Embodied Fabrication:
Body-Centric Devices for Novice Designers

by

Benjamin A. Leduc-Mills

B.A., University of California, Santa Cruz, 2003

M.P.S., New York University, 2008

M.S., University of Colorado, 2013

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

2014

This thesis entitled:
Embodied Fabrication:
Body-Centric Devices for Novice Designers
written by Benjamin A. Leduc-Mills
has been approved for the Department of Computer Science

Michael Eisenberg

Clayton Lewis

Tom Yeh

Date

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.

Leduc-Mills, Benjamin A. (Ph.D., Computer Science)

Embodied Fabrication:

Body-Centric Devices for Novice Designers

Thesis directed by Prof. Michael Eisenberg

devices under the umbrella moniker of “embodied fabrication”. These devices and the development of the term “embodied fabrication” is rooted not only in computer science, but also in cognitive science, children’s educational theory, digital fabrication technology, and the convergence of these strands present in the do-it-yourself community known as the“maker movement”. As such, we operate under a certain set of premises that guide and direct this work. First, that embodied cognition - which places the body at the center of our cognitive operations - provides a framework in which to ground our decisions to design physical peripheral devices as opposed to purely screen-based software. Second, that a strong line of research supports providing children with tangible, “manipulative” objects to learn with. Third, that digital fabrication technologies - 3D printing in particular - provide a wonderful new opportunity for children and novice designers in general to make, play, and explore creatively - and that the current design options for 3D printers are not suited to meaningful design and creation of objects by non-expert 3D modelers. Finally, that by continuing in the best traditions of body-centric interaction design for children, a suite of devices can be created to provide an educationally and technically rich environment to connect kids to the creative potential of 3D printing. We unpack these ideas more in the introduction, followed by an overview of all three of the prototype devices and the companion software that constitutes this work, a chapter on related work, a chapter on the three user studies we performed on these devices, a discussion of the presented studies, and finally we present a vision of the future of this work and of embodied fabrication devices as a whole before concluding.

Dedication

To all the killas and the hundred dolla billas.

Acknowledgements

I was incredibly fortunate to have found the supporting cast that I had during my time at Colorado. My advisor, Michael Eisenberg, is one of the true gems of humanity. An unbelievable scholar, gifted writer, and a wonderfully supportive and warm person. Over our many hours of discussion he taught me how to love and appreciate the world where children's crafts, cognitive science, and computation come together. My father, for urging me to visit Mike as I was applying to grad school. You always let me know how proud you were. You and I are more similar than I ever would have imagined as a child. I see that as a wonderful thing. My mother, for her unwavering faith that I would be excellent. You were much better at convincing me than I was at convincing myself. Your visits were always a chance to recharge my batteries, feel loved, and be fed extremely well. Monika, for taking the risk of leaving your world behind to join me here in Colorado, and for supporting me so well during a very stressful time. You're the best cheerleader a guy could ask for. Many others contributed to this work, academically and socially. I owe an enormous debt of gratitude to Nathan Siedle, whose generosity allowed my work to reach its completion. My committee members were all amazing. Ann Eisenberg kept the lab from self-destruction more times than I can count. My CTG and CU compatriots for welcoming me to Colorado and convincing me to stay. Lindsay Levkoff Diamond, for being an incredibly understanding and flexible boss during my studies. The DOE and all my friends at SparkFun Electronics - Toni, Amanda, Dave, Brennen, Jordan, (etc.) without you, I don't know if I would have made it. And finally, Fran, Anna, Adrian, Victor, Galen, and the amazing kids at Gold Crown for their support. Thank you all.

Contents

Tables

Table

Figures

Figure

Chapter 1

Introduction

Digital fabrication technologies are increasingly finding their way into educational spaces of all shapes and sizes. These new technologies (3D printers, laser cutters, etc.) afford opportunities for exploring these new ways of 'making' and how they may change the way we learn, explore, and play. Although there is much excitement surrounding the 'maker movement' - and 3D printing in particular - there has been little examination of how to introduce a younger audience to 3D printing in an empowering way. This proposal argues that tangible interfaces - as opposed to 2D screen-based media - can be designed not only to support spatial reasoning and mathematical intuitions in children by engaging them in exploratory modeling and play, but that these interfaces can act as a democratizing force by enabling children to create physical objects with digital fabrication devices. The proposed work presents a series of novel tangible input devices for enhancing mathematical and spatial reasoning in kids with a focus on generating output for 3D printing. We discuss related work, the status of the proposed work, additional improvements to be made, a timeline for completion, and a discussion of risks, limitations, and outcomes inherent in the proposal.

A number of computer scientists, technologists, and educators have declared that the era of personal fabrication is upon us[10][53]. New devices aimed at increasing the ability of the individual to physically manufacture their own ideas are being released at breakneck speed. The cultural and technological shifts caused by this change are taking many forms, yet few technologies associated with the 'maker' movement have received as much attention as 3D printing - the ability (by various means) to digitally design and then print out physical 3-dimensional objects. Media outlets from

Forbes[60] to The Economist[1] have extolled the disruptive and democratizing possibilities that 3D printing offers - at least as it affects the traditional manufacturing supply chain. Less examined has been how to introduce novices, specifically pre-teens and early adolescents, to 3D printing - and perhaps more importantly - discussing what (and how) they might learn by being exposed to it. While the variety of desktop 3D printers continues to increase and the cost of adding a 'fab lab' of digitally-based manufacturing tools in the home or classroom steadily declines, the types of interfaces by which children can easily and intuitively design and explore the capabilities of 3D printers still remains a barren landscape consisting primarily of software-only solutions. It is this landscape that we are interested in seeding, following the best practices in computational and cognitive science with particular attention to children-centered design.

To this end, we present a class of tangible user interfaces (TUIs) designed to scaffold a child's ability to design, explore, and play in three dimensions, with a particular focus on enabling output for 3D printing. Significant work has been done on three devices that allow users to specify points on a physical, interactive, volumetric interface that simultaneously displays active points in real-time on a computer. The software on the computer allows for certain modeling operations on the set of input points (e.g., taking the convex hull), as well as exporting shapes to stereolithography (.stl) format, the preferred format for 3D printing. We propose that these designs form a new class of tangible input devices, and present early work on a portable pop-up book building upon the ideas expressed in the two earlier designs. The rest of the paper proceeds as follows: discussion of related work, description of completed work, the proposed work, a timeline for completion, and concluding thoughts.

The work presented here draws on the stages of childhood developmental theories and conception of space developed by Piaget and refined by Papert, notions of cognitive development and embodied mathematics discussed by Lakoff and Nuñez, the democratization of digital fabrication technologies discussed by Gershenfeld and Lipson, and the previous adaptation of these achievements into computer science.

Chapter 2

Prototype Systems

Over the past several years we have been working on the creation of a family of child-friendly tangible user interfaces that would serve as input devices for exploring 3D modeling and digital fabrication in an "embodied" fashion. As discussed in the first chapter, the motivations behind this work are thematically diverse, but can be distilled into an attempt to create a more intuitive, body-centric way for novices to design for 3D printing while also strengthening the user's sense of spatial translation from 3D to 2D (screen based) representations. To this end, we have created three prototypes: the UCube, an initial proof-of-concept device using simple components, SnapCAD, a more expressive and capable iteration of the UCube relying on magnetized LED circuit boards, and PopCAD - a paper-based interface addressing several of the cost and portability concerns raised by SnapCAD. These systems all communicate with versions of a companion software program running on desktop computer. This chapter describes (in chronological order) the development of these three systems, the software that interfaces with them, the motivations behind their design, and the technical work involved in their creation.

2.1 UCube

The UCube represents our first attempt to create a cooperative system of hardware and software that encapsulated and combined our beliefs about embodied cognition and the importance of accessible digital fabrication. The idea for the UCube originally came from the attempt to create a "3D Geoboard". 2.1 shows a rudimentary 2D geoboard consisting of a 3x3 grid of nails stuck into a wooden block. Simple geometries, such as the triangle shown in the referenced image, can be made by stretching rubber bands around some number of “pegs”. The geoboard invites a kind of tangible, exploratory, and embodied play that (as we discuss in Chapter 3) promotes children’s learning in powerful ways. The initial design goal was to capture the “gestalt” of the traditional 2-dimensional geoboard and extend it - into 3-dimensions, and with a computationally-enhanced interface that could translate physical manipulations on a device into a software program that could display the actions performed on the geoboard in a “meaningful” way - that is, in a way that could potentially extend spatial reasoning abilities between the 3D representations created on the device and the 2D, screen-based images displayed on the computer screen.

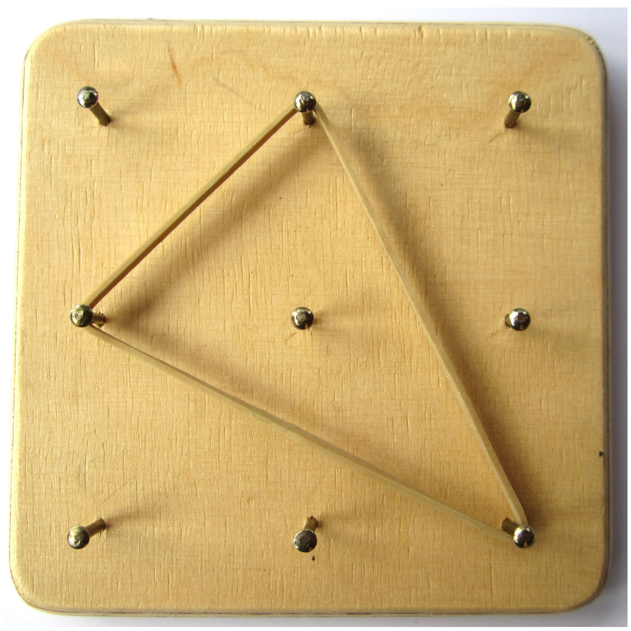


Figure 2.1: A simple 3x3 geoboard, with a rubber band stretched around several pegs, forming a triangle.

The UCube (as seen on the left in 2.2) is the initial result of this goal. The physical interface consists of a set of vertical “towers” that are placed (and optionally re-placed) onto a grid of 4x4 evenly spaced nodes or sockets, which act somewhat like the nails in the 2D geoboard. The towers themselves contain four switches placed vertically along the tower, creating a potential for

64 (4x4x4) distinct points to be activated. The towers are "plugged in" when placed into one of the 16 socket nodes, connecting them to the underlying circuitry responsible for providing power to the towers and relaying the state of each of the switches to the computer, via an Arduino Mega[93] microcontroller. Thus, when a tower is placed in a specific node on the board and a switch is flipped on, a particular (x,y,z) coordinate in three-dimensional space is activated and sent to a piece of software on the computer. An abstracted illustration of the hardware system is seen on the right in 2.3.

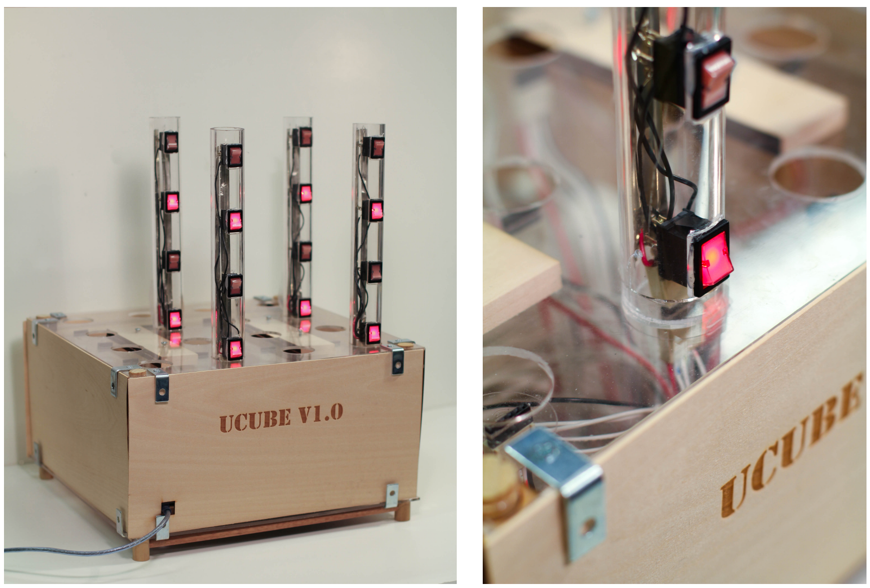


Figure 2.2: Left: The UCube device, with four towers and eight lit switches, representing (in one instance) the eight vertices of a cube. Right: A detail view of one of the towers placed into the UCube modeling board with the bottom switch lit.

Figure 2.2 shows two views of the UCube interface. The picture on the left shows the device, with four towers placed in an evenly spaced square, with one "board unit" separating each tower. The lowest and third-lowest switches on each tower are lit, marking eight active points. Thus we have eight active points, spaced evenly in such a way to describe a cube of 2 "board-units" in length if we were to take the convex hull of those points. The photo on the right gives a detailed view of the UCube hardware. A tower has been plugged into the board and its bottom-most switch turned to the "on" position.

The UCube software (discussed more thoroughly later in the chapter) takes the incoming coordinate data from the microcontroller and translates it into a real-time visualization on screen. The graphical user interface centers around a "ghosted" grid of all the potential points, with the active points being highlighted. In the first version of the software, the interface also provides a set of operations that can be performed on the set of active points in addition to normal scene manipulations like zoom and rotate. These functions are explained more thoroughly in the software section later in the chapter, but to give a brief list, include: taking the convex hull of the point set (as imagined in 2.2), creating a sequential path or knot through the active points, exporting the convex hull or knot to .STL format for 3D printing, drawing a (non-printable) spline through the active points, saving and loading a shape, and editing the vertices of a convex hull via a click-and-drag interface.

2.1.1 Technical Implementation

The physical system for our first UCube prototype, as outlined earlier, consists of a platform with a four-by-four grid of potential sites, each of which can hold one tower with four switches, thus describing a 4x4x4 array of 64 potential points. The platform structure consists of three different horizontal "layers". The top (or upper surface) layer is a clear 1/4" acrylic square, into which a four-by-four grid of circular holes has been laser cut in such a way that the towers fit snugly. This layer of clear acrylic acts as a brace to hold the towers upright, helps guide the pins from the tower into alignment with the socket into which they must be placed, and ensures that the towers themselves are resistant to being knocked over.

The next layer down holds a set of headers, six per socket (one each for power and ground, and four input lines, one for each switch), which allow the towers to "plug in" and connect to the rest of the circuit. Wires from the headers go down to the bottom layer, which holds the breadboarded circuit and Arduino Mega microcontroller[93]. The header wires connect directly to the breadboard, where each switch circuit runs through a 10KΩ resistor, and then to a digital input pin on the Arduino Mega. When plugged in, the Arduino is able to communicate (via asynchronous serial communication) the set of active switches (and corresponding coordinates) to the computer through a USB cable. 2.3 depicts a schematic diagram of the UCube hardware.

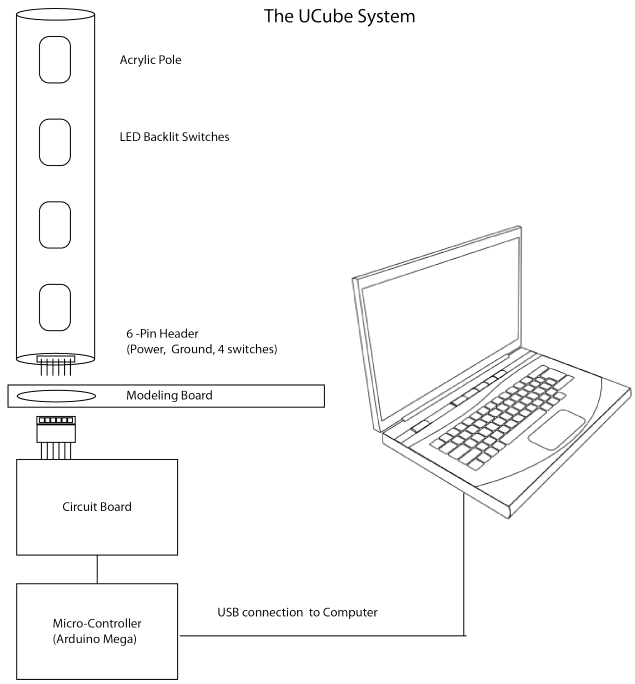


Figure 2.3: A schematic illustration of the UCube hardware.

The towers are made of transparent acrylic, cut from a 1” diameter circular tube. The towers were laser-cut in order to house the four switches and corresponding circuitry elements. Four laser-cut rectangles are sized to allow the back of each switch to be placed inside the tower while the faceplate remains on the surface. The switched are backlit with in-built red LED’s. Each switch has a 270Ω current limiting resistor soldered between two of its legs to protect the lighting element. Headers were soldered on to the power and ground pins of the switch, and connect to a strip of conductive tape affixed to the back on the inside of the tower (one can make this out somewhat in Figure 2.2). The signal line from the switch (responsible for letting the microcontroller know it has been switched) is soldered to a wire which reaches down a six-pin header at the base of the tower. As the switches are LED-backlit when active, it becomes more apparent which points are active as well as giving a more accessible “gestalt” of the shapes being modeled. The luminosity also allows for some potentially interesting applications in dimly-lit circumstances, such as modeling constellations in a classroom or planetarium: in these situations, the lights of the selected spatial points stand out especially vividly.

2.1.2 A Sample UCube Scenario

As a sample scenario, imagine that we wish to create a triangular prism solid employing the UCube. We can begin this process by selecting three points to form a triangle; then, by placing two more towers and creating the same triangular shape ”shifted over” by two units (as seen on the left of 2.4) we create the entire prism. Naturally, there might be many alternative pathways to forming the same eventual shape: for example, we might begin by placing four (or more) towers in the platform, and then experiment or fiddle with the chosen lights to approach the eventual goal of creating our prism. Alternatively, we might begin without any towers in the device at all: by placing our hands or fingers above the device, roughly indicating where the prism should be, we might then use our imagined locations as “guides”, helping us to place the necessary towers in the platform and select the correct lights for the vertices of the prism. In any event, having designed the prism using the UCube platform, and having checked that it looks like the correct shape on the computer screen (as seen in the center of 2.4), the final step is to export the shape into a format suitable for 3D printer output. The UCube software, as noted earlier, includes a feature for doing just this; and finally, we print out the prism, as shown on the right in 2.4.

2.1.3 Limitations

The astute reader will have picked up on some of the more obvious limitations of the early system: as a three-dimensional modeling device, it is quite limited in the scope of things it can effectively model, certain geometric shapes are impossible to model on an integer lattice (a dodecahedron, for instance), a 4x4x4 resolution is clearly insufficient for complex shapes, and the inability to create curved surfaces precludes most “natural” objects (such as human faces) from being represented in a life-like manner. It is thus important to differentiate this system (and the

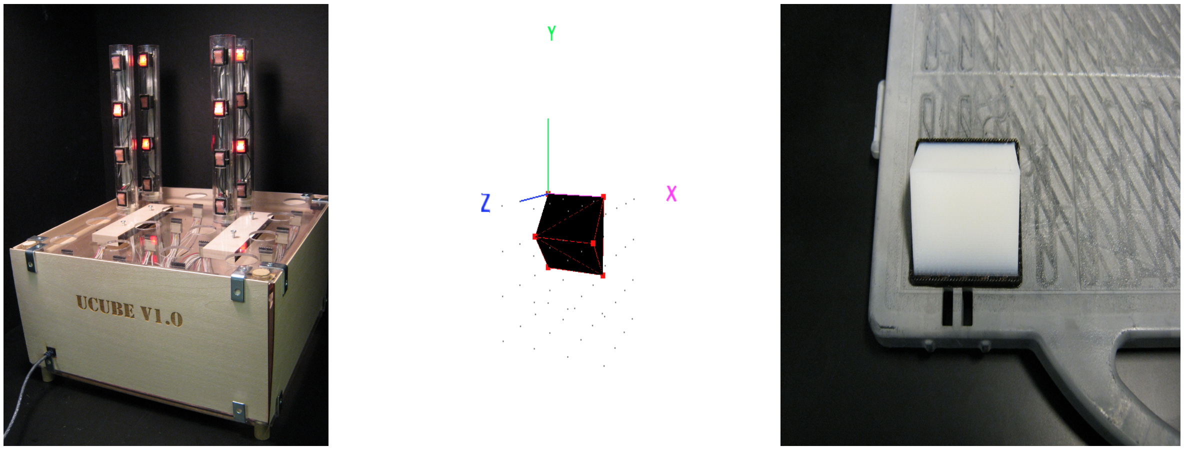


Figure 2.4: Left: The UCube device, with four towers and six lit switches, representing the six vertices of a triangular prism. Center: An early version of the UCube software, representing the convex hull formed with the six active points from the picture to the left. Right: The resultant 3D print, exported from the software to a 3D-printer friendly format.

others mentioned in this chapter) from a “professional” 3D modeling system; our focus is on ease of use for novices, to provide a visual and tactile bridge between 2D and 3D worlds, and to provide a simple way to create shapes suitable for 3D printing. Even so, this does not preclude us from attempting to make a more powerful, expressive, and stable interface to present to users.

2.2 SnapCAD

Based on the feedback and observations from the two user studies we performed with the UCube (discussed in chapter 4), a second, more powerful instantiation of the UCube has been created. Called SnapCAD (formerly known as UCube v2) this next generation device consists of a total input space of 7x7x7 points, forming 343 distinct coordinates (as opposed to the 64 points of the UCube). We focused on two main design goals with the SnapCAD: greater expressive power and greater stability in the system operation.

2.2.1 Technical Implementation

To start with system stability, then; custom printed circuit boards have replaced loose wires, the towers are designed in Rhino[119] and 3D printed to safely and securely house the towers, which are in fact printed circuit boards of their own. Instead of a rickety housing, the top layer is 1/2" acrylic which was milled on a CNC machine to precisely fit the newly designed towers. The sides and bottom are hand-crafted wood, with channels cut to allow the top and circuit board layers to easily slide in and out. The frame is not glued on one side to allow for repair and maintenance. This side is secured by custom metal brackets and screws. Each socket has its own printed circuit board, held firmly in place by zip ties around a latticed acrylic layer underneath the boards. The system has since traveled to the Denver Art Museum, the Computer Clubhouse in Lakewood, Colorado, and ridden around in the back of several cars without mishap (not to mention roughly 20 hours of user testing by eager adolescents).

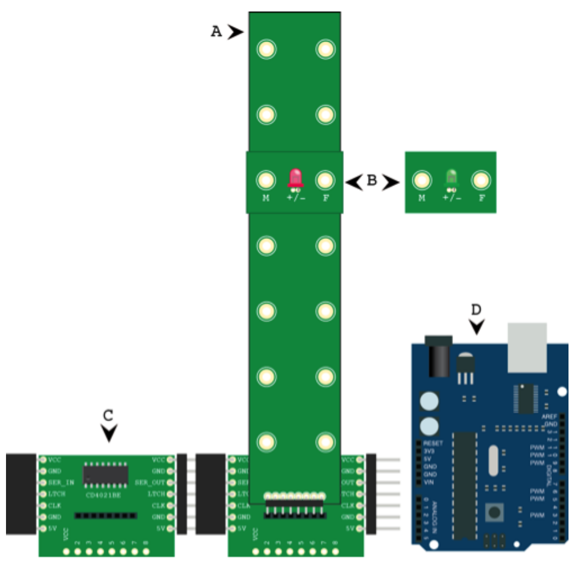


Figure 2.5: A schematic of the SnapCAD technical design, showing a sample tower (A), LED light element (B), shift register board (C) and Arduino (D). The Arduino microcontroller's role is to send coordinates (and colors) of the LED lights, once placed, to a desktop computer. A fuller description of this schematic is provided in the accompanying text.

The goal of expressiveness was met in two ways; by increasing the possible input space from 64 points to 343, and by designing a system that allowed for each point to be activated by more than one color of LED - effectively allowing for multiple shapes to be modeled at once, or multiple "players" to interact with the board at once. Both of these solutions required changes in both hardware and software from the UCube. Working on the scale of multiple hundreds of inputs necessitated the design of custom circuit boards to relay information effectively to the microcontroller. The Arduino Mega has only 54 digital input/output pins, far too few to assign each line directly to an input, even if one wanted to deal with tracking 343 i/o lines (which we most certainly did not). Instead, we designed a circuit board around an input shift register chip from Texas Instruments - the CD4021BE - that could effectively provide eight more input lines per chip and operate with the Arduino's ATMega 328 Serial Peripheral Interface (SPI) protocol, which requires only three lines from the Arduino. By breaking out the pins on the CD4021BE so that they could be chained together (by aligning the serial output of one chip to the serial input of the next chip, while also passing along the latch and clock signals, the other two lines necessary for the SPI to work)[1] . By arranging 49 of these daisy-chained boards in a 7x7 grid, we had the framework to read in from 343 inputs in real time. Only one more problem had to be solved: at around 35 connected shift registers, we exceed what is called the "fan-out" of the Arduino microcontroller - the number of connected input gates that a given pin on the Arduino to drive a current load into. To get around this problem, the clock and latch signals are put through a set of two "buffers" - in our case CD4049BE inverting hex buffers - which can used for logic level conversion (the inverting part, which we do not need, hence the second inverting buffer), but also as a "boost" to drive the signal farther. One set of buffers was enough to get our signals to the computer reliably.

This change in scale also meant rewriting most of the modeling software to effectively handle the greater expressiveness of the physical system. The astute reader may have noticed that while the CD4021BE adds eight inputs per board, our system calls for a 7x7x7 array - so what were we to do with the extra input? The dilemma actually ended up solving several problems; in the

[1] We understand this section is somewhat technical. A great introduction to using shift registers in this way can be found at: http://www.arduino.cc/en/Tutorial/ShiftIn

UCube firmware each input triggered an (X,Y,Z) coordinate to be send out the serial port - by switching to shift registers over SPI, we are limited to one character per input - either a 1 or a 0. Normally, a serial string can be delimited in software by looking for certain characters at the beginning or end of a communication, and parsed accordingly, but we only had a string of 343 0's and 1's. Given that most of the sockets would be returning a zero most of the time, by tying the 8th input line high we could count characters and check for a "1" every eighth character to ensure the serial string was correct - and since we were isolating this character anyway, it was simple to throw it away afterwards and thus be left with only the sets of seven digits describing the state of the inputs. The problem of generating the proper coordinates from the input stream was then a matter of creating a "lookup table" where the *nth* character in the input string array was the *nth* element in an array of 3D coordinates.

Figure 2.6: Left: the SnapCAD interface, showing four towers with two red LEDs each, arranged in cube-like configuration. Right: a detail of the SnapCAD hardware - the PCB tower is housed in a 3D-printed shell, which plugs into one of a chained set of shift-register boards. The LED boards snap on to the towers via conductive magnetic snaps.

The other enhancement to the expressive power of the SnapCAD design is the ability to use each socket in each tower with more than one color of LED. In in order to make this a possibility, we had to find a way to make the LEDs detachable from the tower and "swap-able" with other colors. This was achieved by soldering conductive magnetic snaps directly into the circuit boards themselves; the magnets act to both attach LEDs to the tower, but also to close a circuit and light up the LED. We used snaps and not just simple magnets because LEDs are polarized - they have only one correct orientation - and snaps have a male and female part that could be used to indicate the correct orientation. This multi-color capability can be seen in Figure 2.7.

This ability to swap out different colors of LEDs not only results in the ability to represent multiple shapes at once, but for the SnapCAD to become a platform for all manner of multi-player interactions (e.g. games, puzzles, shape matching contests), with each "player" assigned a unique color. To this end, we have created a simple "3D Tic-Tac-Toe" implementation on the SnapCAD, and imagined a sample scenario, explained in the next section. The SnapCAD version of the software includes this "multi-payer" ability as well as some additional changes that include supporting multiple but separate convex hulls of different colors, the ability to create and export shapes created from the minimal spanning tree of a set of input points, and the ability to adjust the width of the segments in the knot/path and minimal spanning tree modes. The click-and-drag editing mode includes the knot/sequential path and minimal spanning tree modes as well as the convex hull mode. We also adjusted the knot-forming algorithm to handle paths that cross or self-intersect, as well as providing a "close knot" button to complete a circuit in a shape, allowing for even more kinds of 3D-printable objects.

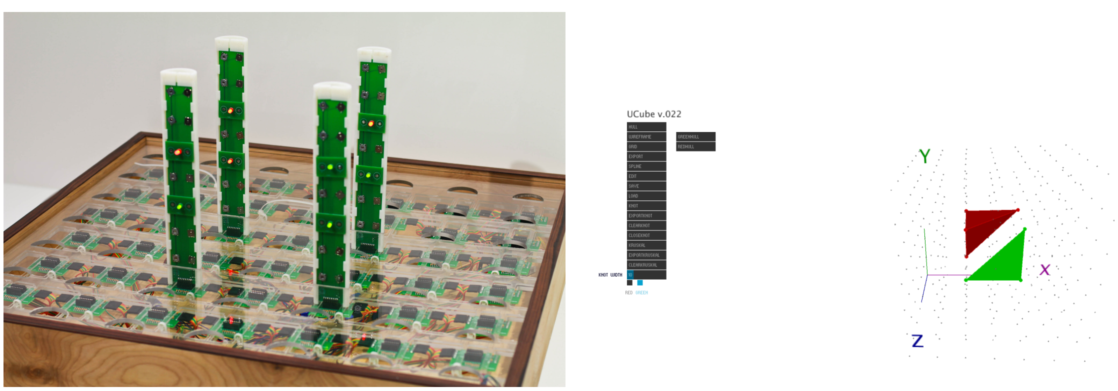


Figure 2.7: Left: The SnapCAD software showing two convex hulls of different colors. Right: the SnapCAD software showing a minimal spanning tree model.

A note on the 7^3 array in SnapCAD: in our user studies with UCube, we noticed that users often encountered initial difficulties when required to “find a middle” in the shape they were attempting to model, given an even number of total grid spaces. For example, to model a pyramid on on a 4x4x4 grid, one needs to construct a 3x3 subset of the 4x4 grid, using the middle point within the 3x3 set as the top of the pyramid. This influenced our decision to create an odd-numbered layout, creating a more “natural” middle point in the hardware.

2.2.2 A Sample (Red/Green Player) Strategy Game for SnapCAD

In this use case, we make use of the two-color capability of the SnapCAD to suggest a hypothetical game, or genre of game, that could be created with the system. The imagined game in question is a geometric strategy game between two players, “Red” and “Green”. At the outset of the game, each player is given four lights of her own color; the two of them are told to place their lights at the eight corners of a cube in the positions shown in the photograph shown in Figure 2.8 on the upper right.

Now, the computer could display the convex hull of the present set of lights (a cube), as shown in Figure 2.8 on the upper left; and then (in our scenario) the computer tells the Green player to move one of her lights to create the new convex hull shown at the bottom-left of Figure 2.8. Thus, the Green player’s job is to change the “cube” hull to the new hull with one move of one green light. A correct answer to this challenge is shown in the photograph of Figure 2.8 at the bottom-right; and if the Green player makes this correct move, the Red player is now given the (current) convex hull and yet another hull that could be created with one move of a red light. In this fashion, the two players take turns moving lights of their own color to produce a new overall configuration of lights at every step, until one player fails to solve the current challenge, at which point the game is over. There are, of course, many variants or extensions of this game that could be imagined (for instance, a player might be asked to shift two lights, or to add a new additional light in her color, to create a new convex hull). The purpose of this example is simply to show that, with the inclusion of two available colors for spatial points with SnapCAD, a sizable potential landscape of geometric activities and puzzles becomes feasible.

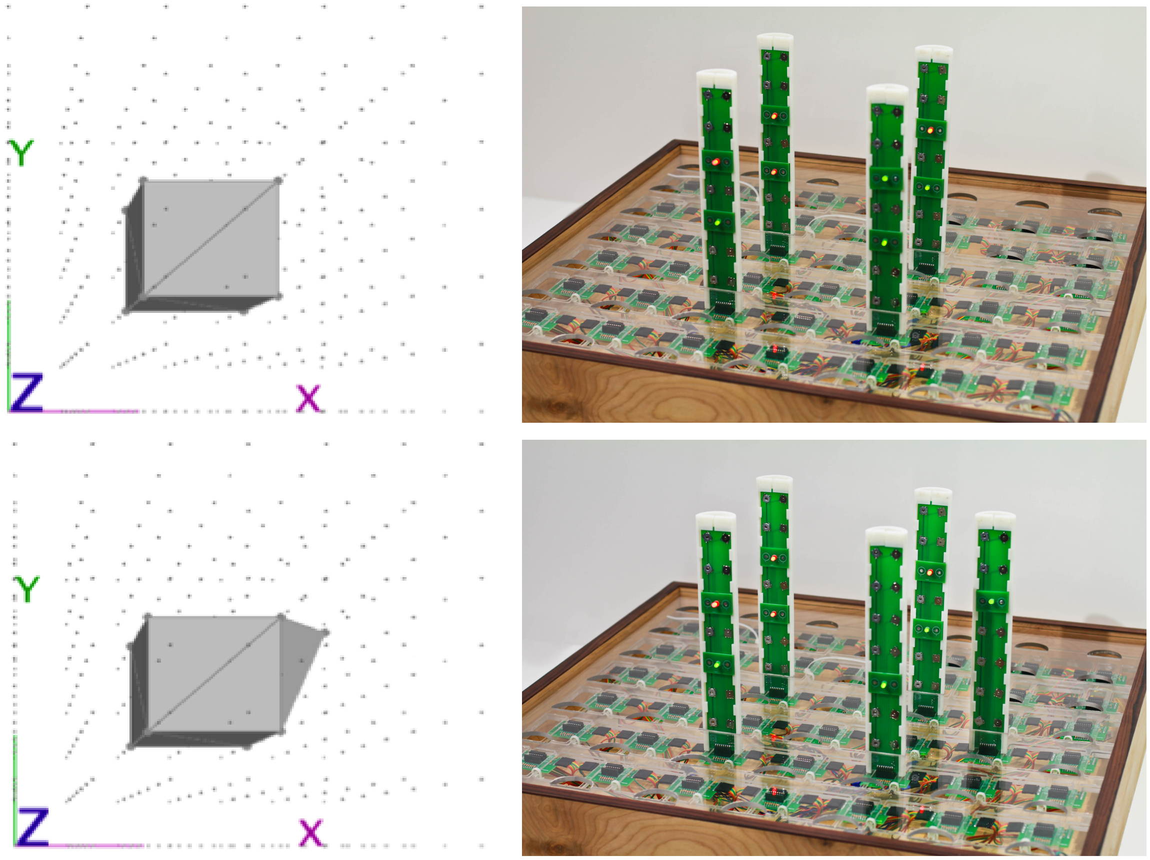


Figure 2.8: Left: The SnapCAD software showing two convex hulls of different colors. Right: the SnapCAD software showing a minimal spanning tree model.

2.3 PopCAD

Our motivations for creating a third, alternative interface to the UCube and SnapCAD stem from the desire to explore this intellectual space more generally; it is far more interesting to discuss a **class** of tangible interfaces for scaffolding digital fabrication than it is to discuss a singular device. To this end, we looked at some of the weaknesses of SnapCAD and towards technologies we had yet to explore. While SnapCAD can admirably perform a number of modeling tasks, it was always envisioned as one device amongst an "ecosystem" of next generation fabrication tools. It has strengths, but obvious weaknesses as well; in particular, the SnapCAD hardware was expensive to produce, and so would be a difficult proposition for some schools or fab labs to produce or purchase; it is also rather unwieldy and unportable - it is moderately heavy, fairly large (over 30 inches square), and has many separate pieces (like the towers and LED boards) that could break or go missing. Thus, an interface with cheaper and more portable materials was desirable.

To address these issues we chose to build a pop-up book combining traditional paper-crafts and paper-friendly electronic components such as copper or conductive tape and conductive inks. In recent years, revolutionary work has been done in combining electronics and paper crafting[111][94], leading to new techniques and new uses for traditional materials. Paper is inexpensive (especially when compared to circuit boards), light, and easily portable, making it an ideal material choice for a device that would not suffer the same limitations present in the SnapCAD. Although we often think of "paper" as a rather static material, there are in fact many variations in the size, weight, color, transparency, and composition of contemporary paper products. We will cover the two paper-based prototypes we created in this vein, dubbed "PopCAD v1" and "PopCAD v2".

2.3.1 PopCAD v1

For the initial prototype, we use a simple construction paper as it provides a balance between strength and flexibility as well as having a consistency well-suited to laser etching and cutting. The pop-up book (named PopCAD) has a 3x3x3 array of 27 points which are evenly spaced three inches apart on a 12" x 18" paper surface. The book folds on a single center crease making the closed footprint of the book roughly 12" x 9".

Each tower has a copper tape circuit consisting of three LEDs on the front face and three corresponding capacitive touch sensors on the left face. The copper tape acts as a paper-friendly conductive material to connect the electronic components together much like traditional wire. The LEDs are soldered onto the copper tape for greater stability. The capacitive sensors are simply a piece of copper tape which is connected to a pin on a microcontroller (in the first version, this is an Arduino Mega Pro). By bringing the internal pull-up resistor connected to the pin "LOW" (to ground) and then timing how long it takes to get back to a "HIGH" state we can tell if the connection is being influenced by a capacitive force. For example, if there is no interference on the

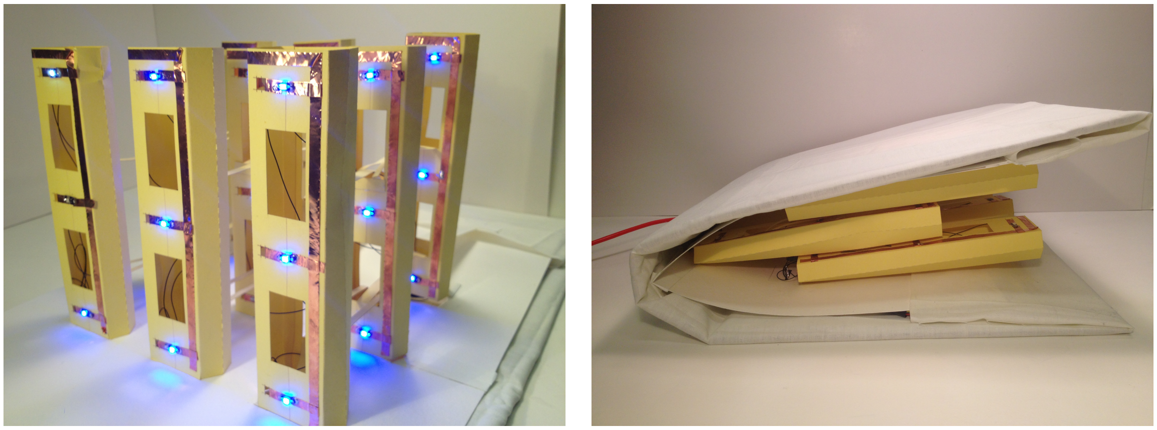


Figure 2.9: Two views of the first pop-up book prototype, showing the interface in both open and closed states.

circuit, the timer will normally only get to “1” before the resistor is back to a HIGH state; if a finger is placed on the copper tape, the reading will be much higher (typically around “17”). Based on this change, we can detect which switch was touched and toggle the associated LED on or off. The hollow interior of each paper tower is used to solder thin 30-gauge wire to the three LEDs, the three switches, and ground. These seven wires are soldered to a row of headers that stick through the bottom of the first layer of the pop-up book. Wires are then run along the backside of the top layer of paper from these headers to the microcontroller. The entire circuit in then encased in a cloth-covered cardboard binder that acts as a book cover as well as a means to protect and hide the electronics.

2.3.2 PopCAD v2

Although the first PopCAD iteration was a fully-functional prototype, as we approached user testing with the PopCAD it became apparent that there were several compelling reasons to iterate on the original design. Through a few informal user evaluations as well as our own reflections on the device, we identified several key issues that could be improved upon: (a) the paper engineering design, (b) the structural integrity of the book as a whole, and (c) the lack of “paper-ness” with respect to the circuitry and electrical components of the design.

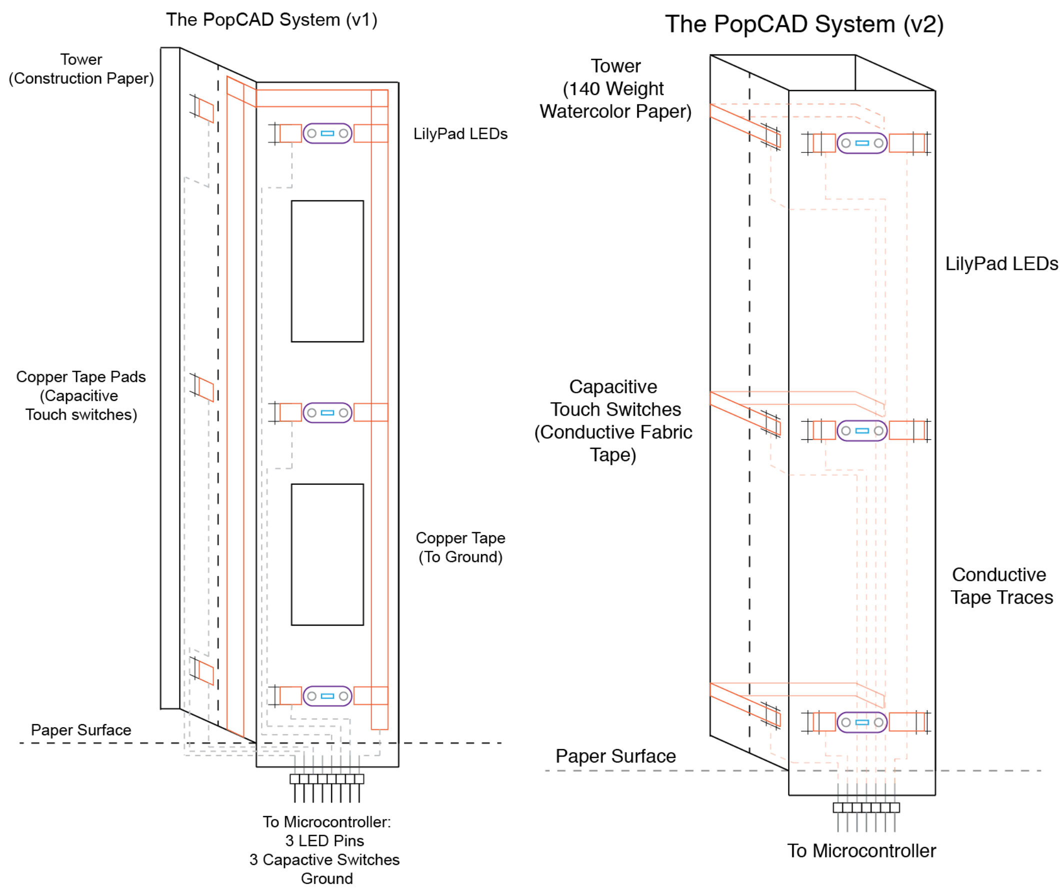


Figure 2.10: The two PopCAD designs side-by-side: PopCAD v1 (left) uses copper tape and 30 gauge wire for the paper circuit, while PopCAD v2 (right) uses fabric-based conductive tape without needing any wires.

To start with the first point above, then: a look at the initial design of the pop-up mechanism reveals the presence of horizontal paper "struts" connecting the towers in the middle column (along the center crease) to their counterparts on either side. These struts were necessary in order to generate a pop-up motion from the middle of a book and force each tower upright. The mechanism was successful; however, the struts directly interfered with the ability to reach many of the conductive tape switches. In version two of the PopCAD, the struts were removed in favor of a pull-tab system whereby each row of towers is raised and lowered by a tab at the front of the row (the pull tab system can be seen in Figure 2.11).

We were concerned about the overall stability of the first prototype; lights would sometimes

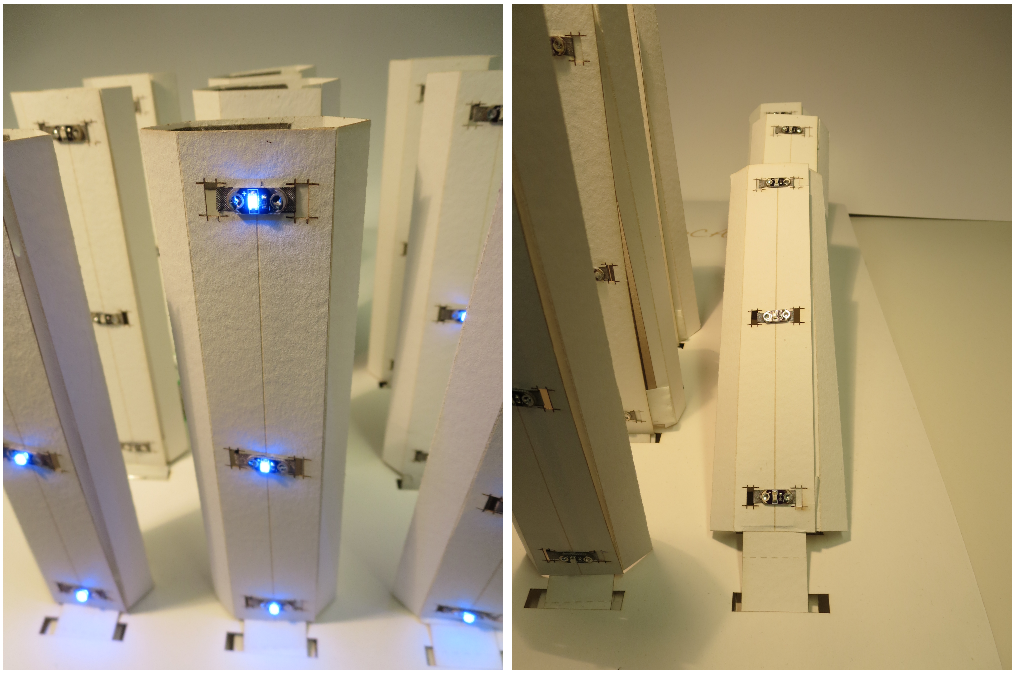


Figure 2.11: Two views of PopCADv2 design: with towers raised and LEDs lit (left), and with the rightmost column of towers laid flat (right).

fail to operate properly, the horizontal struts kept breaking, parts of the towers were weak, certain points in the wiring were weak, and the opening and closing of book (and thus the folding of the towers) put enough strain on the circuitry that we were concerned whether it would survive a user study. The second prototype addresses these issues in several ways: first, we use a heavy watercolor paper (140 weight) for all the paper engineering, making the towers and the pull-tab mechanisms more resilient to repetitive use. Second, we replace the copper tape, which has a tendency to break over heavy creases, with conductive fabric tape, which resists repeated creasing much better, and finally, we made adjustments to lessen the strain on the towers and the circuitry, by minimizing stress points and reinforcing known weak spots.

In the first prototype, we still used traditional jumper wires from the Arduino to connect to the headers beneath the towers, and 30 gauge (still traditional) wire inside of the towers to connect from the headers to the LEDs and copper tape. Admittedly, this does not feel very "book-like", or in the spirit of faithfully exploring paper-based electronics. PopCAD v2 has no traditional "wires"

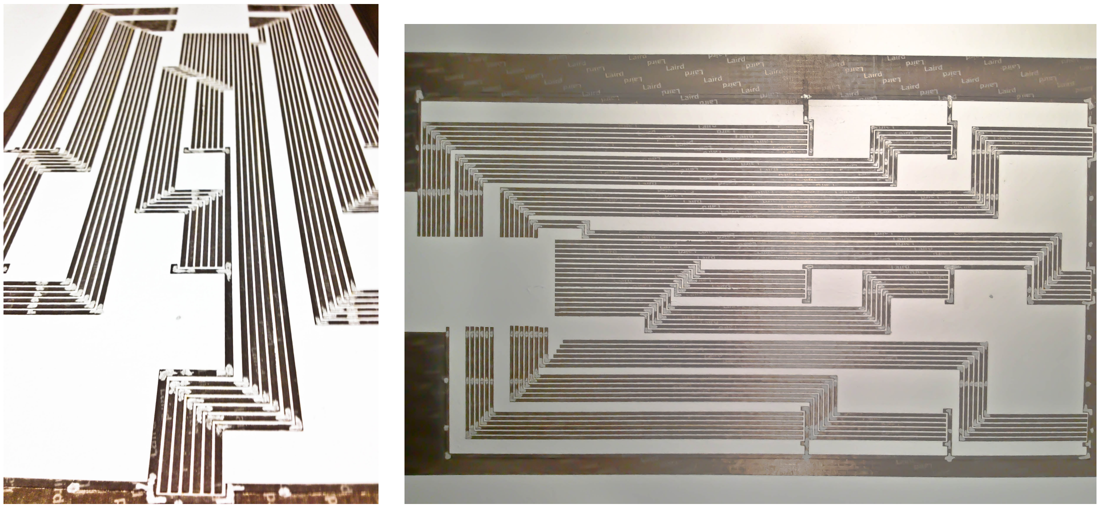


Figure 2.12: Two views of the conductive tape circuit connecting the paper towers to the Arduino Mega microntroller. The circuit was constructed by laser cutting a design through conductive tape (but not through the paper beneath it) and removing the excess material.

at all. Instead, we use a fabric-based conductive tape, which, besides laying flush (unlike wires) and feeling more like paper than copper tape, the fabric tape is (unlike copper tape) able to be used in the laser cutter in our lab. Figure 2.12 shows two views of the PopCAD v2 circuit, constructed by placing strips of the conductive tape on watercolor paper, laser etching a circuit diagram through the tape (but not through the paper), and peeling away the excess. This technique allows us to create a precise yet completely flat circuit layout. The accuracy of this method permits the Arduino Pro Mega (a thinner version of the regular Arduino Mega) to be affixed directly onto the paper. The towers use this conductive fabric for the capacitive touch sensors as well as material to solder the LEDs to, eliminating all the standard wires from our design.

The software for the PopCAD retains all the algorithmic capabilities of the SnapCAD version of the software (convex hull, path, minimal spanning tree), but as the LEDs are affixed to the towers, we lose the ability for multiple player functions. However, it should be noted that this was in part intentional; we did not want any loose parts that could easily break, get lost, or other make the device less portable. As it stands, the PopCAD (unlike its predecessors) is self-contained as one piece, is small and light enough to be carried with one hand, and is considerably less expensive (and less time-consuming) to produce. We offered sample use cases for the previous devices that involved descriptions of various technical features; for the PopCAD it seems more appropriate to instead paint a more general user scenario that speaks to the intent behind PopCAD's design. Imagine an art teacher, girl scout troop leader, hackerspace or FabLab member, or any number of educators either wondering how to get into this "3D printing thing" or who have a Makerbot sitting in a corner gathering dust. They find plans online (perhaps on Instructables[69] or some similar DIY-oriented forum) for a relatively cheap, portable device that they could not only turn into a group project to build with their kids, but once built would offer a new way to introduce children to 3D modeling and 3D printing in a completely new way. The democratization of digital fabrication technology is (as stated earlier) a core goal of this work, and in many ways the PopCAD is the device that embodies this ideal the most. So while it may be less expressive or powerful in some ways (though as we point out in later chapters, this can sometimes be an advantage), the PopCAD does have a place in our suite of devices.

2.4 Software

The software for the aforementioned devices utilizes the Processing Serial library to read in the active coordinates from an Arduino microcontroller; it then displays those coordinates on-screen as larger points against a "ghosted" grid of grey dots. The exact methods used to achieve this varied by device, and were detailed in the device-specific sections above. This on-screen model can be manipulated in a number of ways. Clicking and dragging along any axis rotates the model, as does the use of the arrow keys on the keyboard. Holding the shift key while performing either action moves the entire model around the screen (essentially re-centering it). The "control" key plus an up or down arrow key zooms in or out along the z-axis. In addition to camera movements, there are a limited number of functions represented by a simple graphical user interface which aid and expand the modeling capabilities of the connected device.

For a brief overview of these functions, then: there are three ways of interpreting the active set of points on the connected device: by taking the convex hull of the input set, by connecting each point sequentially with a 3D path, and by connecting the set according to a minimal spanning tree algorithm (more on these modes soon). There is a single “export” button that will take whatever the active shape is, no matter the mode (if there is one) and generate a stereolithography file (.STL), the standard file format for 3D printing, although .STL files can also be opened by more sophisticated 3D modeling software, providing the possibility for the software to be used as a sort of “sketchpad” for rough ideas or shape that can refined afterwards. There is a “close path” button which will connect the first and last segments in an open path (e.g. in constructing a square path, after the fourth point has been placed, there will still be an open side of the square - pressing this button will complete the square). There is an “edit” mode, whereby the real-time input from the device is suspended in favor of being able to click-and-drag the active points around with the mouse. Consequently, there is a reset button, in the case that the user wishes to “snap back” to the normalized integer lattice. There is also a slider element entitled “path width” that will dynamically adjust each segment or branch width when in path or tree mode, making the segments “skinnier” or “fatter”. Finally, two minor aesthetic options - the “wireframe” button will turn off the “fill” of the shape, showing the outline stroke with a transparent fill, while the “grid” button will toggle the visibility of the ghosted grid of non-active points.

As a guiding heuristic for our software design, it should be noted that the device software is intentionally minimal. Our aim is not to produce another sophisticated software modeling program - there are plenty of good ones available already. Instead, the software is meant to aid the user in clarifying their physical actions with the physical device, while maintaining a low barrier to entry, a great possibility of expressiveness, and multiple ways of approaching any given exercise - a trifecta of design heuristics often referred to by Resnick (and others) as low floors, high ceilings, and wide walls[118].

2.4.1 Modeling Modes

This section will describe the three main modeling modes of the software, with particular attention to explaining the methods by which they form shapes, the algorithms behind how they operate, and the modeling domains for which they are particularly suited.

2.4.1.1 Convex Hull: Creating Polyhedral Forms

In observing a set of lights placed on an integer lattice in 3-space, one of the first mental images we thought of was to take the convex hull of that set and create a solid polyhedral form. One may think of the 2-dimensional convex hull as the operation performed on the 2D geoboard mentioned earlier in the chapter; given a randomly scattered set of nails in a board, the convex hull of those nails will be equivalent to a rubber band that stretched around all the points. That is, the minimal form that includes all of the line segments connecting each pair of points, as the rubber band forms a straight line between those nails on the hull as opposed to curving inward (thus the minimal shape of line segments as opposed to area). In three dimensions, this becomes a convex polyhedron instead. Many popular 2-dimensional convex hull algorithms originated in the early 1970's (e.g. Jarvis March/Gift Wrapping, Graham Scan), while a 3-dimensional solution was published in 1977[107] and popularized by the same author in the book, **Computational Geometry: An Introduction**[108].

The version implemented in our program is a derivative of the work presented in [19], and adapted from the implementation at [88] which combines a 2D Quickhull Algorithm with a general dimension Beneath-Beyond Algorithm to achieve a general dimension convex hull solution. In brief, the strategy works as follows: (a) from a given set of input points, where the coordinates are known, create an initial 3-simplex (tetrahedron) - from the min/max points in along each dimension (x,y, and z), and add the four faces of the tetrahedron to the stack, (b) Pop a face from the stack and get the point most distant to that face, (c) Find all faces adjacent to the selected face, find the horizon edges of the adjunct faces and extrude the shape along those edges to the selected point. (d) Put the newly discovered faces on the stack and repeat from (b) until all points have been accounted for.

When the hull mode is active in the software, the coordinates on the connected device must be sent to the hull construction methods each time a point is added or removed to ensure that the hull remains accurate in real-time. The worst case for this algorithm is $\Omega(n^2)$, although in practice is not worse than $\Omega(n\ log\ n)$. Figure 2.13 shows two screenshots of a "before and after" convex hull computation in the software in which the picture on the left in simply displaying the active set of points, while the figure on the right shows the interpretation of those points after clicking the convex hull button.

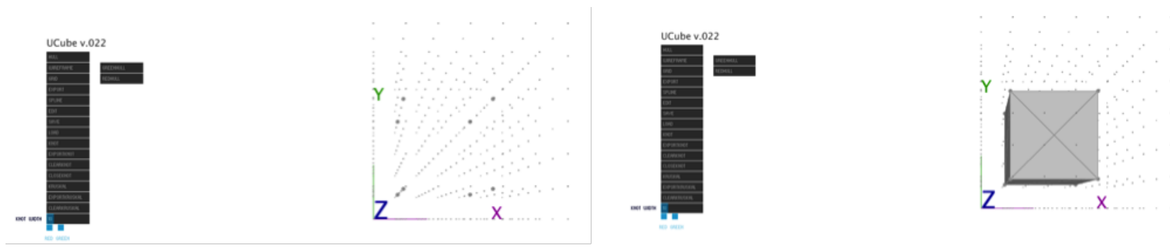


Figure 2.13: Two screen views (left and right) of the device software, illustrating the way in which the software displays the convex hull of a cube. Left: The set of eight input points, before the "Hull" button has been pressed. Right: The resulting convex hull, forming a cube from the input points.

Polyhedral forms obviously have a long history not only in modeling, but in geometry (the Platonic solids), architecture (the Pyramids of Egypt), and numerous other disciplines over the ages (building blocks, paper crafts, etc.). Though not all the Platonic solids can be modeled naturally (i.e. without edit mode) on our devices, certainly pyramids can be, as well as other common convex polyhedral shapes (e.g. a canonical "house" consisting of a cube with a tetrahedron sharing the cube's top face). Some examples are shown in Figure 2.14 below.

2.4.1.2 Paths: Creating Linear Forms and Knots

In the convex hull examples in the previous paragraphs, we have not made use of the fact that the software samples selected points in real time: thus, when a user adds or subtracts a point in space, that change is registered immediately in the desktop software. What this means is that the user can exploit not only the overall set of selected points, but can also make use of the **order** in which those points are selected. A sequence of selected points need not represent only vertices

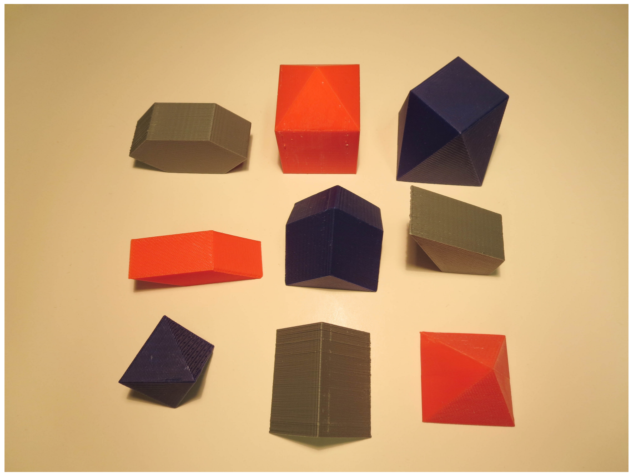


Figure 2.14: A collection of 3D printed shapes modeled using the convex hull mode in the software with the devices mentioned earlier in this chapter.

of a solid; it can also represent a path over time in 3D space. Figure 2.15 shows several sample projects based on this idea. Here, the software has been employed to read points as successive positions of various routes through 3-space. The resulting paths have been printed out on a 3D printer.

In some cases, the path is closed, finishing at the same location where it started; the path printed out at center in red in Figure 2.15 is in fact a well-known mathematical form, a trefoil knot. (It may be worth mentioning here that such a knotted form would be rather tricky to create in standard 3D modeling software, but the form can be created "by hand" with our devices, selecting light positions in space along the path of the knot.)

To briefly explain the workings of this mode, then: each point on the device is stored, in order, in an array. Once the second point of the path is added (one point a path does not make), each point is then "exploded" into a cube, centered on the point. The size of each cube is controlled by the "path width" slider, so the single original point generates 8 points, offset by the current value of the path width slider (e.g., Point p1 = new Point(x + offset, y + offset, z + offset);). The algorithm then takes the cube of points associated with each pair of connected points (e.g., points 0 and 1, 1 and 2, 2 and 3, etc.) and runs the convex hull algorithm discussed earlier, generating

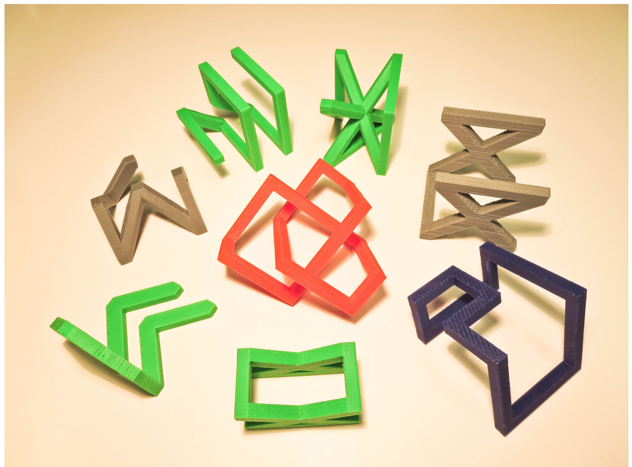


Figure 2.15: A collection of paths modeled on our devices using the path mode in the software, exported from the software and 3D printed in our lab. The red shape in the middle may be recognizable as a traditional trefoil knot.

a sort of rectangular prism between the two cubes. By connecting each new point to the previous one, a trail of rectangular prisms is generated between the points specified, in the order in which the user placed them. Figure 2.16 shows a trefoil knot created with the path mode. By highlighting the strokes in red, one can see that each point is in fact surrounded by a cube, while each cube is connected to its neighbors by rectangular prisms.

The utility in creating a mode like path is fairly self-explanatory; it allows for a vast number of shapes to be modeled, of a wholly different class of objects as the traditional polyhedral style of the convex hull output. As we will discuss in greater detail later, this method was popular with children who used it; in some ways it is akin to writing or drawing, albeit in 3D, where once you have a pen on paper, a line will follow wherever you move your hand. The path mode allows for the creation (or close approximation) of most English letters and numbers, common symbols (like stars), and 3D outlines of normally 2D geometric shapes, like triangles and rectangles.

2.4.1.3 Points as “Blocks”: Creating Non-Convex Polyhedral Forms

Aside from the convex hull and path operations noted above, the device allows for multiple different semantics for spatial locations. For example, in the path mode previously mentioned, if we

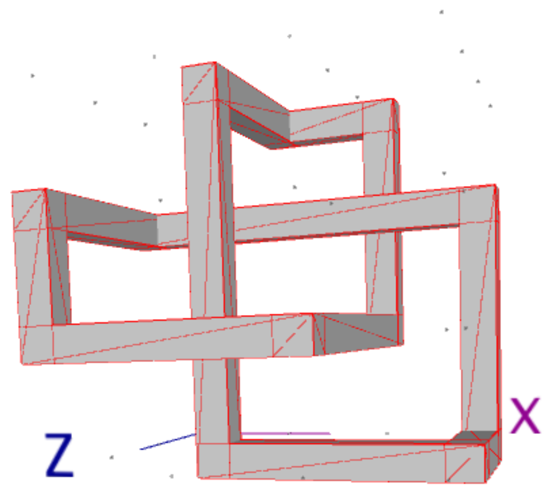
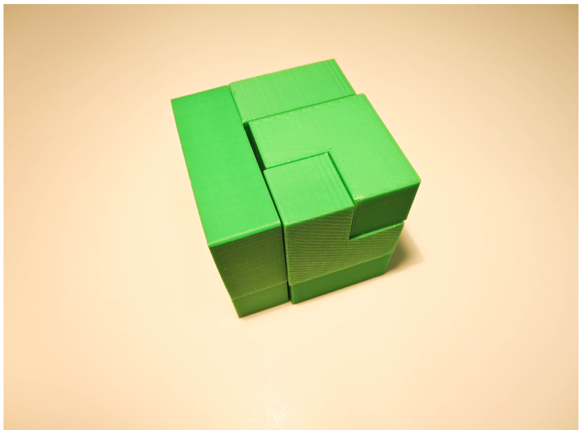


Figure 2.16: A trefoil knot, as modeled on the UCube version of the software. The outlines (strokes) of the knot have been highlighted in red to show the manner in which the software constructs the path; points are expanded into cubes, and adjacent pairs of cubes and then connected with rectangular prisms (the convex hull of two separated cubes).

choose to construct paths with an edge-length of one "interval unit" of the given device - achieved by putting the software in path mode and setting the "path width" slider to one-half of the distance between points - each cube created will fit perfectly next to its neighbors, turning each point into a sort of "block" . In this way, selecting (say) four successive light locations along the length of one tower, then, one could specify a rectangular prism. Likewise, by selecting three point locations in an "L" form, one could specify the non-convex polyhedral form seen at the far left of Figure 2.17 below.

This technique for using the path mode has some interesting advantages in terms of the physical properties of the paths produced in this manner. For those readers interested in recreational mathematics, the shapes shown in Figure 2.17 will be recognizable as the component pieces of the "Soma" puzzle; these pieces can be arranged together to form a larger cube (also shown in 2.17). The software could be employed with any of the devices in similar fashion to produce many such dissection-type puzzles, building blocks, or other for other domains where interlocking, block-like



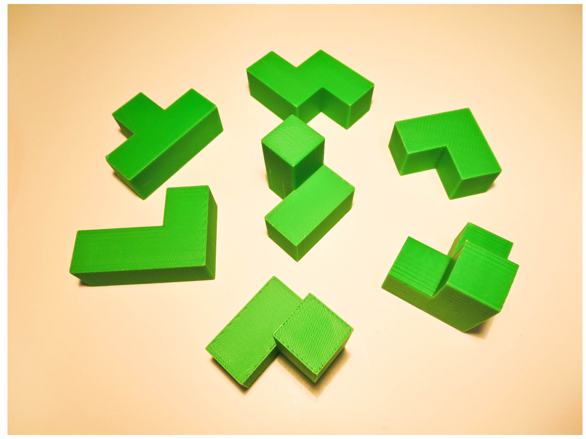


Figure 2.17: A set of non-convex polyhedral forms modeled on the UCube, which constitute the well-known “Soma Cube” puzzle, shown assembled on the left with the individual shapes laid out on the right.

shapes may be useful: architectural mockups, model train environments, real-life Tetris, and a myriad more.

2.4.1.4 Point Clouds: Creating Minimal Spanning Trees

Instead of interpreting points as vertices of a solid (as in the convex hull examples) or as the successive stations of a temporal path (as in the “path” examples above), we could in fact simply treat our set of points as just what they are namely, a set of points. Starting with this interpretation, we might produce a form such as a minimal spanning tree of the set of points (a set of edges of minimal total length connecting all the points). Figure 2.18 shows several examples of forms created this way; one immediately grasps the variance and complexity that this mode is capable of. The yellow “jack” in the middle of 2.18 is the product of nine points, eight of which form the equidistant vertices of a cube (or what would form a cube if the software were in convex hull mode), with the last point perfectly centered in the middle, effectively “bending” the rest of the graph segments in to meet it.

As with the convex hull, the minimum spanning tree is a well-defined, extensively studied algorithm in computer science and mathematics. Given a set of points on a graph, the minimal

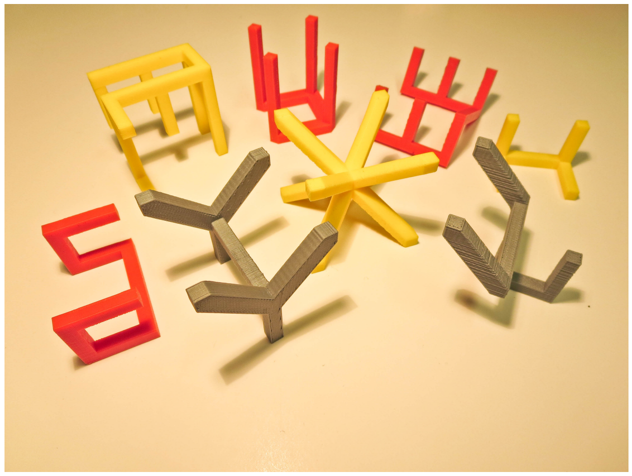


Figure 2.18: Several examples of models produced using the minimal spanning tree mode in our software, exported, and printed out on a 3D printer.

spanning tree will be a solution (possibly more than one) that connects each point on the graph, without cycles (returning to a point already in the tree), and with the minimal possible value of some "cost" variable, often defined as the sum of "weights" of the connected edges in the tree. One may think of the minimal spanning tree like constructing a subway system, where all the stations need to connect and the length of track should be minimized to keep construction costs as low as possible.

The first algorithm for finding the minimum spanning tree was derived by a Czech scientist, Otakar Borůvka in the late 1920's[24], for the purpose of planning electric distribution networks. There are two popular algorithms used today, Prim's and Kruskal's both of which are considered "greedy" (by iteratively choosing the locally optimal edge to determine the spanning tree) and run in polynomial time. Our software uses an implementation of Kruskal's algorithm, whereby Euclidean distance between two points on the graph is used as that connecting edge's weight. Kruskal's algorithm, first described in 1956[81], starts by taking a set of each vertex (thought of as separate trees) and a set of all the possible edges in the graph (with their corresponding weights), then iteratively removes the edge with the lowest weight from the set of edges and adds it to the set of vertices, connecting two of these trees into one, until there is only one tree left from the original set of vertices (or we run out of edges to pull from). If there is only one tree left in the vertex set, then that tree represents the minimal spanning tree. It is, of course, possible to have more than one minimal spanning tree for a given graph however.

In our software, we run Kruskal's algorithm whenever a point is added or removed in "tree" mode. The set of points is sent as inputs, the edges and edge distances are calculated, the algorithm is run, and returns a list of connected edges, the set of which is the minimal spanning tree. This list of edges is treated in much the same way as the points in "path" mode: each edge has two point coordinates, both of which are "exploded" into cubes centered on the point, and then the set of two cubes (16 points) are sent to the convex hull algorithm, creating a 3D rectangular prism between the two cubes.

Including the minimal spanning tree mode is an interesting departure from the convex hull and path modes; it is not easily explained to the novice designer, nor does it have the sort of intuitive relationship to the set of active lights as the other modes do. The addition or subtraction of a single point can radically alter the resultant spanning tree in (sometimes) unexpected ways - not so with the convex hull or path modes. However, it is this lack of immediate understanding and the element of unexpectedness that makes this mode a good fit for the kind of devices we make. The real-time adjustments of the software in combination with the exploratory nature of the devices makes the tree mode highly engaging (in our observations, explained in full later on). The ability to quickly add or remove points from the graph is a feature unique to our devices and allows for a quick way to "check and see" different combinations of points and strategies, while being able to look between the device and the software and start to draw some conclusions about how their actions affect the shapes being displayed.

2.4.2 Other Software Functionality

The software modeling modes mentioned in the last section set the stage for the types of figures that can be constructed with our devices, the software has additional features that are crucial to the overall purpose of the system. This section will go over the operation and methodology of the most important of those: the software's “Edit” mode and the “Export” feature, allowing figures to be saved in a 3D-printer friendly format.

2.4.2.1 Edit Mode

In order to (partially) address the “inflexibility” inherent in having points and thus shapes confined to the integer lattice, we developed a way in the software to “edit” the points by putting the software into a special mode that freezes the serial input from the connected device and allows the user to click-and-drag points off their “hardware defined” locations. The edit mode affects all three of the modeling modes mentioned above, so the user can how the edits they make change each modeling algorithm. We also provide a “reset” button as a way to “snap” back to the original grid of points.

The mode works by combining several pieces of functionality that work together to keep track of the cursor position (to detect if it is hovering over a point) and its click-state, track the relative position of the point as it is being moved, and relay that position information to the data structures responsible for the different modeling modes - all in real time.

Figure 2.19 shows a four-step sequence of screen shots using the edit mode to alter a shape: (upper left) six points have been lighted on the PopCAD and are reflected as simple points in the software; (upper right) the user has selected “tree” mode, taking the minimal spanning tree of the six points, forming a sort of “H” pattern; (lower left) the user selects the “edit mode” button, freezing the serial input and initiating the click-and-drag editing ability; (lower right) the user has dragged each of the four “corner” points outward, altering the original shape into something new, impossible to model using only the “raw” points available on the device.

2.4.2.2 Stereolithography Export

One of the most important functions of the software is to make a user's creations into easily 3D-printable shapes. Many complex 3D software programs allow for export into sterolithography format (.STL), which is the common input format for 3D printer software programs, however, these

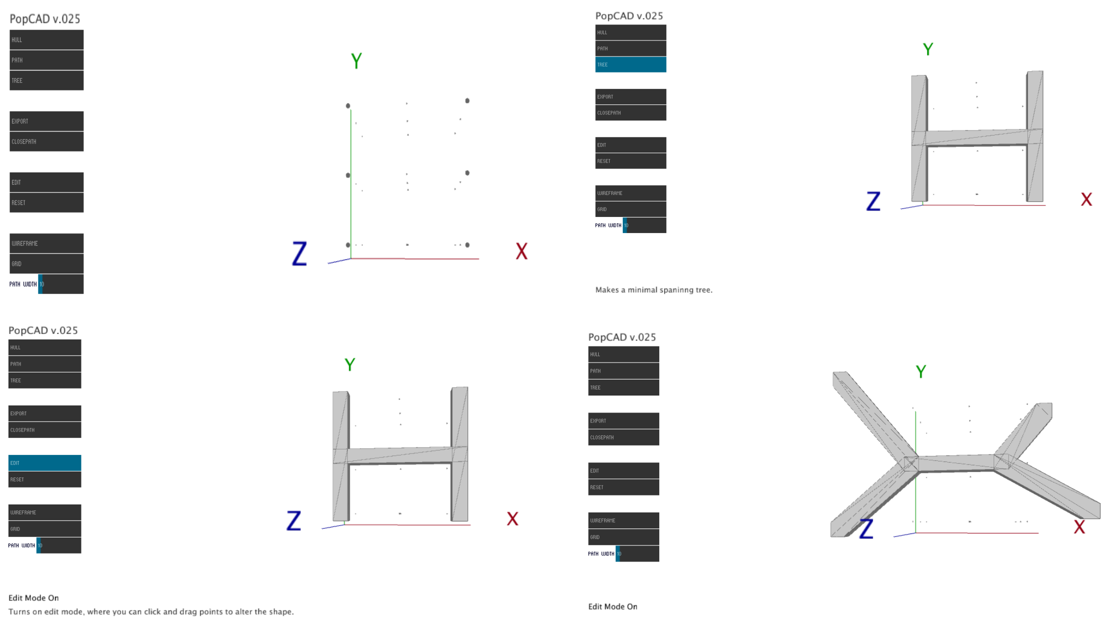


Figure 2.19: A four step sequence showing the operation of the edit mode: (upper left) six unaltered points; (upper right) the points form an "H" shape with tree mode selected; (lower left) the selection of edit mode; (lower right) the edited shape, with the corners of the original shape extended outward.

programs rarely check to ensure that the produced file will actually be printable; many "modelable" shapes will cause errors in 3D printer software - lines, 2D shapes, shapes within shapes, shapes with gaps between faces - and on and on. Our software also exports into .STL format, but takes great pains to ensure that any exported file will print without error.

The export function in our software deals with models formed from all three modes simply by keeping track of the active mode and choosing the correct export method accordingly. The export process is similar for each type of shape: since each shape is actually constructed of one or more convex hulls, the array of 3D vectors describing (in order) each triangulated face of the hull (or hulls) is added to a triangle mesh, which takes in all the faces, flips the Y axis values for each coordinate (because in the Processing environment, (0,0) is in the upper left), flips the vertex order, which corrects problems with sliceform errors (in 3D printing software) resulting from the face normal vectors facing the wrong way, then adds all the faces to an .STL object, which outputs a series of triangles in an .STL file the describes the object.

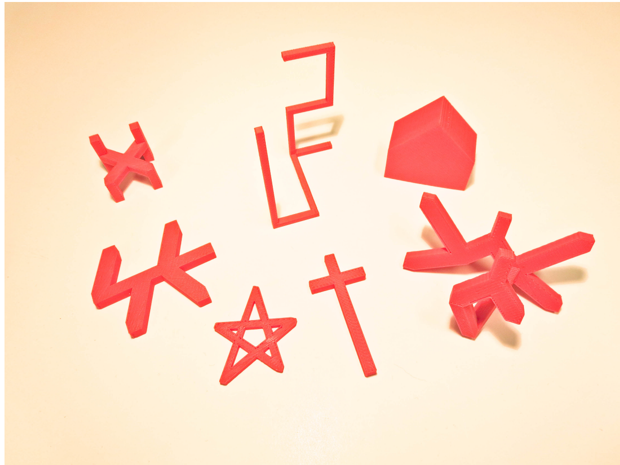


Figure 2.20: Several of the shapes modeled on the PopCAD and SnapCAD devices by novices designers (most of them without any previous 3D modeling expertise) from one of the user studies we performed.

Creating a "novice-proof" stereolithography export (all of the above computation happens with one click, even the file naming) is crucial to the raison d'être of our work - to democratize the process of designing meaningful, personalized objects for 3D printing by novice designers. See Figure 2.20 for a taste of what these novice designers are capable of (more discussion of this occurs in later chapters, but a glimpse is far too tempting to omit here).

Chapter 3

Related Work

The belief that tangible objects[1] play an important role in children's education is relatively recent. Friedrich Froebel's use of 20 wooden forms he dubbed 'gifts' in the first Kindergarten was in 1837[52]. It took until 1907 before an extension of Froebel's ideas and a focus on physical, manipulative objects and tasks was implemented by Maria Montessori in the first Casa Dei Bambini[97]. The interest in children's learning incorporating the use of manipulatives progressed steadily, most notably by Jean Piaget and his work on 'genetic epistemology'. Piaget wrote extensively on the stages of development during which certain kinds of knowledge emerged[68], including logical-mathematical knowledge related to the kind we wish to foster. Additionally, by using our devices as an assessment vehicle for children's spatial reasoning, one can position our work as part of a tradition (dating back at least to Piaget [106]) in understanding spatial thinking and its development (cf. also [102] for a more recent treatment of the subject). While Piaget's specific theories have been strongly challenged[131][115], his influence was (and is!) extremely important. Seymour Papert, one of Piaget's intellectual descendants, published Mindstorms[104] in 1980 and with it introduced his own ideas about constructivism. Combined with the advent of the physical Logo turtle, Papert brought many constructivist ideas into the modern age and opened the door for a technical and cognitive exploration of how computation and interactive objects could be combined to examine the link between tangibles and children's learning.

[1] It is worth noting the difference in this work between 'tangible objects' of the sort that a child might play with (e.g. Lego) and 'tangible user interfaces' (TUIs) that a child might interact with - typically a peripheral device (apart from the keyboard and mouse) that communicates physical interactions to a computer.

While a rich and diverse lineage of tangible and embedded user interfaces has progressed since (and partially because of) Papert, the genealogy of the proposed work derives from an interest not only in constructivist-like activities, but in theories about how interaction with physical objects may be beneficial to learning. In cognitive science, the area of embodied cognition examines the ways in which our interactions with the physical world shape our cognitive experiences from a body-centric point of view. More specifically, embodied cognition holds that our cognitive processes are 'deeply rooted in the body's interactions with the world'[140]. This is in stark contrast to decades of research in cognitive science wherein the mind was viewed as a sort of central but detached information processing unit where motor-sensory functions were more-or-less secondary inputs and outputs to a main system[35].

Although there are several different tenets of this body-centric view, the primary conclusion relevant to our proposal is that interactions with physical objects can shape, clarify, and reinforce our cognitive processes in scores of disparate areas. Of keen interest for this work in particular is a domain referred to as embodied mathematics. Lakoff and Nuñez[82] give a fascinating account of the origins of mathematics from an embodied point of view. They propose that humans, by virtue of their interactions with the physical world, inevitably form certain intuitions of a mathematical nature. Recognizing small numbers of objects (e.g. the pre-verbal ability to do arithmetic with less than five objects), estimation, and simple comparisons are a few of the examples given in[82]. From these basics, they argue that four kinds of physical operations (object collection, object construction, using a measuring stick, and movement along a path) form the basis of simple arithmetic. Although the book postulates about concepts as ungrounded and seemingly abstract as infinity, for our work it is enough to suggest that the interactions present in our designs follow from these four operations and may in fact contribute to the solidification of more complex mathematical ideas in 3D modeling and digital fabrication (e.g. forming correct mental models of 3-dimensional objects). Such notions of embodied mathematics haveeven before the Lakoff/Nuñez textplayed a role in discussions of the development or instruction of mathematical ideas. The link between physical experience and mathematical growth was a strong element, for instance, in Montessori's work (see, e.g., [59]); much of the motivation behind traditional mathematical "manipulatives" such as number rods and balance beams can also be traced to this intellectual tradition. More recently, theoretical discussions of embodied cognition have given rise to fine-grained observations of the connections between bodily activity and mathematical learning: Goldin-Meadow[56], for instance, describes a fascinating line of research in which children's nonverbal gestures appear to both reflect and, in some cases, anticipate their verbal understanding of concepts such as conservation and "inverse operations". In other work, Ehrlich, Levine and Goldin-Meadow show that through an analysis of hand gestures, one is not only able to predict a subject's 'readiness' to learn mathematical concepts[55] but that the kinds of gestures children make (those relating to movement, for example) are correlated with spatial reasoning ability[38] and performance on mental transformation tasks.

Pedagogical research in embodied mathematics has, moreover, proceeded hand-in-hand with the development of desktop, embedded, or portable technological artifacts to support the link between bodily actions and mathematical conceptualization. Papert's discussions of the Logo computer language [104] reveal this connection early in the history of children's computing: Papert discussed, for example, the way in which the program for a Logo circle resonated with children's bodily understanding of moving in a circular path. More recently, Nemirovsky et al. [101] describe the use of a computer-based motion detector system to assist children in the development of intuitions behind graphing; Howison et al. [63] used a device based on a Nintendo Wii remote to assess children's understanding of ratio (the children attempt to move their arms in a manner illustrating a target ratio); Bakker et al. [18] created a collection of handheld objects ("MoSo Tangibles") with embedded sensors to help children learn about musical ideas via hand motions such as waving, squeezing (pressing hands together), and shaking up and down, among others; Mickelson and Ju [96] use sophisticated video and projection equipment as the basis of activities through which children can learn about mathematical ideas (e.g., symmetry, rotation angles) via large-scale physical movements.

In their section on 'Thinking Through Doing', Klemmer et al.[78] give a particularly poignant summary of why we ought to consider the body as instrumental in any human-computer interaction design, stepping through many of the concepts outlined above. In fact, the marriage of ideas derived from Papert's work with the conclusions of embodied cognition are not new, and appear to substantiate our motivations to produce tangible, manipulative interfaces as opposed to purely 2-dimensional screen-based work. In the mid-to-late 1990's, research examining the ways in which physical objects might be infused with computational ability started to coalesce around several themes[43]. Resnick's work with "digital manipulatives"[117][144] specifically references the contributions of Froebel and Montessori in the design of a series of "programmable bricks" with computational ability whose aim is to make certain specific concepts (e.g. systems-level thinking) more salient for the user. Ishii's work on breaking down the divide between physical and virtual worlds into 'tangible bits'[71][70] has subsequently set the stage for a new family of tangible interface designs that support the kind of embodied interactions that our work seeks to produce. By constructing environments and artifacts that focus on the possible physical representations of computational components, these works (among others) created the philosophical space to delve into how tangible objects might affect users at a cognitive level. Our proposal is a confluence of both tangible and cognitive design; as Resnick states, 'We are interested in Things That Think only if they also serve as Things To Think With'[117].

Having shown several PopCAD prototypes in Chapter 2 representative of a "renaissance" in papercrafting by infusing it with electronics, it is worth situating that work in relation to that of other researchers in this (still embryonic) field. The blending of traditional papercrafts with emerging technology is in fact still a relatively novel technique, but there is a remarkable community of researchers beginning to explore this area. For us, a special debt is owed to Leah Buechley's High-Low Technology group at the MIT Media Lab; that group first (to our knowledge) introduced conductive ink and copper tape into paper-based projects. Early (c. 2008) use of conductive ink with microcontrollers on a paper substrate can be found in [32] and [42] with the development of paper-based Arduino processors and simple electronic components (e.g. LEDs, toy motors, switches) that could be placed onto conductive paint to form an electronic connection. This work culminated with a paper application usually reserved for home remodeling: a "living wallpaper"[33] where passers-by could trigger light, movement, and sound by interacting with different parts of the surface (see Figure 3.1).

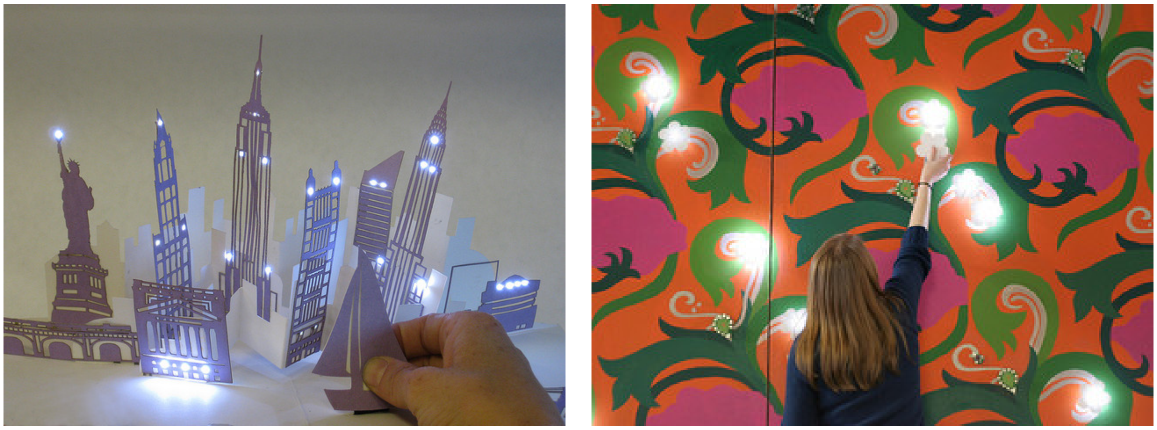


Figure 3.1: Examples of paper-based electronics: Electric Popables (left) is a pop-up book infused with a variety of paper-friendly electronics. The Living Wall (right) is a complete interactive environment embedded in wallpaper, reacting with light, sound, and movement.

These early efforts in turn spawned developments that further refined the expressive potential of paper-based electronics, infusing traditional papercrafts with new elements and abilities. An electronic pop-up book by Qi and Buechley[111] re-imagined the traditional pop-up by infusing each page with paper-friendly, interactive circuitry (e.g. by using a copper tape circuit to power LEDs in a pop-up cityscape), and from which PopCAD certainly owes some debt. Other projects in this vein include techniques to animate origami structures through shape-memory alloy (SMA)[112], using SMAs in the design and fabrication of printable paper-based devices (e.g. speakers and lamps)[121], storytelling and craft-making through electronically-enhanced storybooks and workshops [72][34][132] and the use of small microcontrollers incorporated into programmable paper-based sculptures[94].

These efforts have focused on the creation of compelling(either electronically or digitally enhanced) papercrafts. As noted in the introduction, there are numerous technological developments that, in combination, serve to accelerate the development of paper mechatronics. For instance, Kawahara et al.[77] describe how inkjet-ready conductive ink can allow circuits to be printed easily and directly onto paper; and Koizumi et al.[79] present a toolkit for wireless control of movable paper toys, Zhu et al.[142] describe a method for wireless power transfer for paper computing, and Coelho et al.[36] have achieved the direct embedding of conductive components during the papermaking process.

Of particular interest for the current work are explorations focusing on 3D modeling and perception with tangible interfaces. Prime examples include software that allows for 3D shapes to be flattened into paper-printable, origami-esque polyhedra[44], a construction kit with kinetic memory so as to record and playback certain user-generated manipulations[114], as well as several variations of "smart-cube" interfaces [135][123] that encourage spatial and logical reasoning in order to make use of the computational aspects of the cubes. While diverse in their implementation, these kits point to ways in which interface design can tease out the kind of 3-dimensional problem-solving and exploration present in the proposed work.

Figure 3.2: Left: The ActiveCube system. Right: The Roblocks system.

Related contributions focus more on the cognitive processes involved when exploring embodied interfaces with children. Research on supporting creative problem solving with children[23], arguing for a kindergarten-influenced approach to creative thinking[116], embodied approaches to analyzing children's interactions with smart objects[12], as well as the embodied design of interfaces for introducing mathematical concepts to kids[8] have shown a great degree of correlation between physical interaction and learning in children.

Yet so far, there have been few attempts to design embodied interfaces for children that

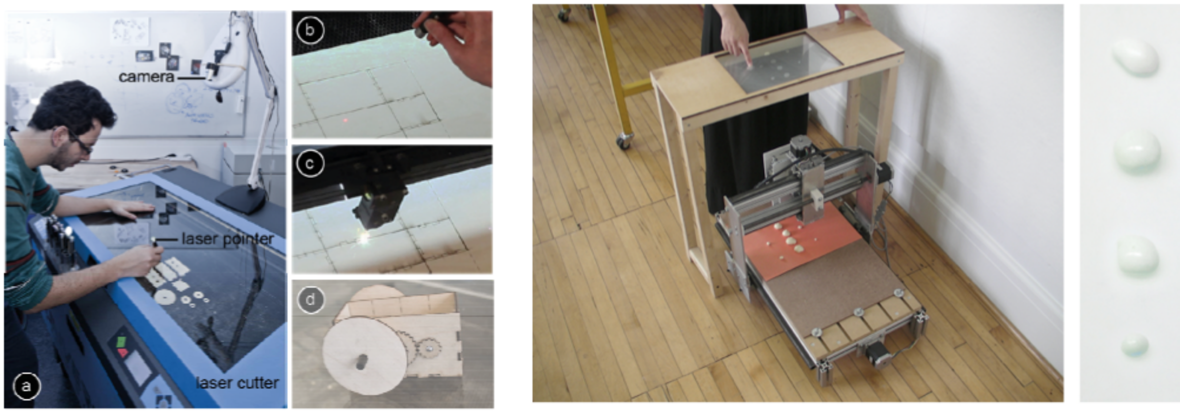


Figure 3.3: Examples of interactive fabrication interfaces: Constructable (left) allows users to control a laser cutter with a set of physical tools as opposed to a pre-defined design file. Shaper (right), and interactive fabrication tool using expanding polyurethane foam.

specifically address the growing presence and availability of digital fabrication tools. KidCAD[50], a deformable pad that captures the 2.5D geometry of depressions made on the underside of the surface, was a very promising idea in that it allowed very young children to take small objects from their surroundings (or their hands) and 'stamp' them into the pad - an intuitive and satisfying experience. Unfortunately, the authors intentions to be able to output the geometry to 3D printers has not yet manifested. Easigami[65] is a set of interchangeable and interlocking polyhedral faces with smart 'hinges' that can reproduce the morphology of a set of connected faces while connected to a computer. In contrast, Easigami **is** able to export this morphology to a stereolithography file ready for 3D printing. There are several other interfaces that deal with 'interactive fabrication'[139]; devices that manipulate materials interactively based on various input from a user, such as controlling a laser cutter with a laser pointer (instead of through a CAD program)[100], or a wearable device that takes in a CAD file and provides haptic feedback to make the physical creation of the device by hand easier, even for a non-fabricator[143]. These projects, as well as several others that deal specifically with digital fabrication for laser cutting[73][138], are examples of the subset of tangible interfaces to which this work belongs - namely, those concerned with providing a means to engage with digital fabrication technologies in a more intuitive, embodied fashion. However, with

Figure 3.4: Left: The KidCAD interface showing a model Zebra and its 2.5D impression on screen. Right: The Easigami system, showing a series of connected polygonal faces with smart-hinges and embedded electronics.

the exception of KidCAD and Easigami these designs are not made with children in mind, nor do they cover the range of possibilities for child-friendly input devices that focus on 3D-printing. Thus, we argue that there is room in this area for the work described in the thesis, as well as a lineage that suggests meaningful results may follow from continuing to explore the incorporation of tangible interfaces with embodied design.

Specifically, we see our devices as part of a larger, burgeoning "technological ecosystem" around the activity of three-dimensional printing. The introduction chapter to this work noted several prominent researchers who argue for the democratization of this technology, and for its applications to education. Indeed, exciting early work has been done in applying 3D printing to education in fields such as architecture [26], solid geometry [61], and mechanical design [87]. Our devices are specifically designed so that they can be employed by younger students younger, for instance, than the typical (undergraduate-age) architecture student - and certainly less skilled or experienced with traditional 3D modeling software. The devices were created to enable children to specify and identify three-dimensional shapes by hand motions (instead of, by contrast, using symbolic commands directed at a two-dimensional screen display). At the same time, they are not simply devices for mathematical instruction, nor even a general tool for mathematical design -

but as a suite of experiential, embodied interfaces for engaging youth in a variety of spatial design activities aimed not only at learning but at democratizing authorship for 3D printing as well.

Chapter 4

Evaluation

This section is devoted to the description and discussion of three separate user studies with the devices discussed in Chapter 2. Two studies were performed with the original UCube device (one more informal than the other), while a longer, more detailed study involved both the SnapCAD and PopCAD systems. We present the procedure, results, and basic observations of each study in this chapter, and discuss the results more thoroughly in the next chapter.

4.1 UCube Pilot

4.1.1 Procedure

Early in 2011, we conducted an initial (and informal) pilot test of the UCube with a group of 12-14 year old middle school children. Fourteen participants, consisting of five girls and nine boys, were divided into six groups (five groups of two, one group of four). Participants were asked to model a sequence of five shapes of increasing complexity using the UCube along with the companion software. The target shapes were displayed on one half of a computer screen, while the UCube software showing the live model was displayed on the other half (as in 4.1). The first shape that participants were asked to model was a straight vertical line; after this, the requested shapes were a diagonal line, a cube, a triangular prism, and finally an irregular polyhedral object. No shape required more than four towers to complete, and shapes were always presented in the same order.

Participants were instructed to place the tower on the board (but not shown how), and were told that the software model could be rotated and filled in using the keyboard and mouse, should

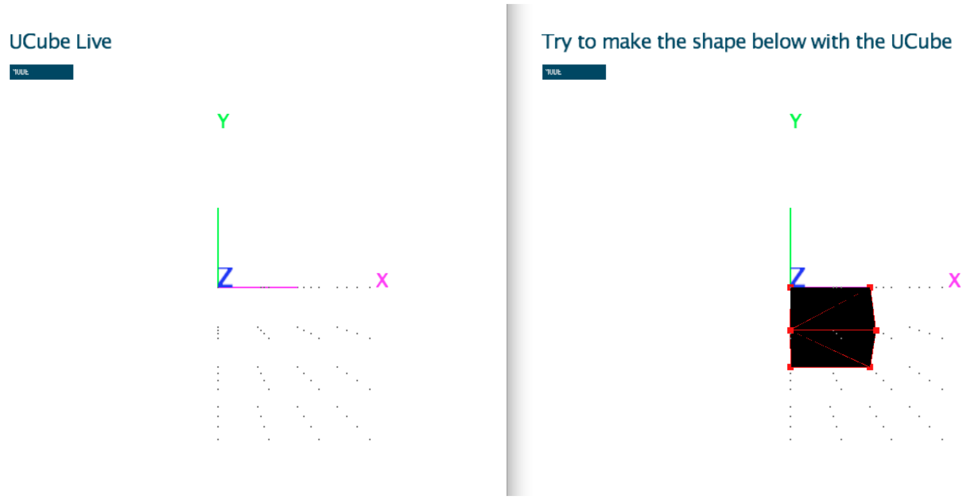


Figure 4.1: A screenshot of the testing setup, with the live output from the UCube on the right and the target shape on the left.

that help them complete the task. The participants were not given any hints as to how to complete the shapes and were not told when they had the correct configuration (they had to indicate their belief that the model was done). Participants were also instructed to "think aloud" about their actions. The main purpose of the pilot study was to get an initial impression of how the UCube would act as an accessible 3D modeling tool - how well it could help "3D novices" overcome the "2D bottleneck".

4.1.2 Results and Observations

Of the six groups who participated, four groups successfully modeled all five shapes, one group ran out of time after three shapes, and one group finished one shape, for a total of 24 of 30 possible shapes, or 80%. Sessions lasted between 17 and 30 minutes. A variety of problem-solving strategies were observed during testing, as the participants tended to treat the exercise as a sort of puzzle to be solved. Simple methods equivalent to "try and see" were common, and seemed to serve as a base point from which to draw conclusions about the relationship between the 3D model and 2D on-screen representation (e.g. "No, not there, up one"). More sophisticated strategies were also observed: "deconstructing" more complex shapes into smaller, easier-to-model shapes (e.g. thinking of one side of a cube as a square) was observed from several groups. Another popular technique was to systematically match the on-screen perspective from the live model with the shape they were attempting to model (e.g. "Okay, first let's do the top view, and then go from the side"). By orienting the two models similarly, participants were able to make more accurate modeling decisions as well as check their model against the on-screen shape. Counting distance in terms of spaces on the board, between switches, or between dots on the screen was also a very common technique of reasoning about and describing position. For example, by counting that two vertices of a shape were separated by "two dots over and one down" on the screen, subjects were able to count the distance out on the physical UCube board. A few of the more mathematically-advanced participants used terms such as "axis" and "origin" to orient themselves and describe various positions on the board to their partners. Another revealing observation in the pilot study was that, in the few instances of mechanical failure (certain switches not lighting up, towers not plugging in properly, or points not showing up on screen) the participants were still able (with a high degree of certainty) to complete the assigned tasks. This appears to indicate that, as opposed to arbitrarily moving the towers around until the two sides of the computer screen looked the same, participants had formed a more substantial mental model of the relationship between the UCube interface and the 2D representations on the screen. That opens the possibility that by performing the embodied interactions necessary to operate the UCube, participants had actually strengthened their understanding of how 3-dimensional space is typically represented on a 2D screen. Although a small, informal study on its own, this finding would strengthen the argument for using the UCube in an educational setting to improve understanding of 3D space, as well as providing a gateway for youngsters to move on to more complex modeling software. While the variety of problem-solving techniques we witnessed is a testament to the participants' ingenuity, it is also indicative of the fact that parts of the UCube are not immediately intuitive. While none of the participants had trouble understanding how to place the towers on the platform, the positions of the towers and switches had to be reasoned out explicitly. It was common for groups to clear the board of any poles when starting a new shape, even in cases where an overlap of points or tower positions existed. Although most groups completed all the shapes (or ran out of time), there were some expressions along the way of the difficulty of the task (e.g. “This is hard”, or “This is like a puzzle”). This indicates that design changes can be made in future iterations to help clarify the correspondence between positions on the UCube platform and the on-screen representation; for example, labeling the both the physical and software grid with a simple alphanumeric system. Despite these drawbacks as well as the inherent limitations of the UCube design, these early results indicate a promising ability of youngsters to effectively engage with the UCube interface. In fact, despite various levels of success in completing the assigned tasks, the vast majority of participants exhibited a high level of engagement with the UCube. For example, although the group that completed only one shape seemed unmotivated to attempt to model the other shapes, they continued to play with the interface and observe the results, even stating “this is fun” and “I like the switches”. Participants also saw potential uses for the UCube outside of the specific exercise we assigned. Comments (unsolicited) included, “you should use this to teach geometry” and “you could make this a puzzle game”. At the very least, these early results indicate that the majority of participants were able to take a 2-dimensional representation on the screen and model its 3-dimensional equivalent using the UCube, a very encouraging result in our eyes, prompting refinement of the UCube software and hardware as well as further user study, as we explain below.

4.2 Further UCube Study

Early in 2012, we conducted another user study of the UCube with a group of 11-13 year olds. The group consisted of ten participants, eight boys and two girls, from a local middle school multimedia class. Every participant was individually led through two separate exercises (outlined below) using the UCube.

4.2.1 Procedure: Modeling

Participants were handed a 3D-printed shape (modeled and printed from the UCube) and were instructed to attempt to model the shape using the UCube. The participant was initially allowed to hold the shape for approximately 10 seconds, after which they would hand the shape back to the facilitator and attempt to model the shape from memory. Participants were instructed that they may ask to hold the shape again, at which point they were allowed to hold it throughout the duration of the modeling task. Additionally, users were instructed that they had the option to skip a shape and return to it at a later point in the exercise. The five physical shapes presented were: a cube, a tetrahedron, a diamond, a "house" (a cube with a pyramid on top), and a complex irregular polyhedron. The models were presented to the user starting with the cube (as this was deemed to be the most basic shape with regard to modeling complexity). To avoid an ordering bias, we randomized the presentation sequence of the next four shapes using an online random order generator. If, after skipping a shape and returning to it, the participant was still having difficulty, we offered them the opportunity to attempt modeling the shape with the help of the UCube software, the effects of which are discussed in the results section. Participants were given a total of 25 minutes for the modeling exercise. We recorded, but did not limit the modeling time per shape, only the total time for all five shapes.

4.2.2 Procedure: Matching

Participants were instructed to face away from the UCube while the facilitator modeled a set of lights on the UCube corresponding to one shape among a set of physical models laid out on the table next to the UCube. Once the lights on the UCube were set up, the participant was instructed to turn around, and indicate which physical object they thought the set of lights on the UCube corresponded to. There were nine physical models presented on the table, and consisted of a cube, a tetrahedron, the "house" shape, a diamond, a triangular prism, an elongated hexagon, a parallelogram, a trapezoid, and an irregular polyhedron (see 4.2 for a picture of all the models). The shapes were always presented on the table in the same order and orientation to avoid discrepancies in perception or association. Of the nine shapes, the participants were asked to match five of them (the cube, the triangular prism, the parallelogram, the elongated hexagon, and the trapezoid). Thus, only the cube was presented in both the matching and modeling exercises. As with the modeling exercise, the cube was presented first, with the remaining four shapes presented in a computer-generated randomized order. Participants were given a total of ten minutes for the matching exercise, corresponding to two minutes per shape, and were instructed to think aloud during the process.

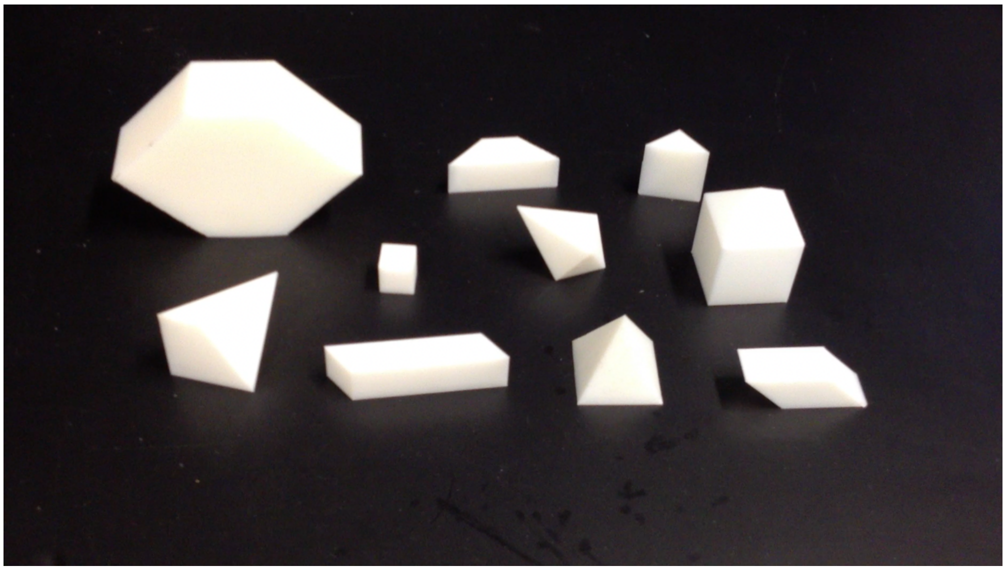


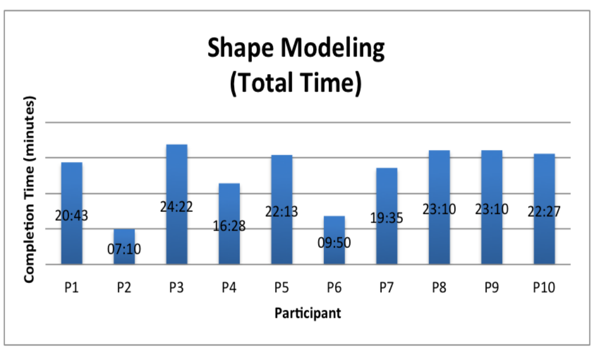
Figure 4.2: The nine models used during the user study: a diamond, trapezoid, parallelogram, cube, elongated hexagon, irregular polyhedron, triangular prism, tetrahedron, house.

4.2.3 Results

While many established forms of 3D modeling systems can be confounding and operationally too complex for a child to navigate, the UCube was positively received and system instruction was accomplished with just a minor introduction and demonstration (system instruction and demonstration lasted approximately 2-3 minutes). We found this first instance of system comprehension to offer some validation that the UCube worked well as a user-friendly 3D modeling device. This section will detail the outcome of both the modeling and matching tasks performed.

4.2.3.1 Exercise 1: Modeling

Modeling occurred under three conditions:recreate the object from memory, construction of the object while it was in the participants possession, and modeling the shape with the help of the UCube software. Overall, 21 of 50 shapes were completed from memory, 12 of 50 were completed while holding the shape, and a further 8 of 50 were completed with the aid of the UCube software, for a total of 41 out of 50 shapes modeled successfully (82%). Of the nine missed shapes, seven were of the same shape, the complex polyhedron. The remaining two misses were from the same participant, who ran out of time before completion. Of the ten participants, eight were able to recreate the cube from memory, whereas only four were able to recreate the diamond and the tetrahedron from memory. Half of the participants constructed the house from memory, and no participants were able to complete the irregular polyhedron from memory. However, once shown the software the majority of the participants found the modeling task significantly easier to perform. The irregular polyhedron was by far the hardest shape and was only able to be completed by three of the ten participants either after continued possession of the shape or using the software.



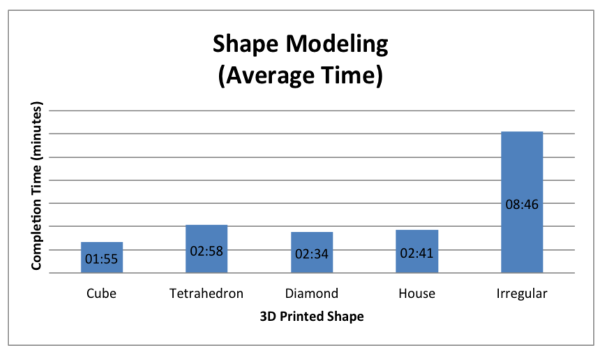
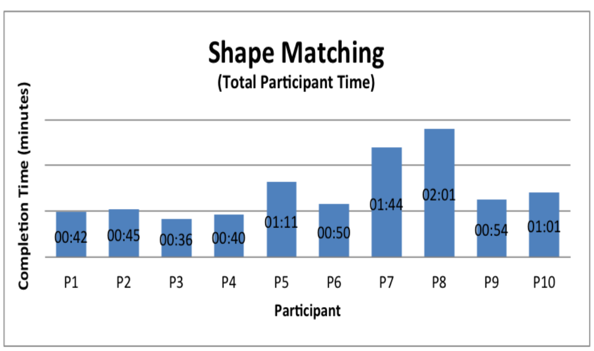


Figure 4.3: Results of the modeling task, showing total modeling time spent per participant (left) and average modeling time spent per shape across participants (right).

The graphs in Figure 4.3 represent the total completion times per participant (on the left) and average time per shape (right). Two exceptional completion times were observed, where participants finished modeling all the shapes in under 10 minutes. However, the majority of participants finished the task in the 19-25 minute range. Only one of the participants ran out of time. Once participants had been introduced to the software, 9 of 10 of participants were able to complete all but the irregular polyhedron. It is interesting to note that of the 10 participants, the child that had the most difficult time modeling, the lowest shape completion rate, and the longest completion time during the matching exercise was the youngest participant.

4.2.3.2 Exercise 2: Matching

Out of 50 matching tasks (five per participant), all but three tasks were completed in 20 seconds or less. 4.4 displays the total time spent on the matching task per participant (left) and the average completion times for each shape (right). No participant selected the wrong shape (a few preliminary “mis-selections” were made that the participants quickly corrected), and all participants completed the task in well under the allotted 10 minutes. The lack of errors in the matching task is highly encouraging as a basis from which to reason about youngsters’ abilities to perceive and reason about convex hulls as a set of lit vertices in space, meaning that this kind of 3D modeling interface might be applied to other domains (e.g., as a cognitive assessment tool, a puzzle game, etc.) with some optimism.



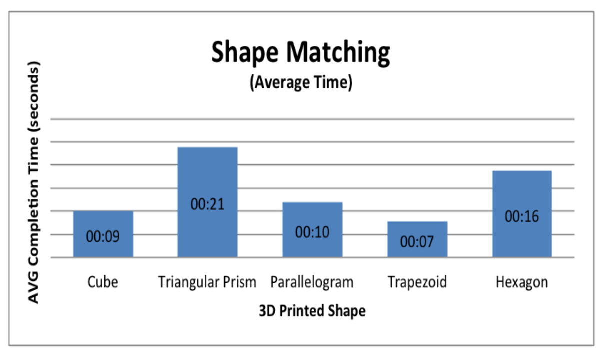


Figure 4.4: Results of the matching task, showing total time spent per participant (left) and average time spent per shape across participants (right).

4.2.4 Results and Observations

Modeling trends as well as distinct modeling behaviors were documented in the process. Common observations included building from the ground up (lowest vertices first), building in the orientation that the object had been presented in, not clearing the poles/lights from the UCube before starting to model a new shape, and modeling a shape by breaking it up into discrete parts (e.g. a participant building a house would commonly build a cube first and then add on a vertex to the top; a participant constructing the diamond might combine two opposite facing triangles.).

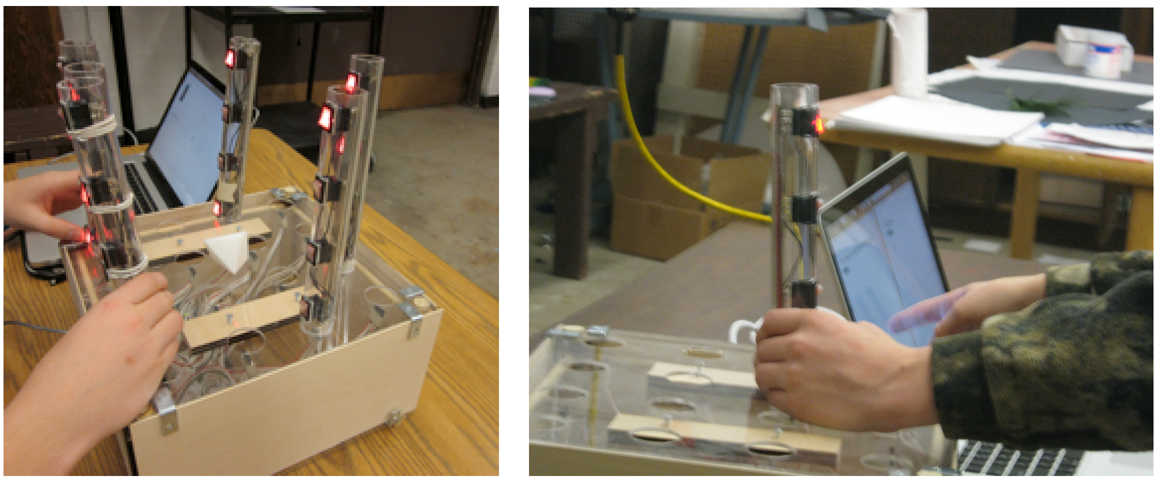


Figure 4.5: (Left: A participant modeling with the UCube, using a strategy of placing the physical model on top of the UCube while modeling, as well as using both hands simultaneously to manipulate the towers. Right: A user pointing at the software representation of the shape with one hand, while manipulating the UCube interface with the other hand.

Unique behaviors were exhibited in the modeling process as well, reflecting a type of user specific construction-based problem solving. One participant used their arm to connect the red lights of the UCube for shape definition. A few participants oriented the object differently than how it had been presentedtypically this occurred for the modeling of those objects with a pyramidal apex (tetrahedron, house, diamond). Apex formation was perhaps one of the most difficult concepts for most participants to grasp, as it required them to strategically align the base on a 3x3 grid so there was a middle plug for them to create the apex. If participants were fixated on designing from a 4x4 grid then there was no center plug for them to create a midpoint. Some participants ended up building an oblong polyhedron as opposed to a cube, or an oblique polyhedron as opposed to an equilateral tetrahedron. Other observed behaviors included a participant who modeled shapes by turning on lights for an entire shape edge, as opposed to just the corners and a participant who built shapes that were floating, as opposed to resting on the base of the UCube. There were also some notable behaviors regarding physical and gestural actions of the participants. Many participants modeled with both hands simultaneously, placing towers and flipping switches without a clear preference for a dominant hand. Participants would often gesture with their arms following an arc in parallel with a face of the object they were currently modeling. This "tracing" behavior was also noticed when participants were holding a physical model and tracing a side of the object with their fingertip, often while rotating the object with the other hand. Finally, during object possession phase three participants actually placed the 3D object on top of the UCube in the modeling space while they reasoned out the construction (see 4.5 for an example). These gestural and "embodied" interactions with the UCube, combined with a high degree of modeling success spurred us not only to create a more robust and expressive system (called - SnapCAD - as detailed in Chapter 2), but to attempt to tease out the relationships between modeling on these kinds of devices and the gestures and speech produced when subjects were explaining their strategy in using the devices. This eventually led to a comparative study using two new devices, two new modeling modes, and introducing metrics to analyze some of the "embodied" aspects hinted at above.

4.3 SnapCAD and PopCAD

Starting in early 2014 we conducted a study using both the SnapCAD and PopCAD devices with a group of 11-18 year olds at a local drop-in enrichment program that focuses on children from under-served and low socioeconomic communities. Twenty participants enrolled in the study, consisting of 12 boys and 8 girls (no one responded with other, although it was an option). We collected some basic demographic information, including age, race, grade level, 3D modeling experience, 3D printing experience, computer ownership and use, interest in engineering, and how difficult they thought classes in school were. Parental consent was obtained (and child assent given) for each subject in the study.

To present a snapshot of the demographic findings, then: the participants were primarily of Latino or Hispanic descent, but also included those of African-American, American-Indian, Asian, and Caucasian descent. Grade levels ranged from 6th-12th, with an overall average of 7.9 (8.33 for boys, 7.75 for the girls). Average age was 14 years, 1 month, 20 days (14 years, 6 months for boys, 13 years, 7 months for girls). 16 of 20 participants had a computer at home. Describing their comfort level using a computer on a scale from 1 to 10 (10 being most comfortable), the participants averaged 7.9 (8 for boys, 7.75 for girls), with no scores below a 5. Of the participants who had a computer at home (all but two of the subjects), two reported using it only a few times a year, five used it a few times per month, four used it a few times per week, and five reported using the computer everyday. Only three of the participants had any experience with 3D modeling software. Interestingly, only two of the participants had never heard of 3D printing before enrolling in the study, but none of them had ever designed or printed anything using a 3D printer. When asked about their interest in engineering, only seven children (all boys) stated they were definitely interested. However, only two kids (both girls) stated that they were definitely not. The rest (11 kids) stated that they were either "maybe" interested, or "not sure". When asked how difficult they felt school classes were, six responded "easy for me", 10 said 'somewhat easy for me', and four responded "somewhat hard for me" (no one responded "hard for me").

4.3.1 Procedure

The study ran for seven weeks total, comprising several stages, the first being a pre-assessment of spatial reasoning skills. The spatial reasoning assessment was done using the "Children's Mental Transformation Task" developed by Susan Levine ([38] p.1260-1261). In the task, participants are shown two pieces of paper, side-by-side. One piece shows a 2D geometric shape, split apart and rotated in one of several different ways. All shapes were symmetrical either horizontally or vertically (or both), and thus split along either a vertical or horizontal line of symmetry. Shapes were translated in one of four different ways: (a) translated perpendicular to the line of symmetry (direct translation), (b) translated and then moved diagonally apart (diagonal translation), (c) rotated 45 degrees outward from the line of symmetry (direct rotation), or (d) rotated and then moved diagonally apart (diagonal rotation). The other piece of paper contained the geometric shape, recombined correctly, along with three incorrect choices. In the study we conducted, participants were given two sets of 10 shapes, one set as a pre-assessment before doing any modeling, and another (completely different) set of 10 after completing the entire study, as a post-assessment. Figure 4.6 shows an example instrument, with the four possible translations.

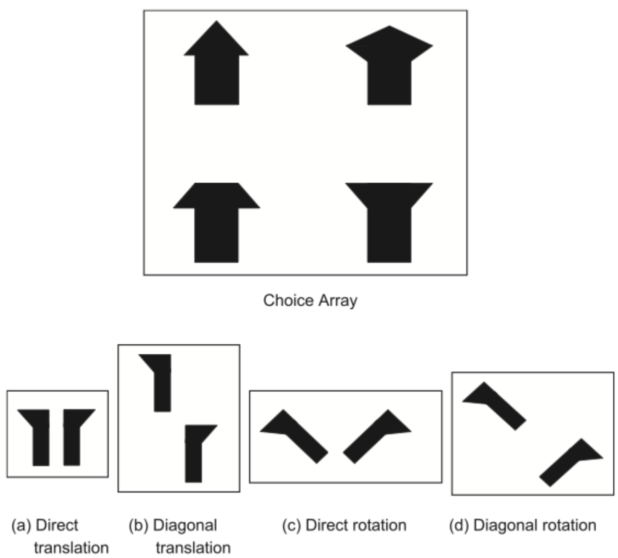


Figure 4.6: An example problem from the spatial reasoning exercise. The figure at the top shows the choice array of four shapes, where the lower left figure is the correct option. Examples (a) through (d) show the four different types of translations found in the exercises - direct translation, diagonal translation, direct rotation, and diagonal rotation.

After the pre-assessment, participants were split into two groups of 10 students each - the selection alternated evenly based solely on order of participation - with group A modeling first on the PopCAD and group B modeling first on the SnapCAD (as described in Chapter 2). Each session begins with a brief ($\approx$ one minute) introduction to the device, during which the participant is told how to operate the device, but not what any of the software buttons do, and given free time to become comfortable with the interface. Participants were encouraged to explore both the interface, and the buttons in the software that control the three primary modeling modes (convex hull, path, minimal spanning tree).

Once the subject indicates that they are ready to move on, we move into a series of three modeling exercises that explore each of the aforementioned modes. The basic operation and a brief explanation of each mode were given to the participants as an introduction to each mode. Four 3D-printed models representative of each mode were presented to the user in an order judged to be from least complex to most complex (and thus was the same for each user), for a total of 12 modeling tasks across the three modes. 24 models were used - one set of 12 was used across every user's first session (independent of device), with a remaining 12 models used in every user's second session. Figure 4.7 shows the two sets of models side-by-side.

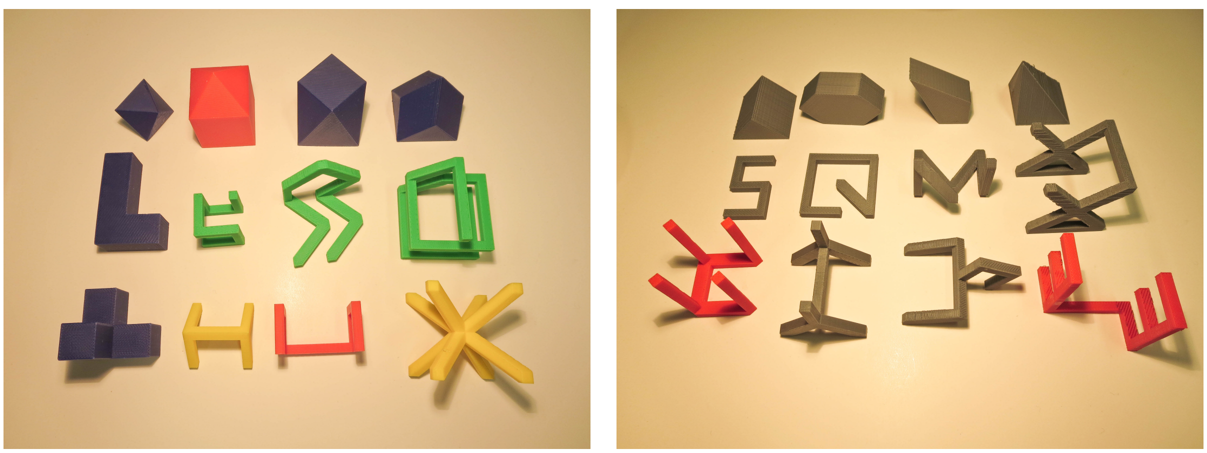


Figure 4.7: The two groups of 12 3D printed models used in the first session (left) and second session (right). Each row is a different modeling mode (back = convex hull, middle = path, and front = minimal spanning tree). The shapes were presented in order from left to right as pictured above.

The tasks that follow are the same for each device:

Tasks 1-3: Convex Hull Modeling, Path Modeling, Minimal Spanning Tree Modeling

Before each set of modeling tasks, the participant will be given a brief demo of how each modeling mode interprets the points from the device. The user will then be presented with a series of four (4) plastic, 3D-printed models that were modeled on the device using the current modeling mode. For each of these shapes, the participant will attempt to recreate the shape using the modeling abilities of the device and the software. The user will be instructed to indicate when they believe they have successfully recreated the shape, as well as to think aloud about their modeling process. The time to completion (of lack thereof), completion code, observational notes, and video shall be recorded. If the user indicates success, they shall be asked to explain their modeling strategy for the purpose of logging gesture and speech data.

Task 4: Freehand Modeling

After the modeling tasks are complete, participants are invited to "freestyle" model an object of their choosing, using any of the three modeling modes. By asking participants to think aloud about their intentions and thinking processes during this exercise, we hope that a deeper understanding may be gained of the strengths and weaknesses of the system, as well as the thought processes and engagement of the users in attempting to model a specific model of their own choosing. These saved models are analyzed, based on which mode was used to create them, complexity (based on number of points used), and whether the shape was 'exploratory' or 'intentional' (i.e., was the end artifact a result of sort of happy accident, or the result of intentional process to create a specific model).

For the first three modeling tasks (but not the freestyle modeling), time to completion (or request to move on) is recorded, along with an outcome code. The outcome is coded according to a set of conditions detailed below in table 4.1, and was developed upon analysis of the recorded video, in an attempt to fit the sorts of repeated behaviors that were in fact observed.

Participants were asked to "think aloud" about their process, difficulties, modeling choices, etc. In the case that the user believed they had correctly modeled the shape (cases C and E2 in table 4.1) they were asked to explain their modeling strategy[1] . Their explanation was videotaped and analyzed based on the coding strategies laid out in "The Importance of Gesture in Children's Spatial

[1] Cases E1,E3,E4, and I did not provide the grounds from which to ask about modeling strategy and so were not recorded.

Table 4.1: The coding used in analyzing the modeling exercise outcomes, based on observations from video taken during the study.

Category	*Code*	*Definition*
Correct	C	A complete and correct modeling of the shape
Error in recognition	E1	The correct shape was modeled, but the user did not identify it
Error in belief	E2	A belief that the modeled shape has been modeled correctly, when it has not
Error in implementation	E3	User knew shape was incorrect, and gave a correct explanation
Error in strategy	E4	Knew shape was incorrect, and did not know why or gave an incorrect explanation as to why
Error in proportion	EP	The general shape is correct, but the proportions in one or more dimensions is off (e.g. too tall, not wide enough, etc.)
Incomplete	I	Participant ran out of time, gave up, or asked to move on

Reasoning"([38], p.1264), laid out in table 4.2 below. The rationale for performing this analysis in based in part on work by Ehrlich, Levine, and Goldin-Meadow [38][85][56], which suggests that the frequency of gesture and relationships between speech and gesture act as a window into the learning state and performance of the subjects.

The second session is similar to the first, with the subject using the device not used in session one, and with 12 new models. Once modeling on the second device is completed, users will take a second spatial reasoning assessment of an additional ten questions to help gauge if any meaningful difference in spatial reasoning skills has occurred throughout the study.

A slightly modified version of the software was used for the user study, eliminating several of the functions not being evaluated for the sake of presenting a clear interface for the users. The multiple hull modes, spline, load, and save functions (described in Chapter 2) were eliminated, and the rest of the graphical user interface was reorganized and streamlined. We combined the three different .stl export buttons into a single export button that handled all three modes, changed the order of the remaining buttons and made them larger, and made the X,Y, and Z axis markings larger.

Table 4.2: The various coding strategies used in the video analysis of subjects' modeling strategy explanations. Borrowed and adapted from [38].

Category	*Definition*	*SpeechExamples*	*GestureExamples*
Movement	Any indication of movement	"Just slide them together and then it looks like that"	Miming movement with the hands
Perceptual Features	Focus on a particular feature of the model	"Because there is a little bend in here and a point thing here"	Pointing to a specific feature on the model
Perceptual Whole	Any indication of seeing the model as a whole	"It looks like an arrow!"	Gesture indicating inclusion of the whole shape
Vague	An expression of strategy that the coder cannot decipher	"Because I looked at that and I looked at the differences"	Waving gestures above the computer device that do not indicate any specific strategy
Other	Any strategy not listed above	"And here is like half of it. But so and two halves make a whole"	Using the hand to form a straight line through the middle of the whole shape to represent the line of symmetry

4.3.2 Results

This section reports on the results from our study, relaying our findings across both sessions, genders, modeling modes, and spatial reasoning scores in an attempt to tease out what conclusions, if any, we might make about the strengths and weaknesses of our devices as well as how interacting with our devices affected user's spatial reasoning abilities, 3D modeling skills, or congruence between speech and gesture in explaining the cognitive learning state of the user.

4.3.2.1 Modeling Results

In this section we will focus on delivering the results from the modeling exercises - users went through two sessions, modeling 12 shapes each time (4 shapes each using convex hull, path, and minimal spanning tree modes) for a total of 24 exercises. For each modeling task, a result code was recorded per the rubric shown in table 4.1. One user dropped out of the study (user 6) before completing round one, leaving us to report on 19 users for the first modeling session, 10 of whom started on the PopCAD and 9 of whom started with the SnapCAD. A further three users did not complete session 2, leaving 16 users, 7 girls and 9 boys, who were split evenly over the two devices (four each on PopCAD and SnapCAD).

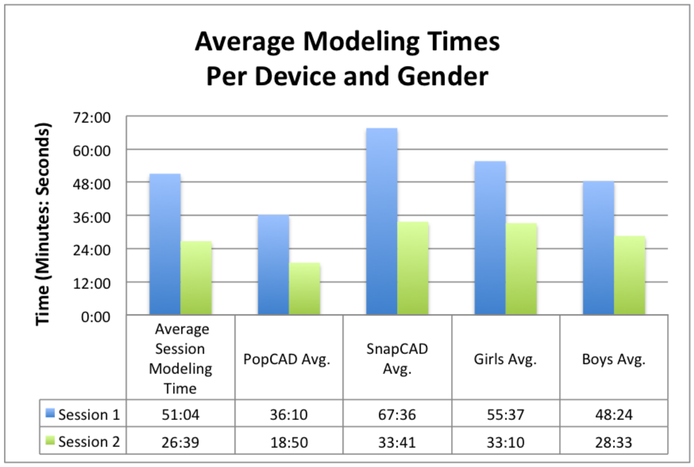
Table 4.3: An overview of the modeling task results, broken down into session number, gender, device, and modeling mode.

	Session 1	%	*Session* 2	%	*Total*	%
OverallCorrect	127/228	55.7%	116/192	60.4%	243/420	57.9%
Girls	45/84	53.6%	55/84	65.5%	100/168	59.5%
Boys	82/144	57.6%	61/108	56.5%	143/252	56.7%
PopCAD	90/120	75%	62/96	64.6%	152/216	70.4%
SnapCAD	37/108	34.3%	54/96	56.3%	91/204	44.6%
Convex Hull	40/76	52.6%	38/64	59.3%	78/140	55.7%
Path	48/76	63.2%	44/64	68.8%	92/140	65.7%
Tree	39/76	51.3%	34/64	53.1%	73/140	52.1%

Out of the 228 modeling tasks in session one, the group successfully modeled 127, or roughly 56%. Those users who started with SnapCAD performed 37 of 108 tasks, or 34%, while those using the PopCAD device completed 90 of 120 tasks correctly, for a success rate of 75%. Girls completed 45 of 84 tasks (54%), while boys correctly completed 82 of 144 tasks (58%). Individual scores ranged from 0 to 12 (perfect), with an overall overage of 6.68 correct shapes per user. Average correct shapes per user was 4.11 for SnapCAD and 9.00 for PopCAD.

In session two, 116 of 192 (60%) tasks were performed correctly, with SnapCAD modelers correctly representing 54 of 96 shapes (56%) and PopCAD modelers completing 62 of 96 shapes, or roughly 65%. Girls completed 55 of 84 tasks (65%) while boys completed 61 of 108 tasks for 56%. Individual scores ranged from 3 to 12 (perfect), with an average of 7.25 correct shapes overall, while the average correct shapes per user was 6.75 for SnapCAD and 7.75 for PopCAD.

The two bar graphs in 4.8 show the average modeling times broken out over device and gender (on the top) and modeling mode (on the bottom). Modeling times were recorded from the time the user was handed the shape until they indicated either that (a) they believed the model to be complete, or (b) they gave up, wished to move on, or thought they were as close as they were going to get (though they knew their model to be incorrect).



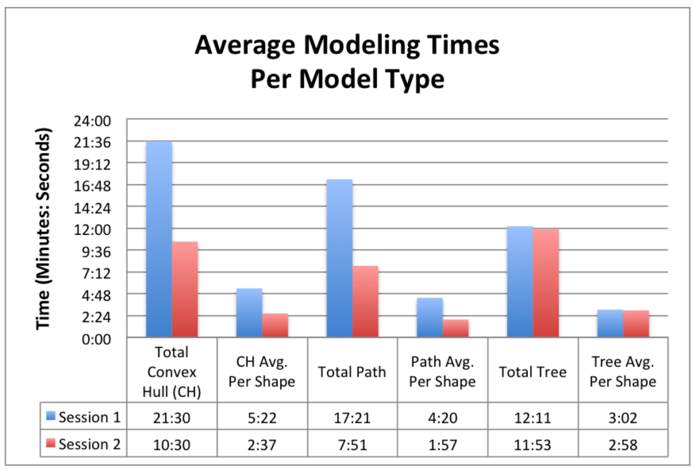


Figure 4.8: The average recorded modeling times for each session, broken out (on top) by device and gender, and (on the bottom) by modeling mode.

We can easily pick out a few trends from these two graphs: average modeling session time went down significantly in the second session, regardless of device or gender, although boys took less time in both sessions, and the PopCAD seemed to take less time overall in each session than modeling on the SnapCAD (although interestingly, the SnapCAD modelers in the second round improved on their times from modeling on the PopCAD in the first round). When examining mode types, we see a similar trend of significantly decreasing modeling times in the convex hull and path modes, but curiously, not in the tree mode where times improved in the second session by only a few seconds. While the minimal spanning tree mode took subjects the least amount of time (of the three modes) in session one, the improvement in both convex hull and path modeling times left the spanning tree with slowest overall and average modeling times in session two. Seeing as the minimal spanning tree mode posted the lowest percentage of correct shapes in both rounds (and thus overall), we might expect the ranking we observed in round two, where average modeling times corresponded with the overall percentage of correct shapes. It seems plausible that mastery of the tree mode is slower to arrive than either the convex hull or path modes, and therefore one extra session produced more dramatic results in the other modes (convex hull and path modeling both improved by almost 7% in session two, minimal spanning tree by less than 2%).

4.3.2.2 Mental Transformation Task Results

Subjects were given two sets of 10 mental transformation problems, as discussed previously in the procedure section. The first set was given before the first modeling session, as a sort of pre-assessment. The second set was given after the second modeling session as a post-test. We recorded performance data by session and by user, and present the results in Figure 4.9 broken out by the type of symmetry represented in the shape (unilateral or bilateral) and the type of translation or rotation performed on the shape (direct or diagonal translation, direction or diagonal rotation), meaning that each shape had both a symmetry type and a translation type.

Overall, subjects performed very well on the Mental Transformation Task, correctly responding to 614 of 720 questions (a little over 85%). Performance was remarkably equal across genders, with girls correct on 256 of 300 (85.3%) and boys on 358 of 420 (85.2%). Accordingly, we found no sigificant difference in gendered responses across any symmetry or translation type - girls and boys succeeded and struggled on the same sorts of tasks. Bilateral symmetry was significantly easier than unilateral, with over 90% of bilateral tasks and only 78% of unilateral tasks performed correctly. Rotation was more difficult than translation, and diagonal transformations were more problematic than direct ones. Hence, diagonal rotations scored the lowest (75%), followed by direct rotations (82%), diagonal translations (91%), and direct translations (93%).

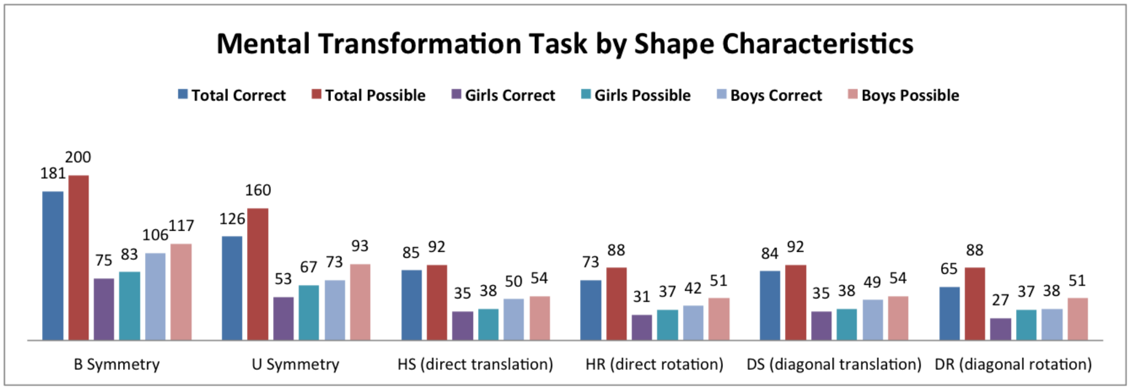


Figure 4.9: A view of the Mental Transformation Task results, broken out by symmetry type (B = bilateral, U = unilateral) and rotation or translation type performed on the shape being transformed.

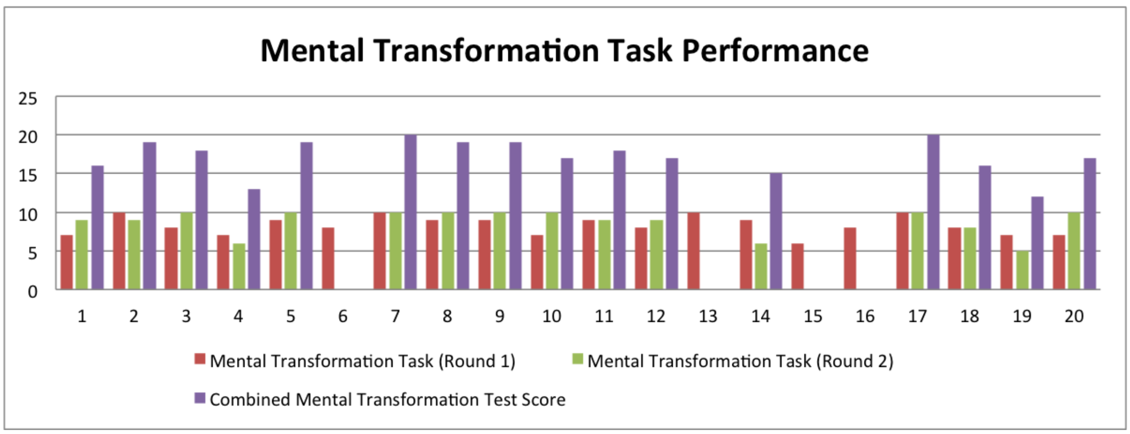


Figure 4.10: Mental Transformation Task results, broken down by session and by user.

Figure 4.10 shows the Mental Transformation Task results broken down into sessions by user. We observed a +7 net improvement in the second round among the 16 users who participated in both sessions. Both girls and boys improved in the second session, though girls improved by a greater percentage when compared to boys - from 82.3% to 88.6% while boys improved from 83.3% to 87.7%, a 2% greater improvement among girls. Four users did worse on the second set of tasks, four did the same, and 8 improved; the greatest change in both directions was +/- 3. There was a weak correlation between improvement between sessions (or lack thereof) and modeling performance overall ($r = .20, p < .5$), but a weak negative correlation between improvement on the Mental Transformation Task and improvement in modeling score from session 1 to session 2 ($r = -.20, p < .5$), suggesting that the **change** between sessions on the spatial reasoning test and modeling performance are mildly related, if at all.

4.3.2.3 Speech and Gesture Coding Results

During the modeling exercises, if a subject believed (whether correctly or not) that they had successfully modeled a shape, the facilitator asked the subject to describe the modeling strategy they used to arrive at their answer. During these explanations, video recordings were analyzed for five types of speech and gesture behaviors: those referring to movement, to the perceptual whole of the shape being modeled, to a perceptual feature of the shape being model, as well as behaviors that were vague or unintelligible, and those that did not fit into any of the above categories (labeled as "other" - a more detailed description is available in the procedure section above). A given strategy was only recorded once per modeling task, but multiple strategies per explanation occurred often and were recorded (as was also the case in [38]). The tables below break down the numbers and types of speech and gestures observed over the two sessions; as such, we only report on the 16 subjects who completed both sessions.

Table 4.4 shows the total number of gesture and speech types we recorded, as well as how they were split between each devices, genders, and sessions. The most common gesture and speech types (by a significant margin) were about specific perceptual features of the models, those relating to movement came next, followed closely by vague gestures and speech. The other two categories, perceptual whole and "other" strategies, were barely represented in gesture - they were far more common in speech, but still ranked as the least frequently recorded. Many users explained their modeling strategy by doing a "step-by-step" recounting of their process that referred at each step to the part of the shape they were modeling at that point. For example, it was common for a subject to point to a segment of the model and say (for instance), "and then I put a point here, for this part…", generating perceptual feature scores in both gesture and speech for nearly every explanation they gave. Movement was often explained along the same lines (though less frequently),

Table 4.4: Gesture and Speech Observations over both sessions. Numbers in this table exclude the totals from the three subjects who finished the first session but not the second.
G = Gesture, S = Speech, .M = Movement, .PW = Perceptual Whole, .PF = Perceptual Feature, .V = Vague, .O = Other.

	Total		*PopCAD*	*SnapCAD*		*Girls*	*Boys*		*Session* 1	*Session* 2
G.M	113		62	51		73	40		39	74
G.PW	13		8	5		8	5		9	4
G.PF	180		102	78		96	84		93	87
G.V	100		50	50		42	58		34	66
G.O	7		4	3		6	1		4	3
S.M	107		64	43		55	52		46	61
S.PW	68		39	29		35	33		38	30
S.PF	186		103	83		97	89		101	85
S.V	104		55	49		40	64		32	72
S.O	70		35	35		33	37		18	52
Gesture	413		226	187		225	188		179	234
Speech	535		296	239		260	275		235	300
Combined	948		522	426		485	463		414	534

often with subject using specific words that indicate motion (e.g. "then I move over here", "I had to go up here, then follow the path back down again") while simultaneously motioning along the directions they were indicating.

Interestingly, even without accounting for the difference in number of subjects, girls "out-gestured" the boys overall (225 to 188), and in every category **except** for vague gestures, where boys were vague in describing their strategies 24 more times over the course of the study. Speech types were more gender-balanced, with the final tally being 260 for girls and 275 for boys, however seeing as boys had more participants in both sessions of the study, the speech-per-participant count actually favors the girls as well. The PopCAD interface produced more gestures (226 to 187) and speech (296 to 239) than the SnapCAD, a finding mitigated somewhat by the fact that users modeled so poorly on the SnapCAD in the first round and therefore did not arrive at a point where a modeling strategy could be explained. If we isolate the second round only, where the performance breakdown was much more even (62 to 54 in favor of PopCAD), then SnapCAD actually produced more gestures (124 to 110) and more speech elements (159 to 141).

Perhaps the most curious data from Table 4.4 is the big increase in both gesture and speech from round 1 to round 2 of the study. Even with three less participants in round 2, overall instances of gestures increased from round 1 by 55 (179 to 234, a 76% increase), and speech instances increased by 65 (235 to 300, a 78% increase), yet the overall modeling performance only increased by 5% in round 2. A bit of a closer look at the types of gesture and speech gives a plausible explanation: in both gesture and speech, the number of **vague** indications rose dramatically (+32 for gesture, +40 for speech), while the number of perceptual feature indications dropped in both cases (-6 for gesture, -16 for speech). If we look at Figures 4.11 and 4.12 these numbers start to make more sense.

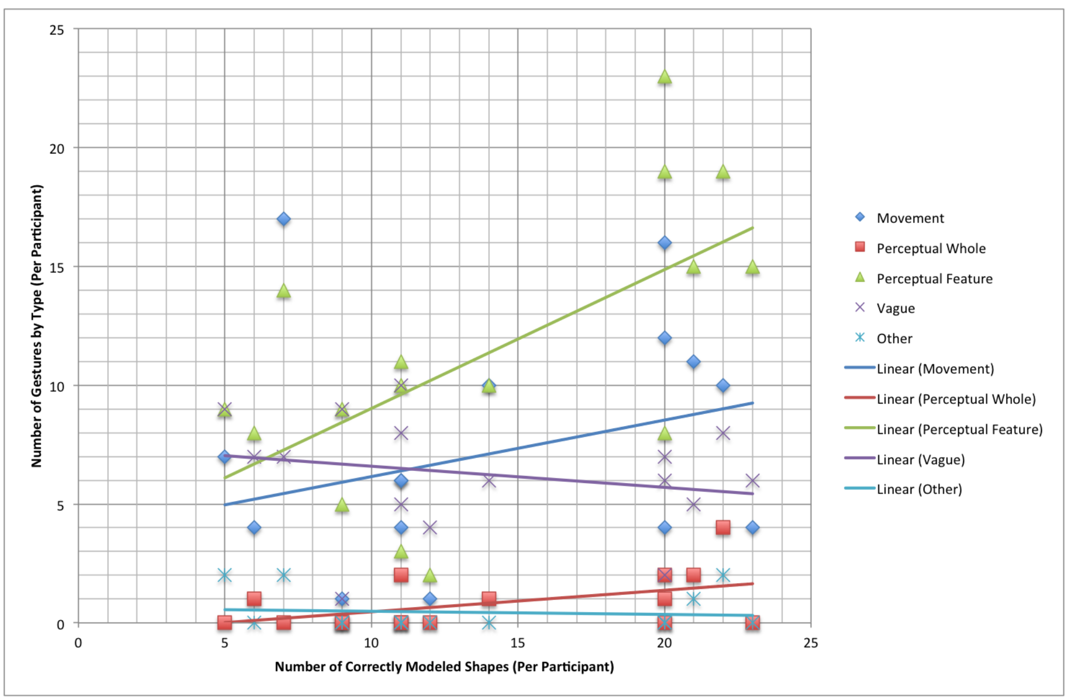


Figure 4.11: A plot of the five types of gestures we coded (movement, perceptual whole, perceptual feature, vague, and other) over the number of correctly modeled shapes. The slope of the lines indicate the strength of correlation between each gesture type and overall modeling performance.

Figure 4.11 shows a plot of the number and kind of gestures produced by a user over the number of shapes they modeled correctly over the two rounds of the study.[2] The lines associated with each scatter plot shows the strength of the correlation between instances of that gesture type and modeling performance; the steeper the positive slope, the higher the positive correlation and vice versa. As we can see from the graph, three of the conditions have positive slopes (perceptual feature, movement, and perceptual whole), while two have negative slopes (vague and other). By far the strongest positive correlation[3] is between perceptual feature gesturing and modeling performance ($r = .61, p < .025$), while vague gesturing has a weak negative correlation ($r = -.22, p < .5$). Going back to our earlier table, then, the sharp uptick in vague gestures and mild decline of perceptual features may help to explain why such an increase in gesturing did not result in a similar upswing in modeling performance.

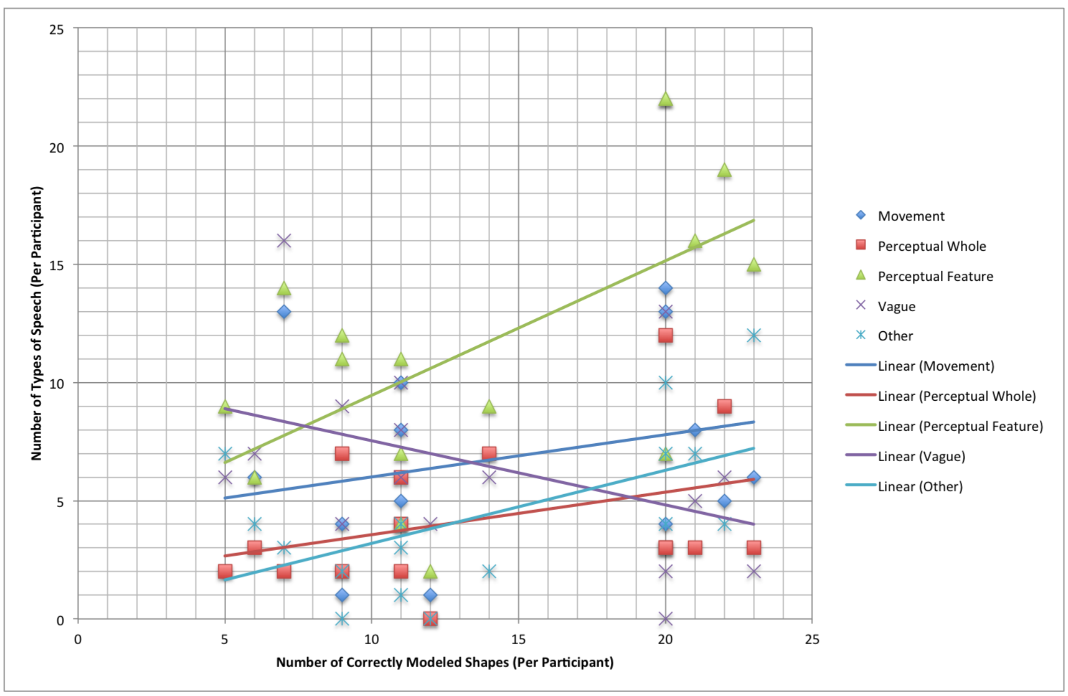


Figure 4.12: A plot of the five types of speech we coded (movement, perceptual whole, perceptual feature, vague, and other) over the number of correctly modeled shapes. The slope of the lines indicate the strength of correlation between each speech type and overall modeling performance.

One might expect that correlation patterns would be similar between gestures and speech

[2] Data from the three users who dropped out of the study has been omitted from this graph as well as Figure 4.12
[3] All correlation calculations were done using Pearson's Correlation Coeffcient.

of the same type (e.g. instances of movement in gesture would be as correlated to modeling performance as instances of movement in speech), and while we did find some similarities, some surprising differences appeared as well. Speaking about perceptual features was (as with gesturing) the most highly correlated type to modeling success ($r = .58, p < .025$), but where gestures marked as "other" had a very weak negative correlation, "other" categories of speech were second most **highly** correlated with modeling aptitude ($r = .56, p < .025$) - nearly as much as utterances on perceptual features. Part of this explanation lies in the frequency discrepancy between "other" gestures, of which there were only seven, and "other" speech utterances, of which there were ten times more (70). The other (pardon the pun) part of the explanation lies in the fact that we have many more words with specific meanings than gestures that are precisely defined, so (for example) explanations referring to looking at the software itself (e.g., "I looked at the screen and it looked like it."), or reasoning about the nature of how the mode works (e.g., "Since it was path I knew it would work."), or internal operations (e.g., "I just look at it and see it"), are harder to perceive in gesture. A possible relation to the potential for specificity in speech lies in the stronger observed negative correlation between modeling ability and speech marked as vague ($r = -.41, p < .25$), compared to gestures marked as vague ($r = -.22, p < .5$), indicating (perhaps) that a failure to speak specifically (given more abundant options) is more harmful than a similar failure when gesturing.

4.3.3 Observations

A few notes on the above findings are worth making here. Broadly speaking, the study indicated many positive outcomes: overall modeling ability went up while average modeling time went down, the participants improved on every modeling mode in the second session, there was a net positive performance on the second mental transformation task when compared with the first, and participants were generally engaged by the experience, which for most subjects was their first computer-based 3D modeling experience. However, even though no user saw the same device or shape twice, it is as yet unclear how much of the improvement might be contributed to a "practice effect". Due to the "drop-in" nature of the user study environment, the time between each single participants' sessions varied, based on their attendance and availability (i.e., in some cases users had homework or other activities to finish).

We observed some moderate correlations between types of speech and gesture and modeling success, though not necessarily the kinds of correlations we might have expected based on prior related studies. Nor were speech and gesture correlated to modeling acumen in the same ways - some types of gesture were less effective than their corresponding spoken elements, and vice versa. In some cases, the results were observed were counter-intuitive - such as the anomaly in average modeling times of subjects when using the minimal spanning tree mode, the fact that boys performed worse during the second modeling session while girls performed much better, and that some subjects performed worse on the second mental transformation task, even though they were arguably "primed" by going through the modeling exercises beforehand. Also unexpected is the sharp decline in performance on the PopCAD device in the second session - over 10% - especially after such a high percentage in the first round and given more "experienced" users in the second session. Equally surprising, given a rather unimpressive first round performance, is the sharp increase in modeling success on the SnapCAD in the second round (a jump of 12%), so much so that when coupled with the decline in PopCAD performance, we may wonder on the possible disparity between the groups in "inherent" ability for these kinds of tasks. Another possibility is of course that the order in which subject encouter the devices is more important than we had originally surmised - perhaps the users who started with PopCAD did better on the SnapCAD (and overall) **because** they started with PopCAD. We examine these, as well as the relevance of age and shape complexity on modeling ability, along with a deeper discussion of results across all three studies in the following chapter.

Chapter 5

Discussion

We devote this chapter to a deeper look at some of factors that may (or may not) shine a light on a few of the more perplexing and intriguing observations made through the user studies mentioned in the previous chapter, as well as a meta-review of what, if anything, we may conclude based on our reported data and observations. We spend the most attention on our latest study (with the PopCAD and SnapCAD) as it is not only the most recent, but the most significant, as it concerns not only whether or not our devices can be used effectively for modeling by novice users, but touches on the relationships between our devices, spatial reasoning, and embodied cognition. We start with the significance of the gesture and speech observations made in the PopCAD and SnapCAD study, discuss the role of age in relation to modeling performance, followed by an analysis of the effects of shape complexity on modeling acuity, and finally some meta-analysis of the observed data over the three user studies described in the last chapter.

5.1 Gesture and Speech Significance

The work of Ehrlich[38], Levine[85], and Goldin-Meadow especially[55][56], serves as a rough guide to our most recent study design, as they touch on the role of gesture in determining spatial reasoning performance, and later provide strong evidence that gesture is a valuable window into the mind, all of which supports the notions inherent in embodied cognition - that body and mind are far more tightly linked than we have traditionally be led to believe. As we operate under these assumptions as reasoning to create tangible, physically involved interfaces (as opposed to pure 2D software) it is worth taking a deeper look into how our study results compare and contrast with this earlier work.

Ehrlich and Levine's studies focus on the gestures and speech produced during children's explanations of how they solved a series of mental transformation tasks. The participants were presented with the same sorts of instruments used in our PopCAD and SnapCAD study, although our studies differ significantly in several ways. Of course, Ehrlich and Levine had no devices, and evaluated the speech and gestures produced in explaining the mental transformation task, whereas we examined the strategies expressed when modeling on the PopCAD and SnapCAD. Apart from one practice condition where wooden blocks were used (which in their study had no effect) all of the tasks in [38] were based on 2D paper representations, and the subject had no physical contact with any of the objects they were trying to model. In our study, subjects were handed a 3D-printed (and thus 3D) model of the shape they were attempting to reproduce. Additionally, in Ehrlich and Levine's studies, subjects were instructed to (in their mind) "move the pieces together" or to "observe the movement" of the pieces as manipulated by the experimenter. These factors, as well as differences in age (our subject population was 5-13 years older than those in [85] and [38]) likely contribute to (at least of some of) the differences observed in our study. We spend the next two subsections dissecting the similarities and contrasts in our results compared to the finding reported by Ehrlich, Goldin-Meadow, and Levine.

5.1.1 Contrasts

Given the differences in the nature of our study compared with Ehrlich, that our finding should differ should come as no surprise. While both of the gesture and speech analyses occurred while subjects were explaining a modeling strategy, the type of modeling activity they had been asked to perform was substantively dissimilar. As noted above, subjects in our study were handed physical, 3D models of the target shape they were tasked with modeling, and could hold on to that object (and rotate it, look at it from different angles, hold it in front of the device or the computer screen, etc.) during their modeling process. Afterward, when asked about their strategy, they still had that object, and often gestured to it (or with it) and (of course) talked about it. No such "hands-on" activity was involved in the studies we reference above, nor (as we note later in the chapter) could we find substantive work involving such manipulative activities to examine spatial reasoning ability.

We contend that the "embodied" nature of the tasks in our study help explain some of the differences in gesture and speech patterns and correlations that we observed. In fact, it is likely that given Goldin-Meadow's body of work and studies involving cognition and gesture, she would concur with us. Furthermore, the way in which the examiner in the above studies introduces the tasks to the subjects involves several direct references to movement (e.g. "In your mind, move the pieces together and then move them back apart."). This is significant, as gestures and speech relating to movement was (in their study) both the most frequent type of strategy expressed, but, as far as gesture correlations with task performance is concerned, gestures coded as relating to movement was the only type of response they recorded that was exclusively related to answering the test questions correctly. To take an excerpt from [38] (pp. 1265):

> "Gesturing about moving the pieces was correlated with the number of problems answered correctly ($r = .461, p < .001$), but it was not correlated with the number of problems answered incorrectly ($r = .202, ns$). Thus, gesturing about moving the pieces together was uniquely related to correct performance, whereas talking about moving the pieces was not."

To summarize our related findings, then: in our study, gestures about movement were the second most observed expression, after those related to perceptual features (113 to 180); speech about movement was also second to perceptual features in frequency (186 to 107), and neither gesture nor speech was significantly tied to performance in our modeling tasks ($r = .29, p < .29$ for gesture, $r = .27, p < .32$). Instead, we found the highest correlation (and highest frequency) in speech and gestures relating to perceptual features ($r = .61, p < .025$ for gesture, $r = .58, p < .025$ for speech). Ehrlich instead found only a negative correlation between modeling performance and perceptual feature coding, both from speech and from gesture.

We are then left to wonder about the rather drastic differences in our findings. What might account for both the frequency and correlation differences in movement versus perceptual feature strategies? Although of course we cannot know for certain, we hinted at some of the possibilities above: the "embodied" nature of our tasks, having the subjects hold onto an object representative of the solution they were striving for, the participants being' "primed" for certain kinds of responses in the earlier studies, and the differences in age all may account for some of the differences. The kind of mental processes involved in a mental transformation task are not all that different (necessarily) from those involved in modeling an object with the PopCAD (for example) - a robust mental image of the shape in question is likely a boon in either case. However, a model can be built step-by-step, and the results observed and reflected upon. When picking a correct shape in a mental transformation task, one may mentally operate upon features of the shape in a step-by-step manner, but there is no opportunity to reflect upon various strategies, a holistic decision has to be made. In a step-by-step modeling process, it seems common (from the data and from our own experiences and intuitions) for a modeling "step" to focus on a perceptual feature of the shape being modeled (e.g. the next segment in a path or the top point in a pyramid-shaped hull), and to do so in a very conscious way. Additionally, subjects in our study were allowed to continue holding the model while they gave their strategy, providing a ready "facsimile" on which to project their modeling intentions. These factors may have "paved the way" for a high number of speech and gesture about perceptual features of the models, as the step-wise nature of the task and the physical surroundings lead themselves toward thinking in terms of the characteristics of the shapes. Although we observed a high number of expressions coded as movement, there is nothing inherently "movement-oriented" in 3D modeling (one's body moves in using our devices, of course, but models can be created in other ways, e.g. strictly from text coordinates). In contrast, a mental transformation task is, explicitly asking the user to "move" the object, mentally, into the correct formation. Thus it is unsurprising that the examiners in Ehrlich's study repeatedly used the word "move" and appealed to references about movement (as noted above). It is equally unsurprising then, that by using this sort of language and then looking at "movement" as a gestural and spoken strategy, many instances were found, as the task and the instructions surrounding the task are both "movement oriented" in a way that the modeling tasks in our study were not.

One of the other major findings in [38] and [85] is that significant performance differences exist between genders on these tasks, and are evident at younger ages than previously thought. Existing research at the time claimed that gendered differences in spatial reasoning developed around puberty, but that several studies had challenges this assumption. In either case, gender differences should have shown up in our study based on the age range of our subjects (11-18). Boys did outperform girls in session one of the modeling task, though it was not by a terribly significant portion (4%), they did model faster than the girls in each round of the modeling exercise (by about 7 minutes total in the first session, 5 minutes overall in the second session), and they produced more speech instances than the girls did overall (275 to 260), although since there were more boys in the study, this advantage is negligible at best. Interestingly, our findings had girls performing better in many areas; they outperformed boys in the second modeling session by 9%, and across both sessions by almost 3%. Despite a disadvantage in numbers, girls produced more gestures (225 to 188) including those most closely linked to modeling success, perceptual features (96 to 84). Girls also produced more speech elements about perceptual features (97 to 89). The results from our mental transformation task have boys and girls performing about equally, with girls edging out the boys by one tenth of a percent (85.3% to 85.2%, respectively).

Did we have an exceptionally bright group of girls? Possibly, though no independent tests were done for intelligence or other factors that would have indicated an advantage - remember, the girls who enrolled in the study averaged almost a full year younger than the boys (13 years, 7 months for girls and 14 years 6 months for boys), so age and experience advantages are unlikely (none of the girls reported any previous 3D modeling experience). Although the nature of the modeling exercises in our study were more piecemeal, possibly allowing girls more of a chance to reflect and correct their mistakes than in the mental transformation tasks[1] , we saw no significant difference in the MTT tasks we administered (in fact girls did slightly better).

[1] There is some evidence, relayed in [38], that girls tend to utilize a step-by-step strategy in mental rotation problems, whereas boys tend to deal with the whole shape at once.

One possibility is that modeling with the sorts of devices we created are somehow more beneficial to girls than to boys; that the spatial reasoning advantages that boys have are either negated, or that the types of modeling exercises we did significantly altered boys' normal spatial reasoning strategies, which has been known to have a detrimental effect on performance[21][89]. Plenty of other possibilities exist (e.g., the girls simply tried harder) and there is no clear way of determining the source(s) for our results, so we hesitate to make any claims. However, we find it encouraging that girls were able to perform (even out perform) when compared to the boys in our study.

5.1.2 Commonalities

Despite the differences mentioned in the previous section, some of our observations did agree (or at least failed to disagree) with the previous studies. In Ehrlich's study as well as ours, the study population improved overall. In each case, girls improved by a markedly greater percentage, whereas boys improved less so, and in some cases performance actually decreased (in our second session overall and in the post-test for Ehrlich's "Imagine Movement" condition). Although flip-flopped in order, movement and perceptual feature strategies were the most common, with perceptual whole instances far behind. Generally speaking, gesture expressions deemed most "task-appropriate" (per our discussion in the previous section) served as the highest observed correlation to modeling success; perceptual feature gesturing in our study, movement gesturing in Ehrlich's study. This is, we believe, the main "gist" of both these experiments as far as gesture analysis goes - that gesture of a strategy appropriate to the task at hand is correlated with success on that task, more so than speech alone. This holds with the core of Goldin-Meadow's findings, that gesture is a window into the cognitive process and that by analyzing gesture we can gain insight into the mind of the learner.

5.2 Age

One of the more profound and noticeable results from the PopCAD and SnapCAD study was the difference in modeling success between the devices in the first round, and how much that difference was erased on the second round. In the first session, users modeling with the PopCAD correctly modeled 75% of the given shapes, the highest percentage of any device in any round. Conversely, those starting on the SnapCAD modeled only 34% of their shapes properly. Given just this data, we might be tempted to conclude that the PopCAD is a much easier introductory device - or that the SnapCAD is insufferably difficult. However, when we factor in the second round data, in which each subject modeled on the device they did not use the first time around, a different picture emerges: PopCAD modelers in the second round modeled 65% of the shapes correctly while SnapCAD modelers achieved a 56% success rate. If we track each group (let us call the first round PopCAD modelers group A, and the first round SnapCAD modelers group B), we would be sorely tempted to declare that the groups themselves are unevenly talented: Group A scored 75% on PopCAD and 56% on SnapCAD, while group B scored 65% on PopCAD and 34% on SnapCAD.

Since we did not perform a intelligence test or any sort of generalizable aptitude test, we are left to guess using other means. Given the massive development (both cognitively and physically) that occurs between the age extremes in our subject population (11 to 18), it would be tempting (and even logical) to assume that the older subjects would perform much better on the modeling tasks than their younger counterparts. As it turns out, the average age of group A was higher than that of group B - but only by 3 months (group A average age was 14, group B was 13.75). We found a very modest correlation ($r = .39, p < .15$) between age and the number of correctly modeled shapes, suggesting that while not to be overlooked, it may play less of a role then we would have suspected. It is also possible that the statistics are slightly misleading here - the subject population was weighted toward the younger end of the spectrum: the average age was 13.8, while median age was 13.5, and the mode was 12 years old. Meaning the few older participants would have had to perform impossibly brilliantly (i.e., higher than the highest possible score) for a strong age to performance correlation to show up. In keeping with these findings, we also found no real correlation between age and overall performance on the mental transformation tasks ($r = .23, p < .45$). This data of course does not discount that age plays a factor, nor that group A may have been more talented than group B in the PopCAD/SnapCAD; simply that within rough parameters, age matters, just not as much as one might think. Take for example our oldest participant, an 18 year-old boy. He correctly performed 11 of the 24 modeling tasks, while the four 12 year-old participants scored 14, 11, 12, and 11. Our youngest participant, and 11 year old, reproduced seven shapes correctly, while a 13 year old did five correctly, and a 14 year old got six right.

5.3 Shape Complexity

In order to attempt to judge each shape's complexity, we sought out a previously-defined set of criteria by which to judge "complexity". As it turns out, there is a long and thorough discussion of complexity in relation to **two-dimensional** shapes, starting seemingly with Fred Attneave and Malcolm Arnoult[16][15] in the mid 1950's, who define methods of generating random two-dimensional shapes and examine their physical characteristics in relation to their judged complexity. As it turns out (in [15] as well as others' follow-up work) the "Number of Turns" in the shape was responsible for significant amount (nearly 80% in Attneave's study) of the preceived complexity of a shape. "Number of Turns" is defined as, "the number of maxima (regardless of sign) in one cycle of the function relating curvature to distance along the contour. This function is a series of spikes for any angular shape, and a step-function for any curved shape…" (see pp. 226 of the aforementioned article). Symmetry, angular variability, and squared perimeter over area also had some affect.

However, as it seems unclear to us how one might adapt a "Number of Turns" rating to a true **three dimensional** model. Although many studies claim to have studied complexity in relation to mental transformation tasks, starting with Shepard and Metzler[125], who instead used perspective line drawings, not actual physical models. This had an advantage for the types of mental rotation tasks they were performing (recognition of matching pairs), and similarly set off a wave of studies using the same (or similar) "faux 3D" stimuli[95][126][133].

All of which leads us to determine (as best we way) the complexity of the shapes we presented in the study, as a way of teasing out any correlation between complexity and performance. In lieu of attempting an exact number of turns estimate, we included three criteria: (a) the minimum number of lights necessary to guarantee the correct shape,[2] (b) the number of faces (for convex hull models only), the number of line segments (for path models only), or the number of distinct branches (for tree models only), and (c) a symmetry score based on number of lines of symmetry, from 3 (indicating asymmetry) to 0 (indicating 3 or more lines of symmetry). The scaling for symmetry comes from the belief that indicators (a) and (b) above are more closely aligned with Attneave's "number of turns" metric (being highly correlated to perceived difficulty), while symmetry was much less correlated to complexity (although symmetry did still play a part), so we made the scale as low as possible so that it would weigh less on the overall complexity score of a model. So, for example, a regular octahedron would have 6 points, 8 faces, and a 0 symmetry score for an overall difficulty score of 14. The complexity score of each shape is shown in 5.1 next to the number of times it was modeled correctly. The shapes in each session were of course different, but are labeled the same in this table, indicating the order in which they were presented.

Table 5.1: Complexity of Models and Modeling Performance (CH = Convex Hull, P = Path, T = Minimal Spanning Tree)

	CH1	CH2	CH3	CH4	P1	P2	P3	P4	T1	T2	T3	T4
Session 1												
Complexity Score	14	19	19	19	7	18	20	24	9	13	17	17
Performance of 19	8	13	8	11	17	14	8	9	12	8	8	11
Session 2												
Complexity Score	12	20	15	13	14	14	18	17	21	17	16	25
Performance of 16	7	9	11	11	11	13	8	12	7	11	7	9

One might expect to see a strong negative correlation between a given model's complexity score and the number of subjects who were able to model it correctly, however the observed correlation was only moderate: $r = -0.41, p < .05$ over both sessions.

[2] In minimal spanning tree models where a placement of lights results in several possible correct formations, only one of which is the desired shape, we add points necessary to "force" the correct representation.

5.4 Error Analysis

For each modeling task, one of seven error codes was recorded, based on the outcome of the task. A more complete detail of the error codes can be found in table 4.1, this section focuses instead on what significance (if any) these error codes have on our observations[3] . To briefly recount the codes and their associations, then: C = correct, EP = error in proportion (general shape is correct, but model is too tall, too wide, etc.), E1 = error in recognition (subject had the correct shape but did not recognize it), E2 = error in belief (thought the shape was correct when it was not), E3 = error in implementation (knew shape was incorrect, but knew why), E4 = error in strategy (subject knew shape was incorrect, but could not explain why), I = incomplete (includes giving up, asking to move on).

Table 5.2: Error Code Breakdown. C = Correct, EP = Error in Proportion, E1 = Error in Recognition, E2 = Error in Belief, E3 = Error in Implementation, E4 = Error in Strategy, I = Incomplete.

	Total	*PopCAD*	*SnapCad*	*Girls*	*Boys*
C	221	98	123	100	121
EP	42	17	25	20	22
$E1$	1	0	1	1	0
$E2$	41	23	18	13	28
$E3$	2	0	2	0	2
$E4$	15	11	4	6	9
I	62	40	22	28	34

5.5 Cross-Study Comparisons

When we compare across studies, we see the first study resulted in 80% (24 of 30) correct models, the second UCube study resulted in 82% (41 of 50), while the third study resulted in only 243/420, or about 58%.

Complexity score

[3] Again, this data is from the PopCAD/SnapCAD study only.

5.6 Demographics

An important factor to take into account here is the subject demographics. Only two participants in the PopCAD/SnapCAD study had indicated previous experience with 3D modeling in any capacity, and the study site was a drop-in program serving primarily disadvantaged youth in a low socioeconomic neighborhood. In contrast, the two earlier studies with the UCube were performed at a fairly affluent, predominantly Caucasian middle school, with their multimedia class, most of whom had been exposed to 3D modeling software (like Goggle SketchUp[129]) as part of the multimedia curriculum. So while we saw performance in the 80% range for the middle school and less than 60% for our drop-in program, not all of the difference is likely from modeling ability or shape complexity or the ease of use of the UCube.

Chapter 6

Vision

Given the different medium of the pop-up book (paper as opposed to circuit boards), it is worth exploring the possibilities afforded by a cheaper, more flexible material. For instance, the flexibility of paper might provide the means for new types of modeling actions. It is plausible to imagine paper tabs or other mechanisms that perturb the LEDs off the integer lattice, or alter the overall topology in such a way that new shapes are possible (e.g. by deforming an equidistant grid into a spherical shape). There may be additional sensors or hardware that could be embedded into the book to provide new functionality (rotation, proximity, pressure). Additionally, due the inexpensive and portable nature of the pop-up book, it is worth exploring the sorts of interactions that could occur between several pop-up books (e.g., extending the input field to include two or more grids, networked interactions like cooperative modeling tasks, or competitive games like 3D-battleship). By using paper as a material to think with, we may find further possibilities as development continues.

Chapter 7

Conclusions

Bibliography

[1] The printed world. http://www.economist.com/node/18114221, 2011.

[2] 123d catch. http://www.123dapp.com/catch, 2013.

[3] Makerbot. http://www.makerbot.com/, 2013.

[4] Ponoko. http://www.ponoko.com/, 2013.

[5] Stratasys. http://www.stratasys.com/, 2013.

[6] Thingiverse. http://www.thingiverse.com/, 2013.

[7] Tinkercad. http://www.tinkercad.com/, 2013.

[8] Dor Abrahamson and Dragan Trninic. Toward an embodied-interaction design framework for mathematical concepts. In Proceedings of the 10th International Conference on Interaction Design and Children, IDC '11, pages 1–10, New York, NY, USA, 2011. ACM.

[9] S. Ananthanarayan, K. Siek, and M. Eisenberg. Towards the Crafting of Personal Health Technologies. Submitted for publication. 2013.

[10] C. Anderson. Makers: The New Industrial Revolution. Random House, 2012.

[11] Alissa N. Antle. Designing tangibles for children: what designers need to know. In CHI '07 Extended Abstracts on Human Factors in Computing Systems, CHI EA '07, pages 2243–2248, New York, NY, USA, 2007. ACM.

[12] Alissa N. Antle, Milena Droumeva, and Daniel Ha. Thinking with hands: an embodied approach to the analysis of children's interaction with computational objects. In CHI '09 Extended Abstracts on Human Factors in Computing Systems, CHI EA '09, pages 4027–4032, New York, NY, USA, 2009. ACM.

[13] Alissa N. Antle and Sijie Wang. Comparing motor-cognitive strategies for spatial problem solving with tangible and multi-touch interfaces. In Proceedings of the 7th International Conference on Tangible, Embedded and Embodied Interaction, TEI '13, pages 65–72, New York, NY, USA, 2013. ACM.

[14] Alissa N. Antle, Alyssa F. Wise, and Kristine Nielsen. Towards utopia: designing tangibles for learning. In Proceedings of the 10th International Conference on Interaction Design and Children, IDC '11, pages 11–20, New York, NY, USA, 2011. ACM.

[15] Fred Attneave. Physical determinants of the judged complexity of shapes. Journal of Experimental Psychology, 53(4):221, 1957.

[16] Fred Attneave and Malcolm D Arnoult. The quantitative study of shape and pattern perception. Psychological bulletin, 53(6):452, 1956.

[17] Saskia Bakker, Alissa N Antle, and Elise Van Den Hoven. Embodied metaphors in tangible interaction design. Personal and Ubiquitous Computing, 16(4):433–449, 2012.

[18] Saskia Bakker, Elise van den Hoven, and Alissa N Antle. Moso tangibles: evaluating embodied learning. In Proceedings of the fifth international conference on Tangible, embedded, and embodied interaction, pages 85–92. ACM, 2011.

[19] C Bradford Barber, David P Dobkin, and Hannu Huhdanpaa. The quickhull algorithm for convex hulls. ACM Transactions on Mathematical Software (TOMS), 22(4):469–483, 1996.

[20] Nicholas A Basbanes. On Paper: The Everything of Its Two-Thousand-Year History. Random House LLC, 2013.

[21] Sian L Beilock, Thomas H Carr, Clare MacMahon, and Janet L Starkes. When paying attention becomes counterproductive: impact of divided versus skill-focused attention on novice and experienced performance of sensorimotor skills. Journal of Experimental Psychology: Applied, 8(1):6, 2002.

[22] RQRQ Berry, G Bull, C Browning, C Thomas, K Starkweather, and J Aylor. Preliminary considerations regarding use of digital fabrication to incorporate engineering design principles in elementary mathematics education. Contemporary Issues in Technology and Teacher Education, 10(2):167–172, 2010.

[23] Allen Bevans, Ying-Ting Hsiao, and Alissa Antle. Supporting children's creativity through tangible user interfaces. In CHI '11 Extended Abstracts on Human Factors in Computing Systems, CHI EA '11, pages 1741–1746, New York, NY, USA, 2011. ACM.

[24] Otakar Borvka. O jistém problému minimálním. 1926.

[25] R Breckinridge Church and Susan Goldin-Meadow. The mismatch between gesture and speech as an index of transitional knowledge. Cognition, 23(1):43–71, 1986.

[26] Jack Breen, Robert Nottrot, and Martijn Stellingwerff. Tangible virtualityperceptions of computer-aided and physical modelling. Automation in Construction, 12(6):649–653, 2003.

[27] Erwin H Brinkmann. Programed instruction as a technique for improving spatial visualization. Journal of Applied Psychology, 50(2):179, 1966.

[28] Norman Brosterman, Kiyoshi Togashi, and Eric Himmel. Inventing kindergarten. HN Abrams New York, NY, 1997.

[29] Craig Brown and Amy Hurst. Viztouch: automatically generated tactile visualizations of coordinate spaces. In Proceedings of the Sixth International Conference on Tangible, Embedded and Embodied Interaction, TEI '12, pages 131–138, New York, NY, USA, 2012. ACM.

[30] Leah Buechley and Michael Eisenberg. The lilypad arduino: Toward wearable engineering for everyone. Pervasive Computing, IEEE, 7(2):12–15, 2008.

[31] Leah Buechley, Mike Eisenberg, Jaime Catchen, and Ali Crockett. The lilypad arduino: using computational textiles to investigate engagement, aesthetics, and diversity in computer science education. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '08, pages 423–432, New York, NY, USA, 2008. ACM.

[32] Leah Buechley, Sue Hendrix, and Mike Eisenberg. Paints, paper, and programs: first steps toward the computational sketchbook. In Proceedings of the 3rd International Conference on Tangible and Embedded Interaction, pages 9–12. ACM, 2009.

[33] Leah Buechley, David Mellis, Hannah Perner-Wilson, Emily Lovell, and Bonifaz Kaufmann. Living wall. In Proceedings of the international conference on Multimedia - MM '10, page 1401, New York, New York, USA, October 2010. ACM Press.

[34] Leah Buechley and Hannah Perner-Wilson. Crafting technology. ACM Transactions on Computer-Human Interaction, 19(3):1–21, October 2012.

[35] A. Clark. Being There: Putting Brain, Body, and World Together Again. A Bradford book. MIT Press, 1998.

[36] Marcelo Coelho, L Hall, J Berzowska, and P Maes. Pulp-based computing: a framework for building computers out of paper. ... Human Factors in Computing ..., pages 3527–3528, 2009.

[37] Matthew Conway, Steve Audia, Tommy Burnette, Dennis Cosgrove, and Kevin Christiansen. Alice: lessons learned from building a 3d system for novices. In Proceedings of the SIGCHI conference on Human Factors in Computing Systems, CHI '00, pages 486–493, New York, NY, USA, 2000. ACM.

[38] Stacy B Ehrlich, Susan C Levine, and Susan Goldin-Meadow. The importance of gesture in children's spatial reasoning. Developmental psychology, 42(6):1259, 2006.

[39] M. Eisenberg, A. Eisenberg, S. Hendrix, G. Blauvelt, D. Butter, J. Garcia, R. Lewis, and T. Nielsen. As we may print: new directions in output devices and computational crafts for children. In Proceedings of the 2003 conference on Interaction design and children, IDC '03, pages 31–39, New York, NY, USA, 2003. ACM.

[40] Michael Eisenberg. Pervasive fabrication: Making construction ubiquitous in education. In Proceedings of the Fifth IEEE International Conference on Pervasive Computing and Communications Workshops, PERCOMW '07, pages 193–198, Washington, DC, USA, 2007. IEEE Computer Society.

[41] Michael Eisenberg. Educational fabrication, in and out of the classroom. In Society for Information Technology & Teacher Education International Conference, volume 2011, pages 884–891, 2011.

[42] Michael Eisenberg, Nwanua Elumeze, Michael MacFerrin, and Leah Buechley. Children's programming, reconsidered: settings, stuff, and surfaces. In Proceedings of the 8th International Conference on Interaction Design and Children, IDC '09, pages 1–8, New York, NY, USA, 2009. ACM.

[43] Michael Eisenberg, Wendy Mackay, Allison Druin, Sheila Lehman, and Mitchel Resnick. Real meets virtual: blending real-world artifacts with computational media. In Conference Companion on Human Factors in Computing Systems, CHI '96, pages 159–160, New York, NY, USA, 1996. ACM.

[44] Michael Eisenberg, Ann Nishioka, and M. E. Schreiner. Helping users think in three dimensions: steps toward incorporating spatial cognition in user modelling. In Proceedings of the 2nd international conference on Intelligent user interfaces, IUI '97, pages 113–120, New York, NY, USA, 1997. ACM.

[45] Fab@Home. www.fabathome.org. 2011.

[46] Desktop Factory. www.desktopfactory.com/. 2011.

[47] ES Ferguson. Engineering and the Mind's Eye. The MIT Press, 1992.

[48] Thomas Fischer and Wing Lau. Marble track music sequencers for children. In Proceedings of the 2006 conference on Interaction design and children, IDC '06, pages 141–144, New York, NY, USA, 2006. ACM.

[49] Mark D Folk and R Duncan Luce. Effects of stimulus complexity on mental rotation rate of polygons. Journal of Experimental Psychology: Human Perception and Performance, 13(3):395, 1987.

[50] Sean Follmer and Hiroshi Ishii. Kidcad: digitally remixing toys through tangible tools. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '12, pages 2401–2410, New York, NY, USA, 2012. ACM.

[51] Sean Follmer, Micah Johnson, Edward Adelson, and Hiroshi Ishii. deform: an interactive malleable surface for capturing 2.5d arbitrary objects, tools and touch. In Proceedings of the 24th annual ACM symposium on User interface software and technology, UIST '11, pages 527–536, New York, NY, USA, 2011. ACM.

[52] F. Froebel and J. Jarvis. The Education of Man. A. Lovell Company, New York, 1886.

[53] Neil Gershenfeld. Fab: The Coming Revolution on Your Desktop–from Personal Computers to Personal Fabrication. Basic Books, Inc., New York, NY, USA, 2007.

[54] Wooi Boon Goh, L. L. Chamara Kasun, Fitriani, Jacquelyn Tan, and Wei Shou. The i-cube: design considerations for block-based digital manipulatives and their applications. In Proceedings of the Designing Interactive Systems Conference, DIS '12, pages 398–407, New York, NY, USA, 2012. ACM.

[55] H. Nusbaum S. D. Delly Goldin-Meadow, S. and S. Wagner. Explaining math: Gesturing lightens the load. Psychological Science, pages pp. 516–22, 1991.

[56] Susan Goldin-Meadow. Hearing gesture: How our hands help us think. Harvard University Press, 2005.

[57] Ira Greenberg. Processing: creative coding and computational art. Apress, 2007.

[58] Mark D. Gross and Michael Eisenberg. Why toys shouldn't work "like magic": Children's technology and the values of construction and control. In Proceedings of the The First IEEE International Workshop on Digital Game and Intelligent Toy Enhanced Learning, DIGITEL '07, pages 25–32, Washington, DC, USA, 2007. IEEE Computer Society.

[59] Elizabeth G Hainstock. The Essential Montessori. New American Library, 1978.

[60] B. Hart. Will 3d printing change the world? http://www.forbes.com/sites/gcaptain/2012/03/06/will-3d-prin 2012.

[61] George W Hart. Procedural generation of sculptural forms. Bridges, 2008:209, 2008.

[62] Susan Lee Hendrix. Popup Workshop: Computationally Enhanced Paper Engineering for Children. PhD thesis, University of Colorado, 2008.

[63] Mark Howison, Dragan Trninic, Daniel Reinholz, and Dor Abrahamson. The mathematical imagery trainer: from embodied interaction to conceptual learning. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pages 1989–1998. ACM, 2011.

[64] http://507movements.com/. 507 mechanical movements. 2013.

[65] Yingdan Huang and Michael Eisenberg. Easigami: virtual creation by physical folding. In Proceedings of the Sixth International Conference on Tangible, Embedded and Embodied Interaction, TEI '12, pages 41–48, New York, NY, USA, 2012. ACM.

[66] Yingdan Huang, Mark D Gross, Ellen Yi-Luen Do, and Mike Eisenberg. Easigami: a reconfigurable folded-sheet tui. In Proceedings of the 3rd International Conference on Tangible and Embedded Interaction, TEI '09, pages 107–112, New York, NY, USA, 2009. ACM.

[67] Amy Hurst and Shaun Kane. Making "making" accessible. In Proceedings of the 12th International Conference on Interaction Design and Children, IDC '13, pages 635–638, New York, NY, USA, 2013. ACM.

[68] Barbel Inhelder and Jean Piaget. The Growth of Logical Thinking from Childhood to Adolescence : An Essay on the Construction of Formal Operational Structures. Basic, 1958.

[69] Instructables. http://instructables.com.

[70] Hiroshi Ishii. Tangible bits: beyond pixels. In Proceedings of the 2nd international conference on Tangible and embedded interaction, TEI '08, pages xv–xxv, New York, NY, USA, 2008. ACM.

[71] Hiroshi Ishii and Brygg Ullmer. Tangible bits: towards seamless interfaces between people, bits and atoms. In Proceedings of the ACM SIGCHI Conference on Human factors in computing systems, CHI '97, pages 234–241, New York, NY, USA, 1997. ACM.

[72] Sam Jacoby and Leah Buechley. Drawing the electric. In Proceedings of the 12th International Conference on Interaction Design and Children - IDC '13, pages 265–268, New York, New York, USA, June 2013. ACM Press.

[73] Gabe Johnson, Mark Gross, Ellen Yi-Luen Do, and Jason Hong. Sketch it, make it: sketching precise drawings for laser cutting. In CHI '12 Extended Abstracts on Human Factors in Computing Systems, CHI EA '12, pages 1079–1082, New York, NY, USA, 2012. ACM.

[74] Samuel Johnson and AnnMarie P. Thomas. Squishy circuits: a tangible medium for electronics education. In CHI '10 Extended Abstracts on Human Factors in Computing Systems, CHI EA '10, pages 4099–4104, New York, NY, USA, 2010. ACM.

[75] Yasmin B. Kafai, Kylie A. Peppler, Quinn Burke, Michael Moore, and Diane Glosson. Frbel's forgotten gift: textile construction kits as pathways into play, design and computation. In Proceedings of the 9th International Conference on Interaction Design and Children, IDC '10, pages 214–217, New York, NY, USA, 2010. ACM.

[76] Yoshihiro Kawahara, Steve Hodges, Benjamin S Cook, and Gregory D Abowd. Instant Inkjet Circuits : Lab-based Inkjet Printing to Support Rapid Prototyping of UbiComp Devices. pages 363–372, 2013.

[77] Yoshihiro Kawahara, Steve Hodges, Benjamin S. Cook, Cheng Zhang, and Gregory D. Abowd. Instant inkjet circuits. In Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing - UbiComp '13, page 363, New York, New York, USA, September 2013. ACM Press.

[78] Scott R. Klemmer, Björn Hartmann, and Leila Takayama. How bodies matter: five themes for interaction design. In Proceedings of the 6th conference on Designing Interactive systems, DIS '06, pages 140–149, New York, NY, USA, 2006. ACM.

[79] Naoya Koizumi, Kentaro Yasu, Angela Liu, Maki Sugimoto, and Masahiko Inami. Animated paper. Computers in Entertainment, 8(2):1, December 2010.

[80] Dennis Krannich, Bernard Robben, and Sabrina Wilske. Digital fabrication for educational contexts. In Proceedings of the 11th International Conference on Interaction Design and Children, IDC '12, pages 375–376, New York, NY, USA, 2012. ACM.

[81] Joseph B Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. Proceedings of the American Mathematical society, 7(1):48–50, 1956.

[82] G. Lakoff and R. Nuñez. Where Mathematics Come From: How The Embodied Mind Brings Mathematics Into Being. Basic Books, Inc., 2001.

[83] Ben Leduc-Mills and Michael Eisenberg. The ucube: a child-friendly device for introductory three-dimensional design. In Proceedings of the 10th International Conference on Interaction Design and Children, IDC '11, pages 72–80, New York, NY, USA, 2011. ACM.

[84] Ben Leduc-Mills, Halley Profita, and Michael Eisenberg. "seeing solids" via patterns of light: evaluating a tangible 3d-input device. In Proceedings of the 11th International Conference on Interaction Design and Children, IDC '12, pages 377–380, New York, NY, USA, 2012. ACM.

[85] Susan C Levine, Janellen Huttenlocher, Amy Taylor, and Adela Langrock. Early sex differences in spatial skill. Developmental psychology, 35(4):940, 1999.

[86] Hod Lipson and Melba Kurman. Factory@ home: The emerging economy of personal fabrication. A report commissioned by the US Office of Science and Technology Policy, 2010.

[87] Hod Lipson, Francis C Moon, Jimmy Hai, and Carlo Paventi. 3-d printing the history of mechanisms. Journal of Mechanical Design, 127:1029, 2005.

[88] John Lloyd. Quickhull3d: A robust 3d convex hull algorithm in java.

[89] Rafer Lutz, Daniel M Landers, and Darwyn E Linder. Procedural variables and skill level influences on pre-performance mental practice efficacy. Journal of Mental Imagery, 2001.

[90] Evan Malone and Hod Lipson. Fab@ home: the personal desktop fabricator kit. Rapid Prototyping Journal, 13(4):245–255, 2007.

[91] James H Mathewson. Visual-spatial thinking: An aspect of science overlooked by educators. Science Education, 83(1):33–54, 1999.

[92] Timothy S. McNerney. From turtles to tangible programming bricks: explorations in physical language design. Personal Ubiquitous Comput., 8(5):326–337, September 2004.

[93] Arduino Mega. arduino.cc/en/main/arduinoboardmega2560. 2011.

[94] David A. Mellis, Sam Jacoby, Leah Buechley, Hannah Perner-Wilson, and Jie Qi. Microcontrollers as material: crafting circuits with paper, conductive ink, electronic components, and an "untoolkit". In Proceedings of the 7th International Conference on Tangible, Embedded and Embodied Interaction, TEI '13, pages 83–90, New York, NY, USA, 2013. ACM.

[95] Jacqueline Metzler and Roger N Shepard. Transformational studies of the internal representation of three-dimensional objects. 1974.

[96] Jason Mickelson and Wendy Ju. Math propulsion: engaging math learners through embodied performance & visualization. In Proceedings of the fifth international conference on Tangible, embedded, and embodied interaction, pages 101–108. ACM, 2011.

[97] Maria Montessori. The Montessori Method. Frederick Stokes Co., New York, NY, 1912.

[98] Catarina Mota. The rise of personal fabrication. In Proceedings of the 8th ACM conference on Creativity and cognition, pages 279–288. ACM, 2011.

[99] Stefanie Mueller, Bastian Kruck, and Patrick Baudisch. LaserOrigami. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI '13, page 2585, New York, New York, USA, April 2013. ACM Press.

[100] Stefanie Mueller, Pedro Lopes, and Patrick Baudisch. Interactive construction: interactive fabrication of functional mechanical devices. In Proceedings of the 25th annual ACM symposium on User interface software and technology, UIST '12, pages 599–606, New York, NY, USA, 2012. ACM.

[101] Ricardo Nemirovsky, Cornelia Tierney, and Tracey Wright. Body motion and graphing. Cognition and instruction, 16(2):119–172, 1998.

[102] Nora Newcombe and Janellen Huttenlocher. Making space: The development of spatial representation and reasoning. The MIT Press, 2003.

[103] Brian N. O'Connell. The development of the paperbots robotics kit for inexpensive robotics education activities for elementary students. Master's thesis, Tufts University, 2013.

[104] S. Papert. Mindstorms: Children, Computers, and Powerful Ideas. Basic Books, Inc., 1980.

[105] Amanda J. Parkes, Hayes Solos Raffle, and Hiroshi Ishii. Topobo in the wild: longitudinal evaluations of educators appropriating a tangible interface. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '08, pages 1129–1138, New York, NY, USA, 2008. ACM.

[106] Jean Piaget and Barbel Inhelder. The childs conception of space (fj langdon & jl lunzer, trans.). New York, 1967.

[107] Franco P. Preparata and Se June Hong. Convex hulls of finite sets of points in two and three dimensions. Communications of the ACM, 20(2):87–93, 1977.

[108] FP Preparatat and Michael Ian Shamos. Computational geometry: an introduction. 1985.

[109] Processing. www.processing.org. 2011.

[110] The RepRap Project. reprap.org/wiki/main_page. 2011.

[111] Jie Qi and Leah Buechley. Electronic popables: exploring paper-based computing through an interactive pop-up book. In Proceedings of the fourth international conference on Tangible, embedded, and embodied interaction, TEI '10, pages 121–128, New York, NY, USA, 2010. ACM.

[112] Jie Qi and Leah Buechley. Animating paper using shape memory alloys. In Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems - CHI '12, page 749, New York, New York, USA, May 2012. ACM Press.

[113] Hayes Solos Raffle. Sculpting behavior: a tangible language for hands-on play and learning. PhD thesis, MIT, Cambridge, MA, USA, 2008. AAI0821001.

[114] Hayes Solos Raffle, Amanda J. Parkes, and Hiroshi Ishii. Topobo: a constructive assembly system with kinetic memory. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '04, pages 647–654, New York, NY, USA, 2004. ACM.

[115] Betty Repacholi and Alison Gopnik. Early reasoning about desires: Evidence from 14- and 18-month-olds. In Developmental Psychology, pages 12–21, 1997.

[116] Mitchel Resnick. All i really need to know (about creative thinking) i learned (by studying how children learn) in kindergarten. In Proceedings of the 6th ACM SIGCHI conference on Creativity & cognition, C&C '07, pages 1–6, New York, NY, USA, 2007. ACM.

[117] Mitchel Resnick, Fred Martin, Robert Berg, Rick Borovoy, Vanessa Colella, Kwin Kramer, and Brian Silverman. Digital manipulatives: new toys to think with. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '98, pages 281–287, New York, NY, USA, 1998. ACM Press/Addison-Wesley Publishing Co.

[118] Mitchel Resnick and Brian Silverman. Some reflections on designing construction kits for kids. In Proceedings of the 2005 conference on Interaction design and children, IDC '05, pages 117–122, New York, NY, USA, 2005. ACM.

[119] Rhino. www.rhino3d.com. 2011.

[120] Eric Rosenbaum and Jay Silver. Makey makey: improvising tangible and nature-based user interfaces. In Proceedings of the Sixth International Conference on Tangible, Embedded and Embodied Interaction, TEI '12, pages 367–370, New York, NY, USA, 2012. ACM.

[121] Greg Saul, Cheng Xu, and Mark D. Gross. Interactive paper devices. In Proceedings of the fourth international conference on Tangible, embedded, and embodied interaction - TEI '10, page 205, New York, New York, USA, January 2010. ACM Press.

[122] Eric Schweikardt, Nwanua Elumeze, Mike Eisenberg, and Mark D Gross. A tangible construction kit for exploring graph theory. In Proceedings of the 3rd International Conference on Tangible and Embedded Interaction, TEI '09, pages 373–376, New York, NY, USA, 2009. ACM.

[123] Eric Schweikardt and Mark D. Gross. roblocks: a robotic construction kit for mathematics and science education. In Proceedings of the 8th international conference on Multimodal interfaces, ICMI '06, pages 72–75, New York, NY, USA, 2006. ACM.

[124] Shapeways. www.shapeways.com. 2011.

[125] RN Shepard and Jacqueline Metzler. Mental rotation of three-dimensional objects. 1971.

[126] Shenna Shepard and Douglas Metzler. Mental rotation: Effects of dimensionality of objects and type of task. Journal of Experimental Psychology: Human Perception and Performance, 14(1):3, 1988.

[127] Francis T Siemankowski and Franklin C MacKnight. Spatial cognition, a success prognosticator in college science courses. 1971.

[128] Arnan Sipitakiat and Nusarin Nusen. Robo-blocks: designing debugging abilities in a tangible programming system for early primary school children. In Proceedings of the 11th International Conference on Interaction Design and Children, IDC '12, pages 98–105, New York, NY, USA, 2012. ACM.

[129] Google SketchUp. sketchup.google.com/. 2011.

[130] Solidways. www.solidworks.com. 2011.

[131] J. P. Spencer, M. Clearfield, D. Corbetta, B. Ulrich, P. Buchanan, and G. Schoner. Moving toward a grand theory of development: In memory of esther thelen. In Child Development, pages 1521–1538, 2006.

[132] Cristina Sylla, Pedro Branco, Sérgio Gonçalves, Clara Coutinho, and Paulo Brito. t-books. In Proceedings of the 11th International Conference on Interaction Design and Children - IDC '12, page 323, New York, New York, USA, June 2012. ACM Press.

[133] Steven G Vandenberg and Allan R Kuse. Mental rotations, a group test of three-dimensional spatial visualization. Perceptual and motor skills, 47(2):599–604, 1978.

[134] Danli Wang, Cheng Zhang, and Hongan Wang. T-maze: a tangible programming tool for children. In Proceedings of the 10th International Conference on Interaction Design and Children, IDC '11, pages 127–135, New York, NY, USA, 2011. ACM.

[135] Ryoichi Watanabe, Yuichi Itoh, Masatsugu Asai, Yoshifumi Kitamura, Fumio Kishino, and Hideo Kikuchi. The soul of activecube: implementing a flexible, multimodal, three-dimensional spatial tangible interface. Comput. Entertain., 2(4):15–15, October 2004.

[136] Michael Philetus Weller, Ellen Yi-Luen Do, and Mark D Gross. Escape machine: teaching computational thinking with a tangible state machine game. In Proceedings of the 7th international conference on Interaction design and children, IDC '08, pages 282–289, New York, NY, USA, 2008. ACM.

[137] Michael Philetus Weller, Mark D. Gross, and Ellen Yi-Luen Do. Tangible sketching in 3d with posey. In CHI '09 Extended Abstracts on Human Factors in Computing Systems, CHI EA '09, pages 3193–3198, New York, NY, USA, 2009. ACM.

[138] Karl D.D. Willis, Juncong Lin, Jun Mitani, and Takeo Igarashi. Spatial sketch: bridging between movement & fabrication. In Proceedings of the fourth international conference on Tangible, embedded, and embodied interaction, TEI '10, pages 5–12, New York, NY, USA, 2010. ACM.

[139] Karl D.D. Willis, Cheng Xu, Kuan-Ju Wu, Golan Levin, and Mark D. Gross. Interactive fabrication: new interfaces for digital fabrication. In Proceedings of the fifth international conference on Tangible, embedded, and embodied interaction, TEI '11, pages 69–72, New York, NY, USA, 2011. ACM.

[140] Margaret Wilson. Six views of embodied cognition. Psychonomic Bulletin Review, 9(4):625–636, 2002.

[141] Lesley Xie, Alissa N. Antle, and Nima Motamedi. Are tangibles more fun?: comparing children's enjoyment and engagement using physical, graphical and tangible user interfaces. In Proceedings of the 2nd international conference on Tangible and embedded interaction, TEI '08, pages 191–198, New York, NY, USA, 2008. ACM.

[142] Kening Zhu, Hideaki Nii, Owen Noel Newton Fernando, and Adrian David Cheok. Selective inductive powering system for paper computing. In Proceedings of the 8th International Conference on Advances in Computer Entertainment Technology - ACE '11, page 1, New York, New York, USA, November 2011. ACM Press.

[143] Amit Zoran and Joseph A. Paradiso. Freed: a freehand digital sculpting tool. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '13, pages 2613–2616, New York, NY, USA, 2013. ACM.

[144] Oren Zuckerman, Saeed Arida, and Mitchel Resnick. Extending tangible interfaces for education: digital montessori-inspired manipulatives. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '05, pages 859–868, New York, NY, USA, 2005. ACM.

Appendix A

Weird Exam Answers

About appendices: Each appendix follow the same page-numbering rules as a regular chapter; the first page of a (multi-page) appendix is not numbered. By the way, the following are supposedly authentic answers to English GCSE exams!

(1) The Greeks were a highly sculptured people, and without them we wouldn't have history. The Greeks also had myths. A myth is a female moth.

(2) Actually, Homer was not written by Homer but by another man of that name.

(3) Socrates was a famous Greek teacher who went around giving people advice. They killed him. Socrates died from an overdose of wedlock. After his death, his career suffered a dramatic decline.

(4) Julius Caesar extinguished himself on the battlefields of Gaul. The Ides of March murdered him because they thought he was going to be made king. Dying, he gasped out: Tee hee, Brutus.

(5) Nero was a cruel tyranny who would torture his subjects by playing the fiddle to them.

(6) In midevil times most people were alliterate. The greatest writer of the futile ages was Chaucer, who wrote many poems and verses and also wrote literature.

(7) Another story was William Tell, who shot an arrow through an apple while standing on his sons head.

(8) Writing at the same time as Shakespeare was Miguel Cervantes. He wrote Donkey Hote. The next great author was John Milton. Milton wrote Paradise Lost. Then his wife died and he wrote Paradise Regained.

(9) During the Renaissance America began. Christopher Columbus was a great navigator who discovered America while cursing about the Atlantic. His ships were called the Nina, the Pinta, and the Santa Fe.

(10) Gravity was invented by Issac Walton. It is chiefly noticeable in the autumn when the apples are falling off the trees.

(11) Johann Bach wrote a great many musical compositions and had a large number of children. In between he practiced on an old spinster which he kept up in his attic. Bach died from 1750 to the present. Bach was the most famous composer in the world and so was Handel. Handel was half German half Italian and half English. He was very large.

(12) Soon the Constitution of the United States was adopted to secure domestic hostility. Under the constitution the people enjoyed the right to keep bare arms.

(13) The sun never set on the British Empire because the British Empire is In the East and the sun sets in the West.

(14) Louis Pasteur discovered a cure for rabbis. Charles Darwin was a naturalist who wrote the Organ of the Species. Madman Curie discovered radio. And Karl Marx became one of the Marx brothers.

Appendix B

Ode to Spot

(Data, Stardate 1403827) (A one-page chapter — page must be numbered!) Throughout the ages, from Keats to Giorchamo, poets have composed "odes" to individuals who have had a profound effect upon their lives. In keeping with that tradition I have written my next poem . . . in honor of my cat. I call it. . . Ode. . . to Spot. (Shot of Geordi and Worf in audience, looking mystified at each other.)

Felus cattus, is your taxonomic nomenclature
an endothermic quadruped, carnivorous by nature?
Your visual, olfactory, and auditory senses
contribute to your hunting skills, and natural defenses.
I find myself intrigued by your sub-vocal oscillations,
a singular development of cat communications
that obviates your basic hedonistic predilection
for a rhythmic stroking of your fur to demonstrate affection.
A tail is quite essential for your acrobatic talents;
you would not be so agile if you lacked its counterbalance.
And when not being utilized to aid in locomotion,
It often serves to illustrate the state of your emotion.

(Commander Riker begins to applaud, until a glance from Counselor Troi brings him to a halt.) Commander Riker, you have anticipated my denouement. However, the sentiment is appreciated. I will continue.

O Spot, the complex levels of behavior you display
connote a fairly well-developed cognitive array.
And though you are not sentient, Spot, and do not comprehend
I nonetheless consider you a true and valued friend.