Digitizing Data: Computational Thinking for Middle School Students through Computer Graphics

R. Cutler[1] and M. Hutton[2]

[1]Purdue University, Indiana, USA
[2]The Girls' Middle School, California, USA

Abstract

The concept of digitized data is fundamental to computer science, yet for many students, there is a disconnect between objects they encounter outside the computer and the data they interact with in the computer. A programming-based approach can exacerbate the problem for young students who are developmentally unready for the abstraction required to translate the world into objects described through the syntax of a programming language. This case study describes the creation of a curricular unit called Digitizing Data*, delivered in an eighth grade all-girls computer science class. The unit extends the* CS Unplugged *"Image Representation" lesson into a series of coordinated projects, culminating in students using a custom-built application to visualize three-dimensional objects and spaces. The project successfully engaged students in computational thinking, communicated a fundamental computer science topic without the barriers of programming, and allowed them to express computer science concepts creatively.*

Categories and Subject Descriptors (according to ACM CCS): K.3.2 [Computers and Education]: Computer and Information Science Education—Computer science education I.3.3 [Computer Graphics]: Picture/Image Generation—Digitizing and scanning

1. Introduction

Middle school, typically defined as grades 6-8 when students are approximately 11-14 years old, is an opportune time to develop computational thinking skills in students. By this time in their education, students should have received a solid grounding in arithmetic and been introduced to the scientific method and inquiry-based learning in their science classes. They have begun to expand their critical reasoning and analytical abilities thorugh formal essay writing and more in-depth mathematical processes in pre-algebra and algebra. As students mature from the concrete operational stage of cognitive development to the formal operational stage, they are ready for the introduction of computational thinking's higher-order skills. [Pia72]

Computational thinking is variously described as "solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science" [Win06], "a problem-driven approach focused on scientific discovery through computational methods grounded in computer science principles" [AHPS09], and "the automation of abstractions" [Win09]. A related term, *computationalist thinking*, is depicted as having "core competencies in modeling, scales and limits, simulation, abstraction, and automation" [ISC*09].

As applied to middle school computer science, we concur with all of these descriptions and approach computational thinking as a data-focused method of problem solving using higher-order thinking skills such as abstraction, conceptualization, modeling, and simulation.

We distinguish an emphasis on computational thinking from the programming focus of traditional computer science courses. While programming is often considered "the language of computer science", trying to learn complicated computational thinking concepts and skills while simultaneously learning a new language is a cognitive barrier for many beginning students. Students can engage more deeply in complex computational questions when they are freed from the stricture of programming syntax.

Experimental validation that boys have better spatial skills than girls has been broadly documented [Fen74]. While boys and girls have largely overlapping areas of strength and weakness, "gender differences were well established in only four areas: verbal ability, visual-spatial ability, mathematical ability, and aggression." [Hyd05,MJ74] Students with strong spatial abilities are more successful in mathematics, possibly due to the structure of mathematics relying on spatial visualization [Cle04]. Due to the shared basis of logical thinking, mathematically strong students are frequently good at computer science as well. Two- and three-dimensional spatial puzzles have been demonstrated to improve students' spatial reasoning skills [JB06]. A curriculum focused on modeling two- and three-dimensional objects should effectively improve girls' spatial reasoning skills, and therefore performance in math and computer science.

At the same time, the teaching of computational thinking must be both contextualized and visualization-centered. Its skills must be taught within the concrete framework of a specific domain, such as mathematics [HEB08, RHL*09], storytelling [KPK07], history, literature, or art [RHL*09], and its learners must be actively engaged in the visualization process [NRA*02, FH95]. Connecting computational thinking skills to domain knowledge and presenting the material visually helps students broaden their understanding, apply the concepts outside of a narrow focus in computer science, and engage in the material by reinforcing the value of both the skill and the content [Rie91].

Finally, the appropriate tools used in conjunction with proper scaffolding and curricular structure can help students "express themselves creatively" and "develop as creative thinkers." [RMN*05] This serves to increase students' engagement with computational thinking, diversify the artifacts they create, and allow them to use computational thinking in other creative domains.

2. Digitizing Data

Digitizing Data was designed to be a unit in a computer science course, taking place in approximately 25 class periods of 50 minutes each, spread out over a timeframe of ten weeks. The goals of this unit are for students to understand what it means to digitize data and to be able to digitize objects and pictures in their world. Specifically, *Digitizing Data* was designed to use a variety of computational thinking skills to deeply understand how data is digitized and handled computationally.

Projects were designed to build on students' previous coursework in mathematics, engineering, and design, and to engage both analytical and creative thinking processes. In modeling two- and three-dimensional pictures, objects, and spaces, students had to analyze the component parts and convert them into numbers. Then students had to understand the transformation of the numbers into abstracted visual representations of the original objects. Emphasis was placed on modeling and abstraction rather than on precisely defining or describing the real-world objects.

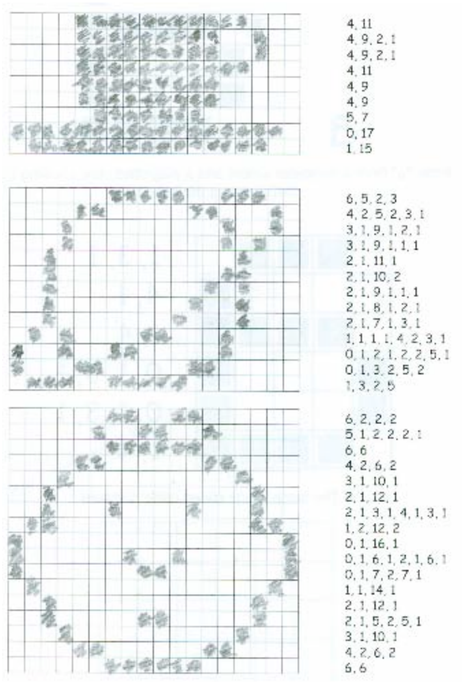

Figure 1: *Simple blank and white decoding activity from* CS Unplugged *[Com05]. The numbers alternately represent the numbers of white and black pixels in a given row.*

2.1. Setting

The unit was tested in the eighth grade computer science class at The Girls' Middle School (GMS) in Mountain View, California. GMS is located in Silicon Valley. The school has a diverse student population: over 40 percent of eighth graders are students of color. 20 percent of students are on full financial scholarships, which include tuition and fees, lunch program, uniforms, and transportation to school. The school has a 1:1 laptop program, where each student is provided a laptop to use at school and home, which provides equity of access to technology.

GMS provides a challenging academic curriculum. There is a focus on project-based learning and group work. Computer science is a required class at all three grade levels and engineering is a core part of the eighth grade science curriculum. Students frequently reflect on their learning; for example, instead of grades, narrative reports are provided which include both teacher feedback and student self-evaluation.

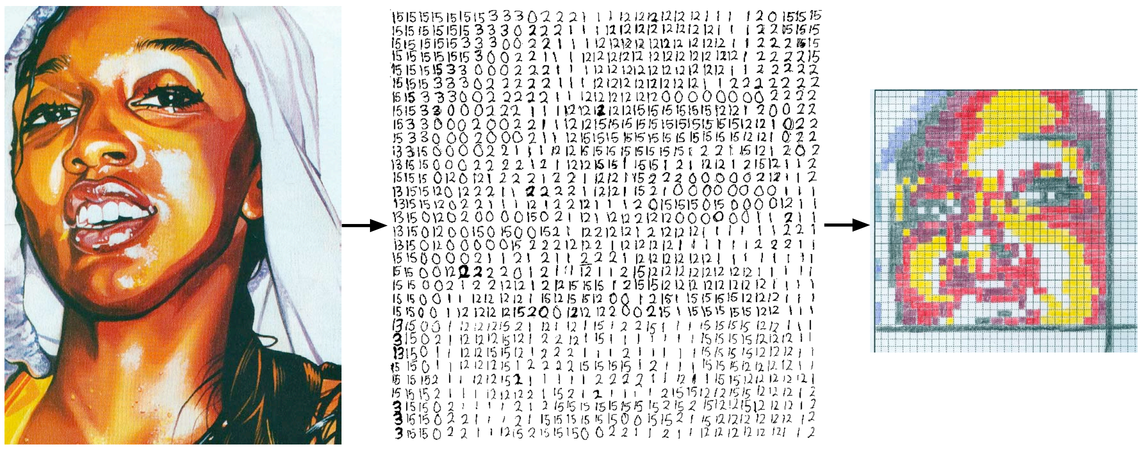

Figure 2: *The student samples a portion of the image, assigning values from 0-15 based on the closest color from a palette. She then decodes the digitization using the colors from the same or a different palette.*

In sixth and seventh grade, the school utilizes a traditional approach to computer science education, including units on robotics, user-centered web design, database design, and object-oriented programming using the visual environment *Stagecast Creator*. In eighth grade, the curriculum focuses on computational thinking separate from the technology used. Digitizing data followed an initial unit about modeling data, in which students learned to describe data through attributes and methods. Students began this unit already comfortable with creating data visualizations, including pictures and small animations.

2.2. Curriculum

The unit began with a discussion about what "digital" means. The teacher explained computer science definitions of digital ("property of representing values as discrete numbers rather than a continuous spectrum") and digitize ("put data into a digital form so that it can be processed by a computer"). Throughout the discussion, students connected the idea of digital data to previously learned material such as modeling data.

Next, the teacher used the *CS Unplugged* activity on Image Representation [Com05] to teach how to digitize black and white pictures and how to compress the encoded representation using run-length encoding (figure 1). Students practiced encoding and decoding simple black and white pictures.

2.2.1. Digitizing color pictures

In the first project, students transferred their knowledge of digitizing pictures to digitize color pictures (figure 2). Given a picture, a transparent grid overlay, and a 16-color palette, students encoded the picture onto graph paper, choosing the "closest" color for each square. They decoded the picture by coloring in squares on a paper grid. After encoding and decoding the picture once, the students were encouraged to create a compression algorithm and re-encode the picture with a different pixel size. The project concluded with a discussion about pixillation, file size, encoding strategies, and connections to the computer.

2.2.2. Digitizing LEGO figures

As an introduction to three-dimensional visualization and the visualization software we built, students digitized an abstract LEGO figure. The goal was to provide them with a small project in which they could use to learn the necessary skills for the room-digitizing projects. They used 10 or more LEGO pieces to create a figure, drew an engineering drawing of all six sides, plotted its points using x, y, and z values, created a data file, and used custom-built software to visualize the figure (figure 3).

Creating engineering drawings reinforced a skill learned in science class. Because each side was often unique and colors mattered, students had to draw all six sides instead of the usual three.

The teacher reminded students about the three axes in three-dimensional objects, material they had covered previously in computer science and math. As a whole class activity, students digitized a simple two-block LEGO figure, with the teacher modeling how to approach the engineering drawings, keeping track of points, numbering, and determining the x, y, and z values. Then students digitized their own, more complex figures, keeping track of the data on paper.

Finally, the teacher demonstrated custom software built for this project, used to visualize the digitized figures. She showed several example LEGO figures as well as a digitized

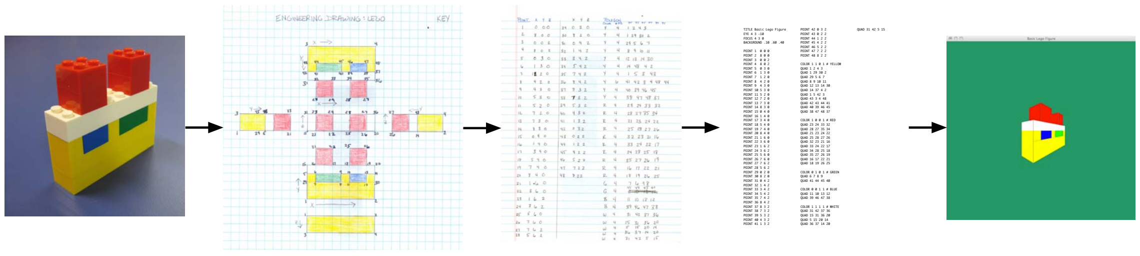

Figure 3: *The five steps in digitizing a LEGO structure: 1) build the structure, 2) make engineering drawings, 3) digitize the points, 4) create the data file, 5) visualize the model.*

room, demonstrated how to manipulate the visualized figures, and explained how to format the data for interpretation by the software. Students input their data, encouraged to stop and test the visualization at frequent intervals.

2.2.3. Digitizing the classroom

The final project in the unit was a group project, with students digitizing the classroom. Students brainstormed the parts of the classroom: bulletin boards, bookcases and cabinets, walls, doors, windows, tables, and chairs. Students formed small groups, each assigned a few objects to digitize. For example, one group digitized the bookcases and cabinets while another took on the walls, doors, and windows. The teacher provided a common coordinate system for the room, with all groups using the same origin and axes, and a numbering system so that each group would have a unique set of point ids when the various data files were combined.

This project also provided ample opportunity for differentiation. In the LEGO structures project, students used an absolute coordinate system to define their objects. In the more complex classroom project, the weaker students continued to use the absolute measurement system of the classroom to create their objects. The stronger students, however, were shown the "define" command provided in the software. This command provides the capability to define an object once and then place it multiple times at different locations in the scene. This allowed these more advanced students to be assigned the digitization of objects, such as the tables and chairs, which appear multiple times in the classroom (figure 5).

This project took substantially less time than the LEGO project, encompassing only approximately five class periods in comparison to the ten for the LEGO project. This was due in part to the absence of an engineering drawing requirement when digitizing the classroom. Although all groups did create pictures to help identify points, they used quick sketches instead of the time-intensive, accurate engineering drawings created for the LEGO structures. Additionally, most groups created far fewer points for their objects in the classroom than they had for the relatively more complex LEGO figures.

2.3. 3D Visualization Tool

In order to allow students to see the results of their data modeling and to demonstrate how the mathematical models they built of the LEGO structures and the classroom could be viewed, we built a small computer application, named *McEfs*. (The name is made up of the initials of the first names of five girls who participated in a focus group about the software. An interesting outcome from naming the software after the students was that it served as a tremendous motivator for *all* of the students, helping to give them a very personalized ownership of the digitizing task.)

McEfs is a cross-platform application written in C++ using the OpenGL libraries. It reads in an ASCII text data file consisting of tagged three-dimensional data (table 1) such as points, lines, rectangles, and colors, and displays a perspective visualization in a window. Keyboard commands (table 2) allow the students to rotate and/or move around in the scene. For both the data file and keyboard interface, commands were deliberately kept simple in order to allow the students to focus on the models rather than on the syntax of the language or interface.

Table 2: *Keyboard Commands*

Key	Function
x / X	Rotate object around x-axis
y / Y	Rotate object around y-axis
z / Z	Rotate object around z-axis
i	Move forward
k	Move backward
j	Move left
l	Move right
[space]	Move up
,	Move down
u	Turn/rotate left
o	Turn/rotate right
m	Look down
.	Look up

3. Results

As an effort to put the *CS Unplugged* materials into context and to have students understand digitizing data, this project was a success. Students were able to complete the projects and describe their understanding of fundamental concepts. Students overwhelmingly expressed the opinion that while the unit was challenging, it was worthwhile and they were proud of their accomplishments.

The unit, particularly the third project on digitizing LEGO structures, was affected by several outside interruptions which negatively affected students' understanding. Two days after the project was introduced, the students went on an extended field trip, missing two weeks of computer science classes. Following the trip, an outbreak of H1N1 influenza hit the class which resulted in numerous absences and finally in school being closed for a week.

3.1. Digitizing color pictures

The digitizing pictures section of the unit was particularly successful. Students intuitively understood the two-dimensional grid and how to convert between colors and numbers. They made connections between this process and computer-based topics such as scanning and pixelation. The project was portable, easy to understand, and engaging for students. (Additionally, it looked nice on the class bulletin board and students were excited to see their work displayed.)

Students created several compression strategies, most centering on doing less writing rather than conveying the information efficiently. Examples included drawing lines to encompass blocks of color on the graph paper then writing the number in the block and writing a number for the color followed by the number of squares to fill with that color in parentheses.

Several extensions would enhance this project. Students could be asked to decode a classmate's digitized picture. This was the original assignment, but had to be abandoned due to time constraints and the necessity of students' completing the assignment at home. Different palettes could be used for decoding the picture, such as grayscale or sepia.

3.2. Digitizing 3D objects and spaces

Students were challenged by the individual LEGO project, though ultimately all were successful. Overall, students liked the assignment. Not only did students express a great deal of enthusiasm about being able to see and manipulate the figure in the computer, but parents commented on how excited their daughters were. One parent mentioned that her daughter had mentioned this project in a high school interview as something of which she was particularly proud.

Students enjoyed creating and coloring their drawings, and were surprised to find engineering drawings a skill used outside of science. Some students struggled with lining up the different views and creating the drawings took some students two class periods, which was much longer than anticipated. Many students created very creative, but quite complex LEGO figures, which accounted for some of the challenge in creating drawings (figure 4).

Students found the data file format easy to use. With very little explanation, they were able to successfully put the data into the computer and visualize their figures. One student commented, "I liked it when the points had to be typed into the software because that's when you didn't have to think, you could just type it in."

A number of students had difficulty determining the x, y, and z values for each point, which may be due to weak spatial skills. If digitizing the figure was challenging, for some students, debugging problems in the data was nearly impossible. Due to the dense numerical nature of the data and because this was a new skill, debugging errors was a test of students' perseverance. Some students debugged individually, many asked for help from the teacher, who usually told them to go back and double-check their points and quads. A few gave up and started over, with the belief that it would be easier a second time.

The sequencing of the projects was difficult. Partway through the LEGO project, students asked for it to be revised – to have the entire class work on the same LEGO figure or at least small groups working on the same figure. They believed that in this way they would be able to support each other in learning the skills required. By the time the group room-digitizing project began, most students had cemented their ability to create points and some of the collaborative benefits of students' supporting each other was lost. In the future, we would reverse the order of these two projects.

Some students struggled to transfer knowledge between the LEGO project and the classroom digitization project. Students used prior knowledge of how to measure objects (such as a bulletin board) but some had difficulty translating width, height, and depth measurements into a set of points. Some were confused when placing objects into position using the coordinate measurements of the room, and many students found it difficult to understand how to abstract the objects in the classroom in order to digitize them.

Although students enjoyed working together during the classroom digitizing group project, and were able to support each other in figuring out how to approach the assignment, they experienced several problems. First, different tasks had different ideal group sizes. A group of three was ideal for measuring objects, with two students wielding a measuring tape or yard stick and one student recording the information. However, when it came to converting the measurements into points, it was harder for all three students to participate actively together. Middle school students frequently do not have great foresight, and a few groups had to re-measure

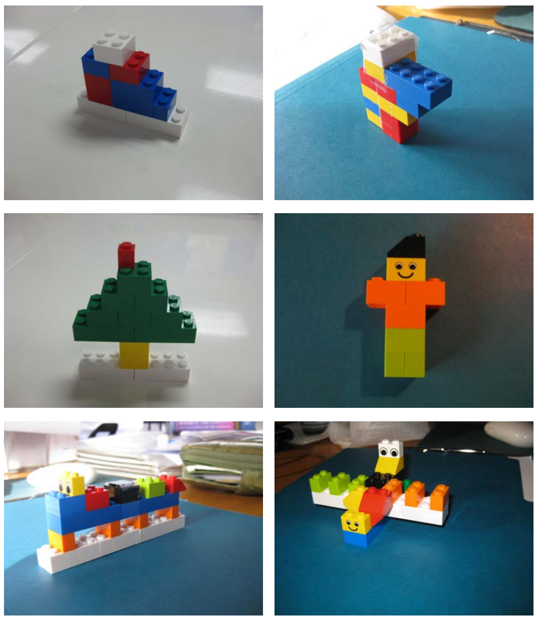

Figure 4: *Examples of creative student-made LEGO structures ranging from the simple to the complex.*

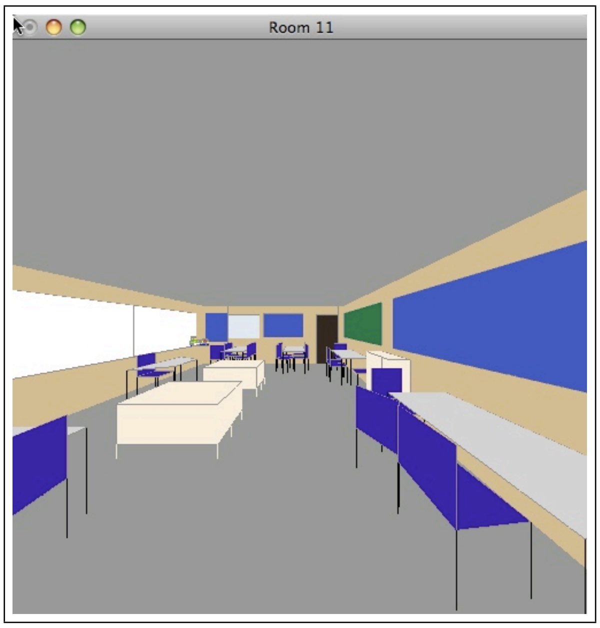

Figure 5: *Visualization (using the* McEfs *software) of student digitization of tables and chairs in their classroom.*

objects when the recorder was absent from class on a subsequent day and had not shared the data with her teammates. The sense of responsibility to the whole class was motivating for most students, and they were able to complete a good representation of the classroom.

4. Related Work

We see potential synergies between the work of engineering educators who are interested in teaching graphics education and our work in teaching computational thinking skills. While both approaches use a descriptive geometry to define points using a Cartesian coordinate system, the underlying goals for using this geometry are somewhat different.

The engineering-centered curriculum focuses on the need to make pictorial sketches of solid objects and learn the mathematical basis for manipulating, viewing, and anotating those objects using transformations (scale, translate, rotate, reflect), projections (orthographic, parallel, perspective, etc.), and dimensioning [SMB98]. A large part of this learning is geared toward improving students' design and visualization skills [Sor03].

Our work, on the other hand, emphasizes data modeling and abstraction. Our students are not interested in precisely defining the real world, but rather using graphics to understand how the real world can be modeled in a way (and with a level of detail) that is developmentally appropriate for their age and educational level. While design and visualization skills are important side effects of the process, our focus is on developing the computational thinking skills that can then be transferred across disciplines.

5. Conclusion and Future Work

Based on our case study of one teacher, forty-nine female students, and the first use of an untested curriculum and software prototype, we are cautiously optimistic about the potential for teaching computational thinking to middle school students using 3D data modeling.

Feedback from the teacher and students was positive. We observed that the students were engaged in the assignments and, despite some difficulties, successfully learned about 3D data modeling through the project work. Several parents of students emailed the teacher to report on their child's interest in and enjoyment of the assignment. Asked if future classes should be assigned the LEGO digitization project, 62 percent of students agreed or strongly agreed. A typical quote was, "I really liked the feeling of accomplishment after having struggled with the project and knowing it was complete and done to my best effort."

Five students participated in a focus group to discuss the 3D visualization tool. Overall, the students said they liked the software; however, they had several suggestions to improve its functionality and interface. One such recommendation – adding rotational commands – was implemented immediately. This improvement was extremely well-received by the students and increased the ease-of-use of the tool when viewing the LEGO structures.

Other student suggestions – such as adding lighting and texturing, and integrating the data file into a separate window of the application – will be integrated into a future version of the software.

Finally, creativity was seen in the learning process of the students in their choices of the pictures for digital encoding/decoding, their designs of the LEGO structures, and in their approaches to digitizing the three-dimensional structures. We saw that the creative aspect of the assignment enhanced the project by melding divergent thinking with the more quantitative analysis required.

Our next step is to continue research in this area by validating these anecdotal results through further and more formal testing of the curricular unit at this and other schools. We also plan to continue to improve the curriculum and 3D visualization tool for future use, and develop other visualization software to enhance these units.

5.1. Future Work

We also see three areas for additional research building off of this project. First, there is great potential for further exploration of the creative process. Measuring the effect of creativity on learning (and finding a positive correlation or causality) would have a transformative impact on education.

Second, we can examine the effect of building computational thinking skills on learning in other disciplines such as math and engineering. There is anecdotal evidence that for several students the visualization software clarified their understanding of the engineering drawings (which they had previously learned how to do in science).

Third, we can look at the effect of improving spatial skills in this manner on math/science learning in middle school. There is evidence to suggest that 3D spatial skills improve success in technical career fields and are fundamental to high-level thinking and creative skills. It appears to be an open question whether teaching these skills to middle school students would be effective at increasing their success.

References

[AHPS09] ASTRACHAN O., HAMBRUSCH S., PECKHAM J., SETTLE A.: The present and future of computational thinking. In *Proceedings of the 40th ACM Technical Symposium on Computer Science Education* (March 2009), pp. 549–550.

[Cle04] CLEMENT D. H.: *Engaging Young Children in Mathematics*. Erlbaum, Mahwah, NJ, 2004.

[Com05] COMPUTER SCIENCE UNPLUGGED: Activity 2: Colour by numbers – image representation. Retrieved from http://csunplugged.org/image-representation, 2005.

[Fen74] FENEMMA E.: Mathematics, spatial ability, and the sexes. presented at the american educational research association annual meeting, chicago. Available from http://www.eric.ed.gov/ERICDocs/data/ericdocs2sql/content_storage_01/0000019b/80/39/5a/02.pdf, 1974.

[FH95] FERGUSON E., HEGARTY M.: Learning with real machines or diagrams: application of knowledge to real-world problems. *Cognition and Instruction 13* (1995), 129–160.

[HEB08] HART M., EARLY J. P., BRYLOW D.: A novel approach to K-12 CS education: Linking mathematics and computer science. In *Proceedings of the 39th ACM Technical Symposium on Computer Science Education* (March 2008), pp. 286–290.

[Hyd05] HYDE J.: The gender similarities hypothesis. *American Psychologist 60*, 6 (2005), 581–592.

[ISC*09] ISBELL C., STEIN L. A., CUTLER R., FORBES J., FRASER L., IMPAGLIAZZO J., PROULX V., RUSS S., THOMAS R., XU Y.: (Re)defining computing curricula by (re)defining computing. In *Working Group Reports of the 14th Annual Conference on Innovation and Technology in Computer Science Education* (July 2009), ACM Press.

[JB06] JONES S., BURNETT G.: Give the girls a chance – should spatial skills training be incorporated into the curriculum? In *The Internet Society II: Advances in Education, Commerce and Governance* (Boston, MA, 2006), Morgan K., Brebbia C., Spector J., (Eds.), WITpress.

[KPK07] KELLEHER C., PAUSCH R., KIESLER S.: Storytelling Alice motivates middle school girls to learn computer programming. In *Proceedings of the 25th Annual Computer/Human Interaction Conference* (May 2007), ACM Press, pp. 1455–1464.

[MJ74] MACCOBY E. E., JACKLIN C. N.: *The Psychology of Sex Differences*. Stanford University Press, Stanford, CA, 1974.

[NRA*02] NAPS T., RÖSSLING G., ALMSTRUM V., DANN W., FLEISCHER R., HUNDHAUSEN C., KORHONEN A., MALMI L., MCNALLY M., RODGER S., VELÁZQUEZ-ITURBIDE J. A.: Exploring the role of visualization and engagement in computer science education. In *Working Group Reports of the 7th Annual Conference on Innovation and Technology in Computer Science Education* (July 2002), ACM Press, pp. 131–152.

[Pia72] PIAGET J.: *The psychology of the child*. Basic Books, New York, New York, 1972.

[RHL*09] RODGER S. H., HAYES J., LEZIN G., QIN H., NELSON D., TUCKER R.: Engaging middle school teachers and students with Alice in a diverse set of subjects. In *Proceedings of the 40th ACM Technical Symposium on Computer Science Education* (March 2009), pp. 271–275.

[Rie91] RIEBER L.: Animation, incidental learning, and continuing motivation. *Journal of Educational Psychology 83* (1991), 318–328.

[RMN*05] RESNICK M., MYERS B., NAKAKOJI K., SCHNEIDERMAN B., PAUSCH R., SELKER T., EISENBERG M.: Design principles for tools to support creative thinking. Unpublished draft, 2005.

[SMB98] SORBY S. A., MANNER K. J., BAARTMANS B. J.: *3-D Visualization for Engineering Graphics*. Prentice Hall, Upper Saddle River, New Jersey, 1998.

[Sor03] SORBY S. A.: *Introduction to 3D Spatial Visualization: An Active Approach*. Delmar, Clifton Park, New York, 2003.

[Win06] WING J.: Computational thinking. *Communications of the ACM 49*, 3 (March 2006), 33–35.

[Win09] WING J.: Computational thinking [PowerPoint slides]. Retrieved from http://www.nsf.gov/events/event_summ.jsp?cntn_id=115161&org=CISE, 2009.

Table 1: *Data File Commands*

Command	Description
# *comment*	Any text on a line after # is ignored. Note that there must be a space between # and the *comment* text.
TITLE *text*	Sets the title of the window to *text*.
BACKGROUND *r g b*	Sets the background to the color (r,g,b), where r, g, and b are values between 0 and 1 representing the amounts of *red*, *green*, and *blue* respectively. This command should occur only once near the beginning of the file.
EYE *x y z*	Sets the user's "eye" to (x,y,z). The user will "look" from the **eye** in the direction of the **focus**.
FOCUS *x y z*	Sets the **focus** of the environment to (x,y,z). The center of the perspective view is set to (x,y,z), and the distance between the **eye** and the **focus** affects the initial view.
POINT *p x y z*	Define point (x,y,z) with id p (where p is an integer). Two points cannot have the same id.
QUAD p_1 p_2 p_3 p_4	Draw a quadrilateral surface with corner points p_1, p_2, p_3, and p_4 (where p_1, p_2, p_3, and p_4 are the ids of points defined using the **POINT** command. Points must be defined before they are referenced. The quadrilateral is drawn and filled with the current drawing color.
COLOR *r g b a*	Sets the drawing color to (r,g,b,a), where r, g, b, and a are values between 0 and 1 representing the amounts of *red*, *green*, *blue*, and $alpha$ blending respectively.
STEP *amount*	Sets the amount that the user moves for each press of a motion key. Defaults to 1.
LINE p_1, p_2	Draw a line with endpoints p_1 and p_2 using the current drawing color.
TEAPOT *size x y z*	Draw a teapot with "radius" *size* centered at (x,y,z) using the current drawing color.
SPHERE *r x y z*	Draw a sphere with radius r centered at (x,y,z) using the current drawing color.
DEFINE *id* ... END *id*	Define an object with numeric id *id*. Any commands between the **DEFINE** and **END** commands are executed each time the object is displayed using the **DISPLAY** command. It's best to center the object definition at $(0,0,0)$.
DISPLAY *id x y z r s t*	Display the object defined by *id* such that $(0,0,0)$ in the definition corresponds to (x,y,z) in the environment. The object is rotated (before transformation) around the x-, y-, and z-axes by r, s, and t degrees respectively.