TEACHING FUNDAMENTAL COMPUTER PROGRAMMING CONCEPTS TO MECHANICAL ENGINEERING STUDENTS USING PALPABLE INTERACTIVE VISUAL LEARNING AIDS

Kush Bubbar, and Yang Shi
Faculty of Engineering, University of Victoria, BC, Canada
Corresponding Author: kbubbar@uvic.ca

Abstract –*Pointers have long been the Achilles heel of mechanical engineering students attempting to master dynamic memory allocation in mechatronic applications. They are abstract and intangible, both opposing characteristics of a discipline based on the concrete (and often hands on) physical world. With this said, pointers are considered an important threshold concept opening the door to the implementation of complex microcontroller applications in our digitally connected world.*

One of the primary challenges in learning the application of pointers is that the programming syntax and the abstract memory management concepts are often taught simultaneously. The natural progression of learning is to first comprehend the concepts followed by the syntax. Further newer learning theories suggest a conceptual understanding can only result through abstraction of experiences using metaphorical linkages.

The following research body is focused on proposing a new strategy for teaching this complex concept using low cost physical props as a palpable interactive visual medium to provide the requisite experiences for concept abstraction. The learning aids are designed to enforce a strict process flow mimicking the invisible actions occurring internal to the microprocessor. Data is collected via questionnaires administered pre and post lecture delivery. Analysis of the results suggest moderate to high improvement in student comprehension of computer memory allocation concepts.

Keywords: programming, computer memory allocation, pointers, experiential learning, learning aids, tangible, palpable, teaching

1. INTRODUCTION

With the advent of low cost semiconductor technology combined with the growing popularity of digital systems, interest in the multidisciplinary field of mechatronics has observed a steady growth over the past decade. The University of Victoria has responded accordingly by offering a Mechatronics and Embedded Systems Option to its engineering undergraduates upon completion of a set of required courses for which MECH 458 – Mechatronics is included.

A core component of MECH 458 is the term project in which students design and implement C code to control an ATMEL™ embedded microprocessor interfacing with a series of sensors and actuators. These hardware components form an autonomous serial conveyor system designed to sort parts based on their composition as seen below in Fig. 1.

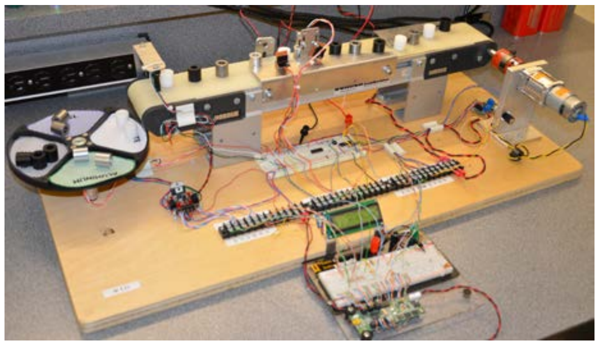
Fig. 1. Image Conveyor Sorting Project.

One of the fundamental challenges of this project is the system must be able to accommodate an unknown number of parts input onto the conveyor belt. The dynamic nature of this problem requires students to both understand the concepts surrounding Pointers along with their implementation in constructing Linked Lists.

In programming, Pointers are a fundamental threshold concept [1] used as a building block for more complex dynamic software patterns [2]. A misunderstanding of Pointers often leads to misconceptions with all constructs utilizing them [2]. It is thus imperative that students understand Pointers early on and understand them well.

The challenge lies in the fact Pointers and Linked Lists are both concepts requiring an abstract understanding of the invisible (and intangible) operations occurring inside computer memory [2]. In addition, the concepts surrounding Pointers and Linked Lists are often taught simultaneously alongside their implementation using coding syntax adding to further confusion.

In this research paper we seek to accomplish the following:

- Separating the teaching of the concepts surrounding Pointers, Linked Lists, and memory allocation from their coding syntax.
- Design and fabricate a palpable learning aid as a physical interactive visualization tool for representing memory in a microprocessor.
- Develop an experiential based lecture relating an example from the conveyor sorting design project with memory allocation methods through the palpable interactive visual learning aids.
- Teach the memory allocation concepts via lecture delivery using this methodology and solicit feedback from students on their learning experience.

2. BACKGROUND

It is no secret that novice students struggle when learning how to program. Several self-assessment studies have been conducted at institutions across the globe, each of which consistently revealed pointers and memory management as the most difficult concepts to master [2]–[4]. This result raises the question, why are pointers so difficult to grasp? The general consensus is related to the abstract nature of the invisible actions which occur "under the hood" in computer memory and their relationship to the coding constructs [2], [3]. The obvious follow up question is how can we adapt our teaching methodology to better support student learning? In searching for the answer to this question we first turn our attention to understanding the wider context of learning and teaching[1].

The majority of STEM courses are still believed to be taught using a transmission [6] style of teaching in which instructors view knowledge existing independent of those individuals who possess it [6], [7]. It is the belief of these traditionalists that learning occurs as the result of instructors *presenting* knowledge in a systematic order with learners ascending the role of knowledge *absorption*. Gibbs [7] labelled this model of learning "*disseminating knowledge*" as those followers believe individual study and practice play important roles in *obtaining* knowledge. In this model knowledge is thus *delivered* from the teacher to the student.

Newer learning theories contrast this traditional model by describing knowledge as an entity that must be *reconstructed* by the learners themselves [5], [6]. In this regard the environment and social context play a substantial role in student learning and are the basis for the Problem Based Learning (PBL) movement (i.e. learners need to question and challenge the information to create their own meaning) [6].

Fundamental to this learning model, the reconstruction of knowledge occurs as the result of the learner abstracting concepts and principles from their experiences [5]. In this regard metaphors play an instrumental role in describing complex concepts and processes through their relationships with prior experiences [5]. Incidentally visual learning aids may be considered a medium to present metaphors and are used extensively in programming textbooks.

Interactive visual learning aids can play an even bigger role as they allow learners to interact with the metaphors and explore intricate relationships describing various edge cases, helping to further solidify conceptual understanding. The standard delivery method of interactive visual aids are typically application or web-based. The Codewitz (www.codewitz.net) project is an example of an online inventory of interactive visual programming aids [3].

Web-based interactive visual learning aids are beneficial for learners heavily inclined to interact with computer based mediums but what about the demographic who prefer a hands on tangible approach? Mechanical engineering students fall into this category and are the basis for exploring the development of a *palpable interactive visual learning aid*. To the extent of the author's knowledge, this further sub class of metaphorical learning aids has not been detailed in literature within the context of teaching computer programming.

2.1 Palpable Interactive Visual Learning Aids

In developing the palpable interactive visual learning aid (also labelled as the *physical memory board*) the first critical step is identifying the design requirements. These originate from the conceptual rules governing the processes to be taught. In the context of memory management in microprocessors a few basic conceptual rules are considered:

1. Data cannot be stored in unallocated memory.
2. The allocation of memory defines the type of data than can be stored.
3. Linked lists can be created by saving the address of the next allocated memory block.

[1] An excellent overview of learning and teaching in the context of programming can be found in [5].

4. The start (HEAD) and end (TAIL) addresses of the linked list must be known in order to traverse the list.

From these design requirements a palpable interactive visual learning aid was constructed. Figure 2 is a cropped picture of the memory board fabricated out of plywood using a two axis CNC laser cutter. The board was designed with slotted features for placement of memory cards. Within the memory cards themselves are slotted features for data cards and hole features for plugs representing memory links.

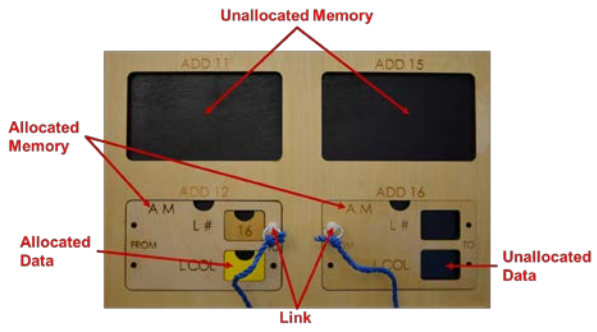

Fig. 2. Cropped Image of a Palpable Memory Board.

In terms of the conceptual rules, the large black slots on the memory board represent unallocated memory. Placing a memory card into one of these large slots embodies the act of allocating memory (AM) at this particular address (ADD). The small black cutouts within the memory cards symbolize unallocated data. The bottom cutout labelled LCOL is for stored data (colour) while the top cutout labelled L# is for an address pointing to the next allocated memory location in the linked list. Finally, the plugs with rope attached represent a physical path to traverse between linked memory addresses. There are two sets of holes in each memory card, each able to represent a different linked list for the same data set in memory. Both linked lists are easily distinguished through use of a rope colour scheme.

We note an important outcome of using this palpable learning aid; data cannot be stored unless memory is first allocated. On the board, this means we cannot place a coloured data card into a black unallocated memory slot as the data card will literally fall out. This is an excellent example of how a palpable interactive visualization tool can be used to create a metaphorical linkage between a simple perceivable experience and the abstraction of a complex concept.

2.2 Learning Outcomes

The intention of this research is to explore a new methodology of teaching basic memory management concepts to MECH 458 students as applied to their conveyor sorting project. As such four important learning outcomes are identified:

1. Understanding the abstract relationship between the allocation of memory and the storage of data.
2. Comprehending how data in memory are connected through dynamic linkages.
3. Understanding the detailed workflow in memory for each of the cases below.
4. Linking the workflow in memory to the observed actions on the physical conveyor system.

2.3 Distinct Cases Considered

Within the lecture three distinct cases are considered; the first two relate to data structures and the third focuses on data sorting.

2.3.1 Automatic Memory Allocation. Automatic memory allocation represents the situation where memory is allocated on the memory stack at compile time. An example data structure is an array which is a “consecutive group of memory locations that all have the same name and the same type.” [8]. Arrays have a fixed size and thus can only hold fixed amount of data, but their implementation requires very little overhead.

Novice learners often pick up the concepts related to utilizing arrays fairly quickly. Initiating the lesson with this case servers to familiarize students with the palpable learning aids in the context of something they already understand well. It also establishes a baseline for comparative purposes.

2.3.2 Dynamic Memory Allocation. In dynamic memory allocation, memory is allocated at run-time (as we need it) on the memory heap. An example data structure is a linked list which is a “collection of self-referential data structures connected by links [8]. Linked lists are only limited by the amount of available unallocated memory on the heap. Due to the complex nature of dynamic memory allocation, this case is presented after the automatic memory allocation example is discussed in detail.

2.3.3 Data Sorting Using Linked Lists. Sorting is the act of organizing data in a particular order accomplished through the execution of an algorithm. Several algorithms exist with various levels of complexity and computational overhead. The intention of this section is not to teach the sorting algorithm but to enforce how the sorted data is stored and managed in memory within linked list data structures. It is of interest to note, sorting algorithms could very well be taught using the physical memory board but is outside the scope of this research paper.

3. METHODS

The following section describes the methodology and structure of the lecture delivered to address the learning outcomes detailed in Section 2.2.

3.1 Lecture Set Up

Due to the large class size (110 students) and the requirement for students to view both the animated conveyor scenario at the same time as the physical memory board, a lecture hall with a dual projector was needed. One projector was used to view the animated PowerPoint™ presentation representing the conveyor system while the second projector displayed a live video capture of the physical memory board. This left the blackboard available for noting important points and discussing queries as they arose in the lecture.

3.2 Pre-lecture Assessment

An optional anonymous pre-lecture questionnaire was delivered to class participants via a web-based form a day prior to the lecture. The questionnaire consisted of three self-assessment questions (two dichotomous Yes/No and one 5-point Likert Scale) followed by a single skill testing question. This established a reference for class level competency in computer memory allocation.

3.3 Description of Teaching Methods

Following the pre-lecture assessment, a detailed description of the two conceptual representations used in this lecture were presented to the class.

3.3.1 Palpable Interactive Visual Learning Aid. The physical memory board was demonstrated to the class indicating all the features and their relationships to the conceptual rules detailed in Section 2.1.

3.3.2 Conveyor Sorting Example. The physical actions of the mechanical conveyor system was presented using the animated PowerPoint™ slide shown in Fig 3. The process flow of three coloured parts were examined in detail as they moved past a pair sensors in series. The optical sensor detected the presence of a part in the system while the colour sensor detected the part's colour.

This animation was repeated to teach all three cases detailed in Section 2.3 using a knowledge scaffolding methodology [1].

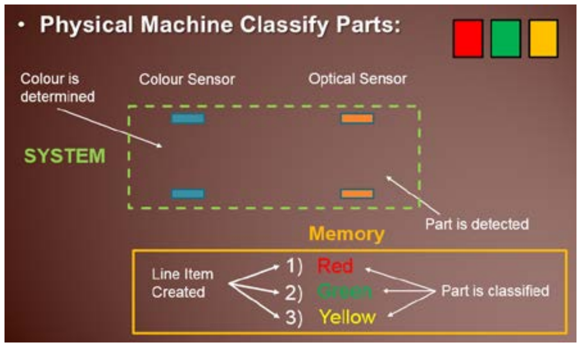

Fig. 3. Screen Capture of Animated Conveyor System.

3.4 Data Structures with Examples

The lecture proceeded by first teaching the conceptual rules describing each of the two data structure types. This was followed by a step by step walkthrough of the conveyor animation in conjunction with execution of the required steps in memory. The required steps in memory were determined by students through instructor led classroom inquiry. The following subsections describe each case in detail.

3.4.1 Example with Arrays. By far the simplest case considered utilized arrays to store the sensor data. For array declarations, a three element array was allocated in memory at compile time meaning three memory cards were placed onto the memory board in consecutive memory addresses prior to even considering the conveyor animation. This is once again an important metaphorical linkage.

The conveyor sorting animation followed and the colour data was stored in each of the three array elements through placement of data cards. The inflexible size limitations became blatantly obvious along with the simplicity of data storage in arrays. The physical memory board with data stored can be viewed in Fig. 4 below.

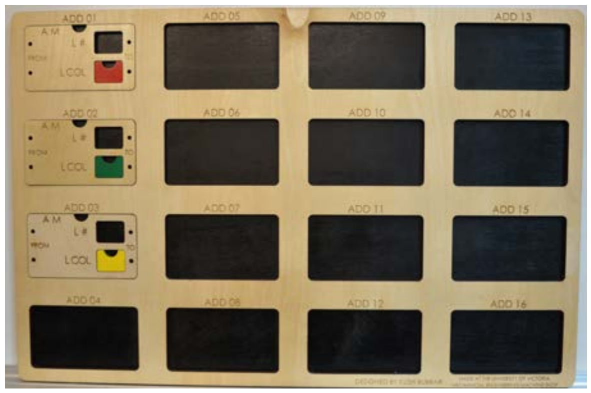
Fig. 4. Image of a 3-element Array on the Physical Memory Board.

3.4.2 Example with Linked Lists. For the same conveyor sorting scenario, the process flow with linked lists is much more complicated. We begin by allocating the HEAD and TAIL memory cards since they are assigned at compile time.

The conveyor animation proceeds. As each part passes the optical sensor, a part is detected and space is dynamically allocated in memory to store the impending colour data. This requires the execution of a series of steps.

- Memory is allocated at a random address (ADD)
- A linkage is established between HEAD, the new allocated memory, and TAIL
- The L# slot in HEAD is filled with the address of the new allocated memory
- The L# slot in the new allocated memory is filled with the address of TAIL
- Rope is connected between the OUT plug of HEAD into the IN plug of the new allocated memory
- Rope is connected between the OUT plug of the new allocated memory into the IN plug of TAIL

Once the part passes through the Colour Sensor, the part's colour is measured and assigned in the LCOL slot of the new allocated memory. The animation continues with the next part entering the system. Once again new memory is allocated at another random address (ADD). Linkages are broken between the previous allocated memory (AM) and TAIL and the new memory is "slotted" into this gap. New linkages are created forming a continuous chain. The final configuration of the physical memory board is observed in Fig. 5.

These physical steps serve to connect the learner with a tangible experience which can be abstracted into a conceptual understanding of computer dynamic memory allocation.

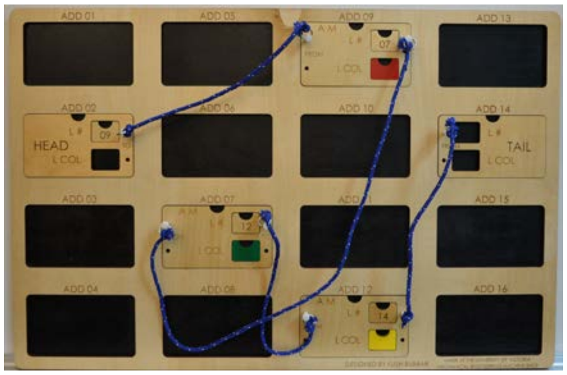

Fig. 5. Image of a 3-element Linked List on the Physical Memory Board.

3.4.3 Example with Data Sorting Using Linked Lists. Finally with a conceptual understanding of dynamic memory allocation established, the last example serves to showcase how multiple linked lists can be used in conjunction to enforce multiple sorts on the same data set.

The same conveyor animation is demonstrated once again but with the caveat that two linked lists are managed simultaneously representing different rules for sorting the same data. The first is an operational sort based on the order the parts enter the system (resulting in the same linked list as generated in Section 3.4.2). The second sort is based on a colour schema listing priority from highest to lowest as Green, Blue, Yellow, and Red. The same parts are sent through the conveyor system and during each step both linked list chains are managed according to their associated sorting rules.

Through manual manipulation of the memory board, it becomes evidently clear the number of operations to maintain the linked list chain is directly proportional to the number of linked lists; once again creating an important metaphorical linkage. The final configuration of the physical memory board is observed in Fig. 6.

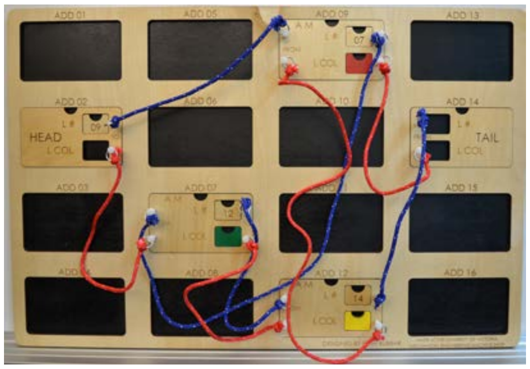

Fig. 6. Image of a 3-element Linked List on the Physical Memory Board.

3.5 Post-lecture Assessment

At the end of the lecture another optional anonymous questionnaire is delivered to the class participants through a web-based form. The motivation for this questionnaire is to assess student learning through comparison of both questionnaires.

This questionnaire consisted of the same self-assessment and skill testing questions in addition to a series of 5-point Likert Scale, and short answer questions on student experience.

4. RESULTS AND DISCUSSION

Despite the ambitious nature of the lecture it was easily completed within the allocated 85 minute time slot. The results of the lecture were determined through comparison of the pre and post lecture questionnaires.

4.1 Pre-lecture Assessment Results

Seventy-two students partook in the pre-lecture questionnaire. Results detailed in Fig 7. clearly indicate the vast majority of the students who completed the survey self-assessed as not having a clear understanding of computer memory allocation nor understanding the differences between automatic and dynamic memory allocation.

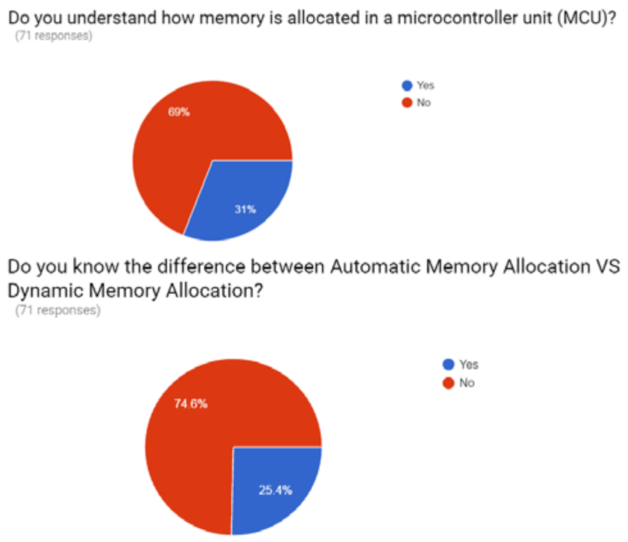

Fig. 7. Pre-Lecture Self-Assessment on:
Top: Memory Allocation in MCUs
Bottom: Difference between Automatic and Dynamic Memory Allocation.

In addition, even fewer students (13 out of 71 or 18.3%) responded correctly to the skill testing question exhibited in Fig. 8 reinforcing literature conclusions in which students are often over confident with their competency levels resulting in overrating their capabilities in self-assessments [9].

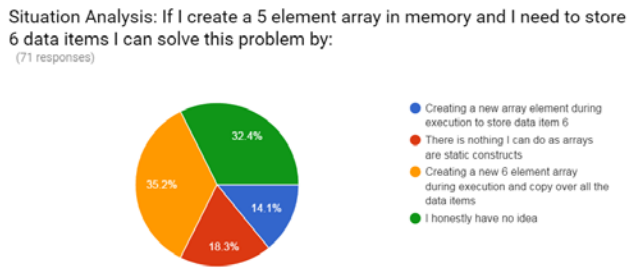

Fig. 8. Pre-Lecture Skill Testing Question on Computer Memory Allocation
(18.3% Respondents Answered Correctly).

4.2 Post-lecture Assessment Results

For comparative purposes, forty-seven students partook in the post-lecture questionnaire delivered at the end of class. Results clearly indicated a dramatic shift in student self-assessment of their competency in computer memory allocation concepts as viewed in Fig 9.

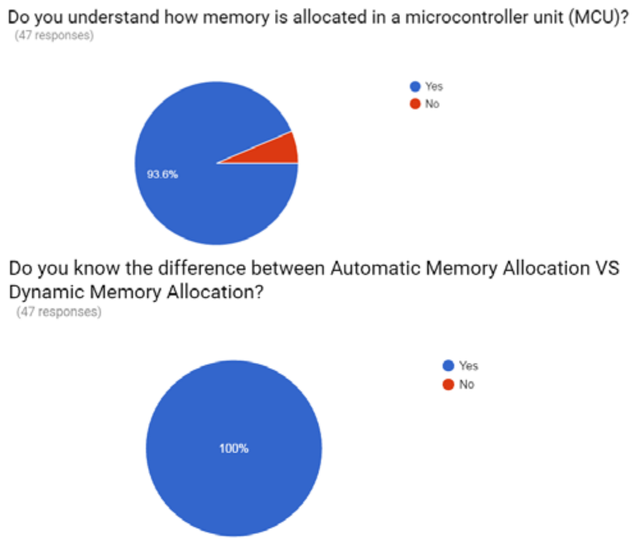

Fig. 9. Post-Lecture Self-Assessment on:
Top: Memory Allocation in MCUs
Bottom: Difference between Automatic and Dynamic Memory Allocation.

Correct answers to the skill testing question increased dramatically to 38 out of 47 (80.9%) as presented in Fig. 10. Again in a similar trend to the pre-lecture assessment, students overestimated their competency in the subject matter.

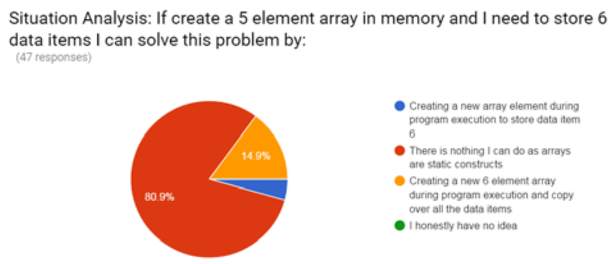

Fig. 10. Post-Lecture Skill Testing Question on Computer Memory Allocation (80.9% Respondents Answered Correctly).

Finally students assessed their experience with the lecture through answering questions on a 5-point Likert Scale with 5 and 1 representing Extremely Helpful and Not Helpful at All respectively. The average value from this measurement was 4.34 out of 5 suggesting students believed the palpable interactive visual learning aid promoted their understanding of computer memory allocation concepts.

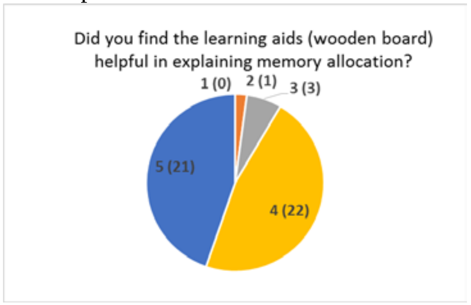

Fig. 11. Post-lecture Student Assessment on Their Perceived Value of the Physical Memory Board in Explaining Memory Allocation Concepts
1 – Not at All, 5 – Extremely Helpful
Value in brackets () represents the number of respondents

5. CONCLUSIONS AND RECOMMENDATIONS

Upon conclusion of the lecture a number of students stayed back to pose questions and provide feedback. The major theme of these questions centred on access to the memory board outside of class for their own personal practice. An important conclusion was realized: for students to truly obtain sufficient experience to allow for concept abstraction, they need direct interaction with the physical memory board working at their own pace.

The major constraint with addressing this challenge lies with the large class size considered. If a lecture is facilitated in which students work in groups of 4 to interact with the memory board; for a class size of 110 students, 28 boards would need to be fabricated. The logistics surrounding this endeavour are unreasonable.

Consequently a future recommendation is to consider converting this lecture into a short experiential based laboratory activity, executed by a small subset of students at different time periods. This would reduce the inventory demands for the physical memory boards.

Overall the results from utilization of the palpable interactive visual learning aid were positive in the context of teaching memory allocation in computer programming applications for the MECH 458 conveyor sorting project.

The authors of this research study are strongly considering releasing the design documentation to replicate the physical memory board via an open source license for use at academic institutions.

Acknowledgements

I would like to personally thank Rodney Katz and the University of Victoria’s Engineering Machine Shop for always finding a way to fabricate my crazy ideas and Professor Yang Shi for offering the opportunity to trial this unique teaching method in his course. I would also like to thank Joe Parsons at the University of Victoria’s Learning and Teaching Centre for his guidance in assessing student competencies.

References

[1] W. McKeachie and M. Svinicki, *McKeachie’s teaching tips*. Cengage Learning, 2013.
[2] I. Milne and G. Rowe, “Difficulties in Learning and Teaching Programming—Views of Students and Tutors,” *Educ. Inf. Technol.*, vol. 7, no. 1, pp. 55–66, Mar. 2002.
[3] E. Lahtinen, K. Ala-Mutka, and H.-M. Järvinen, “A study of the difficulties of novice programmers,” in *ACM SIGCSE Bulletin*, 2005, vol. 37, pp. 14–18.
[4] P. H. Tan, C. Y. Ting, and S. W. Ling, “Learning Difficulties in Programming Courses: Undergraduates’ Perspective and Perception,” in *International Conference on Computer Technology and Development, 2009. ICCTD ’09*, 2009, vol. 1, pp. 42–46.
[5] L. P. Baldwin and J. Kuljis, “Learning programming using program visualization techniques,” in *Proceedings of the 34th Annual Hawaii International Conference on System Sciences, 2001*, 2001, p. 8 pp.–.
[6] D. D. Pratt, “Good Teaching: One Size Fits All?,” *New Dir. Adult Contin. Educ.*, vol. 2002, no. 93, pp. 5–16, Mar. 2002.
[7] G. Gibbs, *Improving Student Learning through Assessment and Evaluation*. ERIC, 1995.
[8] P. J. Deitel and H. M. Deitel, *C++ how to program*. PearsonPrentice Hall, 2008.
[9] D. Dunning, C. Heath, and J. M. Suls, “Flawed self-assessment implications for health, education, and the workplace,” *Psychol. Sci. Public Interest*, vol. 5, no. 3, pp. 69–106, 2004.