A Tangible Construction Kit for Exploring Graph Theory

Eric Schweikardt[1], Nwanua Elumeze[2], Mike Eisenberg[2], Mark D Gross[3]

1: Computational Synthesis Lab, Cornell University, Ithaca NY
2: Computer Science Dept, University of Colorado, Boulder CO
3: Computational Design Lab, Carnegie Mellon University, Pittsburgh PA

ABSTRACT

Graphs are a versatile representation of many systems in computer science, the social sciences, and mathematics, but graph theory is not taught in schools. We present our work on Graphmaster, a computationally enhanced construction kit that enables children to build graphs of their own and investigate their properties by experimenting with algorithms that operate on them. The system is distributed; microcontrollers inside each node execute an interpreted language in parallel. Graphmaster, with its magnetic connectors, illuminated edges, and capacitive sensing, encourages children to develop intuitions about connectivity long before they are introduced to the notation and formulas of graph theory.

Keywords

Graphs, education, tangible, toys, games.

INTRODUCTION: WHY WE CARE ABOUT GRAPHS

Graph theory is an important branch of modern mathematics that is used throughout the physical, social, and artificial sciences. For example, it is used in electrical engineering to model circuits and in anthropology, to represent family structures in a society. In computer science, graph theory is fundamental to understanding the properties of data structures and algorithms that operate on them, for example, object hierarchies, Markov models and Bayesian networks, minimum spanning trees, and so on. An excellent popular introduction to important ideas of graph theory, and how they can be applied to an immense variety of deep and practical problems, can be found in [1].

In light of its importance in modern science and engineering, it is regrettable that young people do not encounter graphs in elementary or secondary school. (Under the rubric of "graphing" students are taught to plot functions on an x-y chart, but this is an entirely different topic.) Some students take a graph theory course in college, but most do not.

We aim to remedy this problem by providing a tangible toy that can serve as an interactive platform for teaching, learning about, and playing with, graphs. We describe here Graphmaster, our first design for and implementation of a tangible graph construction kit. The kit comprises physical components for nodes and edges, and the means to program graph constructions using a simple embedded language.

PLAYING WITH GRAPHS

Toys can give children valuable intuitions about complex concepts before they learn about those concepts formally. Architects, for instance, have described how playing with construction toys as children gave them an intuitive sense of structure and balance [2]. Andrea diSessa argues convincingly that the "phenomenological primitives" that children encounter during play are of the utmost importance for future comprehension of scientific and mathematical concepts [3]. The educational benefits of learning a formal concept that relates to previous experience ("oh, so *that's* what you call it") can be significant. Although we don't claim that GraphMaster teaches the formal notation of graph theory, it serves to expose children to graphs and their operations—to strengthen the intuitions of children—before they encounter the relevant mathematics.

There is a particular benefit to be gained by building graphs and experimenting with graph algorithms, as opposed to just studying their properties. In contrast to the idea of teachers transferring knowledge to children, Piaget's *constructivism* describes how children create their own knowledge based on their experience (including their experience with teachers). Seymour Papert coined the term *constructionism* to describe the benefits of actually building something as part of that process [4]. Although knowledge is invisible, a constructed object gives children and their mentors something external to reflect on and critique. Thoughtfully designed construction kits can provide pieces that encourage kids to build meaningful assemblies. Froebel's "gifts," from the mid-1800s, were an early example [5], and have been followed by commercial products like LEGO, Tinker Toys, and Erector Set.

More recently, the availability of tiny and inexpensive microcontrollers has enabled embedded computation in construction kits, bringing with it a wide range of possibilities for interaction and feedback [6-8]. Graphmaster was inspired by light-up edge design of Senspectra, a construction kit from the MIT Media Lab that is intended to model structural strain in truss systems [9].

The spatial, and therefore visual, aspects of graphs lend themselves to drawing. A variety of paper and pencil games require users to manipulate graphs toward a certain

aim. In Sprouts [10], for example, players follow simple rules to draw nodes and edges in an attempt to block the other player. In "Planarity," an interesting on-line graph game [11], players drag connected vertices around on the computer screen in an attempt to arrange the graph so that none of the edges overlap. These games successfully exploit the two dimensional nature of paper or a computer display, but many interesting games and tasks dealing with graphs are not as easy to represent on a planar surface. A computationally enhanced construction kit, on the other hand, offers a tactile and tangible way to interact with graph theory concepts that can augment the experience available with paper and pencil.

GRAPHMASTER

Hardware

Graphmaster is a computationally-enhanced construction kit made up of a collection of edges and a collection of nodes. Physically its nodes are small (approximately 10cm in diameter) plastic hemispheres with embedded ports where edges can connect. Edges are short lengths of electro-luminescent (EL) wire with connectors at their ends that snap magnetically into the nodes (see Figure 1). Nodes communicate with their neighbors over connected edges. Each edge can light up in blue or green, and uses capacitive sensing to detect when it is being touched. When an edge is connected between two nodes, microcontrollers in each node detect the new link so that Graphmaster "knows" about the connection. Depending on the program that Graphmaster is running (actually, the program runs in a distributed fashion in each of the nodes), the kit behaves differently.

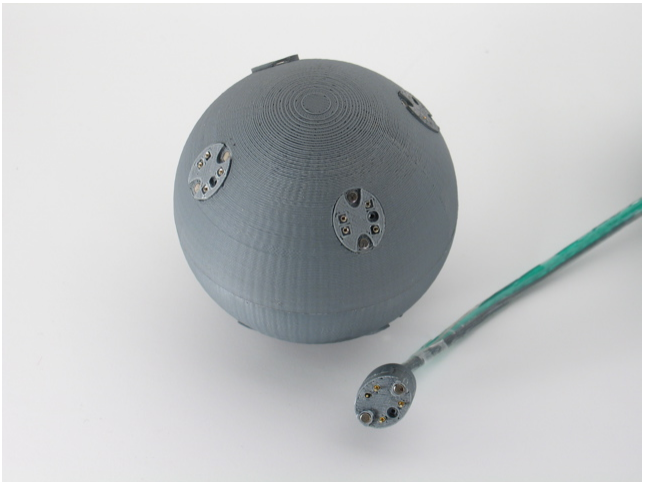

Figure 1. Each Graphmaster node is a plastic hemisphere with four ports for connecting edges. Each edge is a length of electroluminescent wire with a connector at each end.

As illustrated in Figure 2, each node contains a PIC 16765 microcontroller, a triac (solid-state relay) for powering the EL wire, and connections to 4 ports. Each port contains five conductors: one for bi-directional communication between any two nodes (e.g North); two for power and ground (globally shared), and a pair to distribute the high voltage signal that powers each EL wire on command.

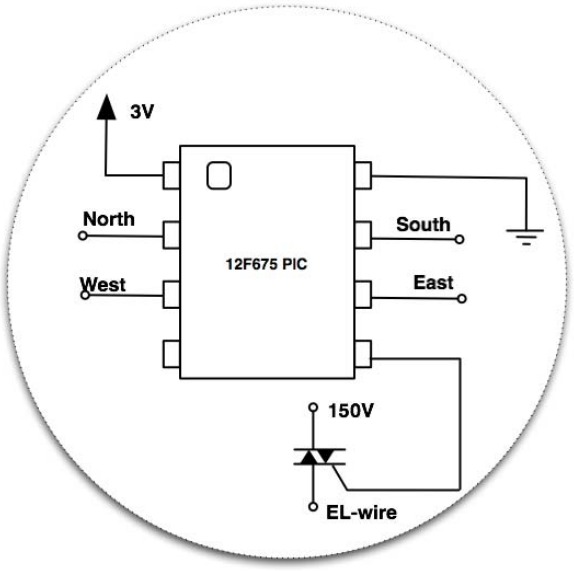

Figure 2. Circuit schematic of a single node.

Data Transfer

The nodes communicate using a homegrown return-to-zero pulse train: each bit is encoded as a unique pulse whose length depends on whether a one or a zero is being sent. Pulses are separated by a short period during which the line is low. When the line initially transitions from low to a high level, a listening node perceives this as the start of a new bit. The node then continuously examines the line until it returns to a low level. It then determines whether the pulse it just observed represents a one or a zero based on the length of the pulse. The length for a one is three times that for a zero, ensuring that the nodes can understand each other even if their timing sources differ by as much as 20%.

In its quiescent state, the communication line is high; thanks to weak pullup resistors, unconnected ports also remain high. To communicate, a node first checks to see that the line is high, then drives the line low to indicate its intent to communicate. As the other nodes might be polling other ports, it keeps the line low long enough, before sending message pulses. To minimize collisions, each node's unique ID is used to produce timing that affects when and how the node can assert its signal on the line.

Messages consist of four pieces of information (each piece is a byte):

the identification number of the sending node;

a sequence number for the message;

the message identifier; and

an argument.

Messages are lightweight yet generic enough to exchange high-level instructions, including new programming code.

Software

Graphmaster is a distributed construction kit; that is, there is no central processor that operates on constructions that users build. Rather, each node contains a microcontroller that runs a small byte code interpreter in EEPROM for the Graphmaster programming language. The kernel on the microcontroller keeps track of what other nodes are connected, and on which ports. Graphmaster programs are written and compiled to bytecode on a PC and then the same program is downloaded into every node. The PC can then be disconnected and Graphmaster works as a standalone system. The behavior of a Graphmaster construction is the result of the program running simultaneously (but asynchronously) on all the nodes.

Each node has a unique identity, represented in the language as a number. Links do not have identities but they can be described by the pair of nodes they connect.

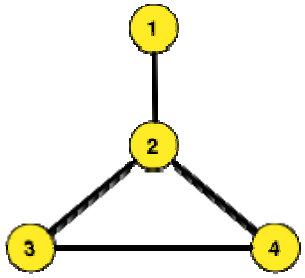

Figure 3

For example, if the kit is assembled to describe the simple graph in Figure, a program running in node 1 could compute:

node-neighbors ➔ [2]

whereas the same query in node 4 yields a different result:

node-neighbors ➔ [2 3]

A program running in a node can also refer to its "paths". This is the set of paths through the tree, beginning with neighboring nodes. The four port node maintains four paths: north-path, east-path, etc.

Each path is a list of node identifier numbers. If a path includes a cycle, the last node in the path repeats the first. For example, a program running in node 1 in figure 3 computes:

(Paths) ➔ (2 3 4 2) (2 4 3 2)

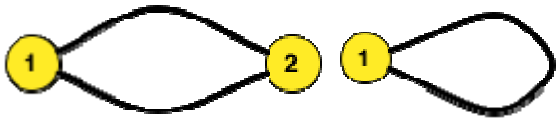

Figure 4

Graphmaster also allows loops, and the construction of multiple edges between two nodes (Figure 4).

A program can illuminate an edge, by calling the turn-on function with the desired port as an argument:

(turn-on north-edge)

Example: Detecting Cycles

Even at a relatively small scale, an interactive kit such as Graphmaster allows for an initial encounter with foundational and important ideas in graph theory. The presence of a cycle in a graph can be naturally derived by programming each vertex ask for its "reachable set", which we call "paths" in the Graphmaster language. Each of its neighbors is asked to identify both themselves and their own paths (not including the immediate neighbor who sent this message). If a vertex finds that it is within one or more of its own paths (and assuming that we disallow multiple edges between vertices), we know that this vertex is in a cycle within the graph. Likewise, if each vertex is equipped beforehand with the identities of all vertices in the graph (or the total number of vertices in the graph), then it can tell whether the graph is connected by seeing whether its "reachable set" includes all vertices. Should one vertex see that it is in a connected graph, all will; and if a graph is connected but without cycles, each vertex can likewise identify that it is in a graph known as a tree.

Here is a piece of a simple program that illuminates cycles in a graph the user builds (Figures 5 and 6).

```
(if (contains? north-path my-id)
   (turn-on north-edge)
   (turn-off north-edge))
```

The Graphmaster compiler on the PC produces the following byte code for the fragment above:

```
0x1f  //if
0x1a //  contains north-edge
0x07 // ID happens to be 7
0x02 // how many instructions to skip if condition not met
0x0a // turn on north edge
0x01 // jump over next instruction
0x0b // turn off north edge
```

Figure 5. Edge connectors snap magnetically and connect electrically to the ports on the nodes. One node contains a battery and supplies power to the others.

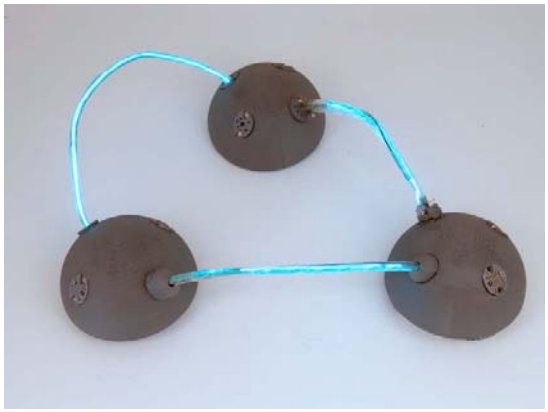

Figure 6. A cycle has been detected in the graph and Graphmaster lights up the cycle's edges.

DISCUSSION

More technological refinement is necessary before we can test the Graphmaster with users. Still, based on our prototype, we are optimistic about the utility of the kit. Even at a relatively small scale, it seems plausible that an interactive kit such as Graphmaster affords direct experience with the primitives of graph theory.

We illustrated Graphmaster with a simple algorithm to detect cycles. Yet another (standard) example involves graphs with Eulerian circuits (this is a path that traverses every edge in the graph exactly once, returning to the starting vertex; the notion is usually introduced through the famous Königsberg Bridge problem investigated by Leonhard Euler). As it happens, a graph has an Euler circuit if and only if each vertex has an even number of edges connected to it; this too is a plausible question to ask of a Graphmaster vertex (though in this case, most sample graphs do in fact allow for multiple edges between vertices; thus a Graphmaster vertex must be able to identify not just its neighboring vertices, but also the number of edges connecting it to each neighbor).

FUTURE WORK

We have thought of a number of enhancements that we plan to add to the Graphmaster project, both hardware and software. For hardware, we plan to add an output (LED or simple LCD display) on each node. This would provide an additional for users to debug their programs, and it might also be useful for programs to be able to highlight not only edges, but also nodes. We also plan to add touch sensing to the edges, a feature that we had implemented in an earlier prototype. Similarly, we intend to provide each node with a sensor input (touch or switch). With sensing on each node and edge as well as actuation, we expect that users will be able to write more interesting and interactive programs.

Somewhat more challenging will be to add continuous (analog) sensing to the edges. Many interesting problems can only be modeled with weights on the edges, for example modeling connection strengths, or flows, between nodes. This entails designing an analog sensor that can be elegantly embedded in the edge, and visually echoing the edge's weight. We might indicate the edge weight with the EL wire brightness, although this poses another set of hardware issues.

Regarding software, we plan to build a more comprehensive language, PC based programming environment, and the supporting bytecode interpereter. It might be useful, for example, to provide a mode where the graph kit is connected to the PC for design and debugging; the graph is modeled on the PC screen as well as physically, and the PC environment can watch and probe individual nodes. Once the user is satisfied with the program the kit would be untethered and work in standalone mode.

ACKNOWLEDGMENTS

This research was supported in part by the National Science Foundation under Grant ITR-0326054. We thank Ben Wojtyna, an undergraduate industrial design student who helped design and prototype the physical connectors and the EL wire edge hardware.

REFERENCES

[1] P. M. Higgins, *Nets, Puzzles, and Postmen*. Oxford: Oxford University Press, 2007.

[2] E. J. Kaufmann, *Nine Commentaries on Frank Lloyd Wright.* Cambridge, MA: MIT Press, 1990.

[3] A. A. DiSessa, *Changing Minds: Computers, Learning, and Literacy*. Cambridge, MA, USA: MIT Press, 2000.

[4] S. Papert, "Situating Constructionism," in *Constructionism*, I. Harel and S. Papert, Eds. Norwood, NJ: Ablex Publishing Company, 1991, pp. 1-11.

[5] N. Brosterman, *Inventing Kindergarten*. New York: Abrams, 1997.

[6] M. Resnick, F. Martin, R. Berg, R. Borovoy, V. Colella, K. Kramer, and B. Silverman, "Digital manipulatives: new toys to think with," in *SIGCHI conference on Human factors in computing systems*, Los Angeles, CA, 1998, pp. 281-287.

[7] E. Schweikardt and M. D. Gross, "Learning About Complexity with Modular Robots," in *DIGITEL 2008: The First IEEE International Workshop on Digital Game and Intelligent Toy Enhanced Learning*, Banff, Canada, 2008.

[8] M. Eisenberg, L. Buechley, and N. Elumeze, "Computation and Construction Kits: Toward the Next Generation of Tangible Building Media for Children," in *Proceedings of Cognition and Exploratory Learning in the Digital Age (CELDA)*, Lisbon, Portugal, 2004.

[9] V. LeClerc, A. Parkes, and H. Ishii, "Senspectra: A Computationally Augmented Physical Modeling Toolkit for Sensing and Visualization of Structural Strain," in *CHI 2007*, San Jose, CA, 2007.

[10] E. R. Berlekamp, J. H. Conway, and R. K. Guy, *Winning Ways for your Mathematical Plays* vol. 2. New York: Academic Press, 1982.

[11] J. Tantalo, "Planarity, http://www.planarity.net/," 2005.