative School met with initial resistance; there was no way to use pre/post evaluations given the enormous variation in student attendance. But we accommodated that resistance by focusing on the discussions that followed the software experience, and thus gained some insights into the elements that increased students' engagement in math and computing conversations.

Inquiry learning works best when it is open-ended. Students need to be able to ask questions, pose answers, and explore the implications of those answers – not necessarily "the one right answer" but rather discovering what new patterns emerge when those answers are used. In that exploration, new questions can then be developed for further consideration. The new pCSDTs offer exactly that scenario. As drawings are created they can be changed by adding additional coding elements. This added complexity will result in scripts that could benefit from rewriting, and the results offer new horizons for further exploration. As Resnick et al. (2009) note about MIT's *Scratch*, it is critical to offer a "low floor" (easy to get started) and "high ceiling" (enormous room for expansion).

CONCLUSION

In closing, we quote from one of the second observer's notes:

> The working session was nearing an end and I requested that students begin to clean up by shutting down their netbooks. There were no students that seemed happy that the session was at an end, in fact all of them seemed genuinely disappointed and expressed an interest in working with the software again. Several wanted to save the script that they had been working on, and I quickly walked them through how to do that. Yesterday, Zahira's friend who was helping with the pattern was practically giddy at the prospect that the tool was designed around the cornrow hair style. She just brightened right up as I was explaining it.

As our study shifts to quantitative data, comparing pre/post tests between control and experimental groups, we will need to leave these qualitative observations behind. But it is certainly these experiences that provide our motivation. If ethnomathematics is the scaffolding that allows ethnocomputing to emerge – scaffolding based as much on issues of social justice and opposition to racism as it is on mathematical modeling – then we do indeed stand on the shoulders of giants, however mangled our inquiry.

ACKNOWLEDGEMENT

This research is supported by NSF grants DGE-0947980 and CNS-0634329.

REFERENCES

Barta, J., & Eglash, R. (2009). Teaching artful expressions of mathematical beauty: Virtually creating Native American beadwork and rug weaving. In J. Braman (Ed.), *Handbook of research on computational arts and creative informatics* (pp. 280–289). Hershey, PA: IGI Global.

Brush, T. A. & Saye, J. W. (2002). A summary of research exploring hard and soft scaffolding for teachers and students using a multimedia supported learning environment. *The Journal of Interactive Online Learning, 1*(2). Accessed April 20, 2011 at URL http://www.ncolr.org/jiol/issues/getfile.cfm?volID=1&IssueID=3&ArticleID=58.

Eglash, R. (1999). *African fractals*. New Brunswick, NJ: Rutgers University Press.

Eglash, R. (2009). Native American analogues to the Cartesian coordinate system. In B. Greer, S. Mukhopadhyay, A. B. Powell, & S. Nelson-Barber (Eds.), *Culturally responsive mathematics education* (pp. 281–294). New York: Routledge.

Eglash, R., & Bennett, A. (2009). Teaching with hidden capital: Agency in children's mathematical explorations of cornrow hairstyle simulations. *Children, Youth, and Environments*, *19*, 58–74.

Eglash, R., Bennett, A., O'Donnell, C., Jennings, S., & Cintorino, M. (2006). Culturally situated design tools: Ethnocomputing from field site to classroom. *American Anthropologist*, *108,* 347–362.

Eglash, R., Krishnamoorthy, M., Sanchez, J., & Woodbridge, A. (2011). Fractal simulations of African design in pre-college computing education. *ACT Transactions on Computing Education*, *11*(3).

Fordham, S. (1991). Peer-proofing academic competition among black adolescents: "Acting white" black American style. In C. Sleeter (Ed), *Empowerment through multicultural education* (pp. 69–94). Albany, NY: State University of New York Press.

Fryer, R. G., Jr., & Torelli, P. (2005). *An empirical analysis of "acting White."* Available from: http://www.economics.harvard.edu/faculty/fryer/files/fryer_torelli.pdf.

Margolis, J. (2008). *Stuck in the shallow end: Education, race, and computing.* Cambridge, MA: The MIT Press.

Mathematical Sciences Education Board. (1989). *Everybody Counts*. Washington, DC: National Academy Press.

Minstrell J., & van Zee, E. (Eds.). (2000). *Inquiring into inquiry learning and teaching in science*. Washington, DC: American Association for the Advancement of Science.

Pickering, A. (1995). *The mangle of practice: Time, agency, and science.* Chicago, IL: University of Chicago Press.

Resnick, M., Maloney, J., Monroy-Hernandez, A., Rusk, N., Eastmond, E., Brennan, K., et al. (November, 2009). Scratch: Programming for everyone. *Communications of the ACM, 52*(11).

Rosa, M., & Orey, D. (2010). Ethnomodeling as a pedagogical tool for the ethnomathematics program. *Revista Latinoamericana de Etnomatemática*, *3(2),* 14–23.

Wing, J. (2006). A vision for the 21st century: Computational thinking. *CACM*, *49*(3), 33–35.