Topical Tapestry: Weaving Threads of Parallel Programming, Computer Graphics, and Artificial Intelligence into Undergraduate CS Courses

James Wolfer
Computer Science
Indiana University South Bend
South Bend, IN, USA

***Abstract*— Traditionally, topics such as parallel computing, computer graphics, and artificial intelligence have been taught as stand-alone courses in the computing curriculum. Often these are elective courses, limiting the material to the subset of students choosing to take the course. Recently there has been movement to distribute topics across the curriculum in order to ensure that all graduates have been exposed to concepts such as parallel computing. This work describes one approach to threading topics from AI, graphics, and parallel processing into coursework starting early in a student's program without substantially altering the basic course objectives. This, in turn, allows the student to sample topics early enough to make informed decisions when selecting electives.**

Keywords—pedagogy, artificial intelligence, computer graphics, parallel computing, instructional approaches.

I. INTRODUCTION

In anticipation of the forthcoming ACM/IEEE Computer Science Curricula 2013 [1] coupled with a resurgence of project driven instruction there is some movement within the computing community to distribute topical instruction across the curriculum. For example, Sheldon and Turbak [2] describe distributing aspects of computing topics ranging from computational theory to artificial intelligence. They also discuss challenges and tradeoffs involved such as identifying prerequisites and the necessity for faculty collaboration.

Given the NSF/IEEE-TCPP Parallel and Distributed Computing [3] guidelines, the move to distribute parallel computing concepts may be the most mature at this time. For example, Minaie and Sanati-Mehrizy [4] review a variety of programs, both domestic and international, in terms of their parallel computing curriculum. They report that in China there are two primary approaches, either a dedicated parallel processing course, or integrating parallel concepts into the Computer Organization, Architecture, Operating Systems, and/or Embedded Systems courses among others. Their review of seventeen universities in the United States identified four integration models: an independent undergraduate course, an independent graduate course, concept integration into existing courses, or a combination of the three previous approaches. In all cases the trend is to move topics out of the elective-only curriculum.

Finally, Danner and Newhall [5] describe a systematic approach to ensuring that all their CS students are exposed to "parallel thinking" during the course of their studies. The approach includes a required introductory course that includes parallel concepts in anticipation of future coursework. The balance of the parallel topics are distributed among various more advanced courses such as Compilers, Operating Systems, and Computer Graphics, among others.

While all of these approaches seem to be effective in the distribution of parallel concepts across the curriculum, all except [2] ignore the potential of distributing topics from other specialties. This work describes one effort to identify three specific courses and to systematically integrate concepts from each of them into a subset of classes that all CS students must take. This serves two purposes. First it exposes the students to the topics, even if only in passing for some cases. Second, it serves as a "sampler" to give the students a taste of each topic to help inform their future course selection. The courses and topics included in this effort are Computer Graphics, Parallel Computing, and Artificial Intelligence.

In the Computer Science program at our institution the Computer Graphics, Parallel Computing, and Artificial Intelligence classes are entirely elective, upper-level computing classes. While this means that students in the classes are there by choice, it also implies that many students will not have encountered some of these topics prior to graduation. Furthermore, those who do take these classes often arrive with no significant concept of the constituents of the respective class. Our approach to this problem is to, as seamlessly as possible, introduce concepts from Computer Graphics, Parallel Computing, and Artificial Intelligence early in a student's program, then allowing the concepts to re-emerge in subsequent courses.

978-1-4799-3190-3/14/$31.00 ©2014 IEEE 3-5 April 2014, Military Museum and Cultural Center, Harbiye, Istanbul, Turkey
2014 IEEE Global Engineering Education Conference (EDUCON)

II. FOUNDATION COURSES

From a pedagogical perspective being exposed to concepts multiple times, and in multiple contexts, is an important part of information transfer [6,7]. Additional benefits include potential interest in the elective classes as well potential recruits for undergraduate research programs.

That having been said, when classes are structured topically, there are constraints to weaving external topics into the class. Specifically, the following principles are adopted:

- Any embedding must not be gratuitous. There must be a natural fit to the material being covered in the host class.
- Any topical embedding must not take excessive time to explain and illustrate.
- Any topical embedding should provide topical insight for future elective classes.

With these principles in mind, the balance of this work provides examples of this embedding in both early and advanced CS courses.

First, it should be noted that, for all the early CS classes described here, a ten minute introduction to the instructor, including research interests and recent published results is presented during the introductory class session. This is typically done in a lightning-talk format and serves to connect students with the instructor.

The first opportunity for introducing concepts from the three classes featured here is CS1, the introductory programming class. This class is taught using C++ for the programming language. While prior exposure to programming is encouraged, it is not required for admission to the course. Therefore this class must start with an overview of basic computer concepts and proceed to develop algorithms and programs at a beginning level. Two specific classroom examples and corresponding assignments serve to subtly, and seamlessly, weave a pre-AI thread into the course. The first is a simple simulation of Eddington's Monkeys [8], based on the proposition that monkeys typing randomly will eventually create all the works of Shakespeare. For the CS1 student this introduces the concept of characters, their underlying numeric encoding, the modulus operator (to constrain them to the alphabet), and random number generators. It also introduces the concept of simulation using the computer. From a "weaving AI" perspective it allows mention of Natural Language Processing and Computational Linguistics. The second assignment, also involving text, has students writing code to count the number of each nucleotide in the DNA sequence for, recently, the H1N1 virus, using an integer array to contain the respective counts. This allows students to learn array indexing in a familiar context, having already worked with character simulation. It also invites a short discussion of other applications of more sophisticated NLP and textual analysis such as the IBM Watson project [9].

The CS2 course expands both the student's knowledge and experience with programming and offers incrementally more latitude for interjecting relevant samples from AI, Graphics, and Parallel Computing. Currently CS2 both expands syntactic knowledge and introduces elementary data structures. For example, multi-dimensional arrays and linked lists are introduced. Other topics include the use of pointers, C++ structs and classes, function and operator overloading, templates, as well as best practice topics such as conditional compilation, error trapping, and separate compilation of program components.

Since students are gaining maturity, the topics from CS1 are expanded in CS2. For example, the "Eddington's Monkey" program is expanded to create bigram and trigram correlation matrices from large bodies of English (in our case), which are then used to inform the "monkey" simulator – increasing the probability that they will produce letter sequences consistent with the language. This makes an excellent segue for discussing additional aspects of NLP, and to introduce tools such as the Google N-gram Viewer [10] as a way of tracking concept and sentiment in the literature over the span of decades.

Computer Graphics illustrations are also a natural fit in a discussion of multidimensional arrays as well as those for dynamic memory allocation. During the introduction of multi-dimensional arrays the concept of a gray-scale pixel is introduced as a character element in a 2-D array. Various C++ functions are developed to populate the array of pixels forming various simple geometric forms. To avoid the complexity of user-interface and graphics library development at this early stage in their experience these arrays are wrapped in appropriate header information and exported in PGM format to be displayed by an independent display program. More complex images, such as the Mona Lisa, illustrate the association of pixel value and image intensity as displayed on the screen. Practice problems using simple image processing, such as local averaging, is sometimes introduced as a means of teaching elementary array manipulation. Finally, images are used as an example data element to be dynamically allocated when developing the concepts and implementation of pointers, dynamic memory allocation, and linked lists

While CS1 and CS2 provide opportunities for touching on AI and computer graphics, parallel processing is barely mentioned. It is in the context of Computer Organization that Parallel Processing gets it's undergraduate debut. The Computer Organization course introduces basic computer structures at the logic level, such as full- and half-adders, multiplexors, flip-flops, and decoders. Students are then exposed to a significant assembly language project culminating in the development of a simulated CPU with embedded robot control instructions. The students then write programs using their own CPU to actually control robots. Since the robots are programmed to respond to sensors and

react to their environment, it is an excellent introduction to some of the pragmatic aspects of Artificial Intelligence.

There is also the first serious introduction to parallel processing in the Computer Organization class. Both in terms of the basic architectural aspects as illustrated with Flynn's Taxonomy, but also an opportunity to introduce contemporary parallel software development tools. Specifically, to tie the discussion to equipment that the students may already possess, we include an introduction to OpenMP [11] for shared-memory machines represented by our current multi-core CPUs, and a short introduction to Graphics Processing Units (GPU) illustrated by a survey of both Nvidia's CUDA [12] language and the vendor-neutral OpenCL [13] programming language. Since the GPU is intrinsically a graphics element there is mention of computer graphics in this class, but it is not particularly significant.

Finally, concepts from all three areas are touched upon in the Operating Systems course. This class is positioned late in the overall program to act as a captstone course encapsulating much of the knowledge and experience gained during the previous three or so years. The class covers classical operating system concepts such as process management, scheduling, interprocess communication, memory and file management, security, and device handling. Parallel processing is a natural fit, including discussions of multi-core threading, OpenMP, and the use of the GPU at the operating system level. Artificial Intelligence is considered in two specific contexts, scheduling where genetic optimization is discussed and security where invasion detection techniques such as Artificial Immune Systems are introduced. Finally, computer graphics is mentioned (but not emphasized) in the context of interactive devices and the considerations necessary for creating drivers for them.

III. AI, Graphics, and Parallel Programming Courses

The courses described in Section I are particularly important since they represent a sequence that all students in the program must take. This ensures that all of the students get some exposure to the relevant material even if they do not elect to enroll in the AI, Graphics, or Parallel Computing courses. That having been said, to weave a tapestry the individual threads must be entangled to create the whole. This section describes interconnecting the Parallel, Graphics, and AI courses.

The Computer Graphics course is a fairly typical graphics course embedded in a CS program. Beginning with the concept of raster and pixels the class moves quickly to 3D, including volumetric imagery, 3D modeling, transformation, and rendering. The course includes a significant project component, including formal proposals and final presentations. Since the GPU is central to modern transformation and shading, an introduction to parallel concepts is intrinsic to the graphics course. A short diversion into general-purpose GPU programming is included since computer graphics and image processing have a largely intersecting knowledge base.

Likewise, graphics is a good fit for some portions of the parallel computing class. The Parallel Processing course covers a wide range of architectures and algorithms, concentrating on the tools and implementation of the later. Software tools include Message Passing Interface (MPI) [14] for distributed clusters, OpenMP and threading for multi-core CPU's, and Nvidia CUDA and OpenCL for programming the GPU. Graphics illustrations, such as dissecting large images, such as mammograms, and distributing their processing provide a very visual indication of success or failure. A quick glance will indicate whether the resulting image is reconstructed, or scrambled! AI is introduced with the parallelization of paradigms such as Pulse-Coupled Neural Networks, a biologically inspired model for computer vision and image preprocessing.

Finally, the Artificial Intelligence class offers opportunity to integrate concepts from both graphics and parallel processing. The AI class is a hybrid of a discipline survey and senior/graduate project-based seminar. Topics include search, machine learning, computer vision, decision support, knowledge representation, neural networks, genetic algorithms, information theory, and natural language processing. In addition to lectures and assignments, students are required to propose and implement a formal project and present the results of their investigation as an integral part of the class.

Computer graphics is a natural fit both in the computer vision component of the AI class, as well as a discussion of Genetic Programming for artistic expression, natural language processing and image retrieval, and a serious discussion of intelligent image segmentation algorithms. Additional discussion in the context of biometric/biomedical applications of AI such as mammogram analysis is also presented, and projects encouraged.

While there would, on the surface, seem to be less overt opportunity for a discussion of parallel processing in AI, in fact parallel concepts form an important topic for discussion. Since much of what we currently believe about intelligence stems from massively parallel biological systems, such as the human brain, there is opportunity to explore parallel topics at a fundamental level. This includes alternate information encodings, such as neural spike intervals, as well as parallel control ranging from that in animal ethology to robotic applications.

IV. SUMMARY AND CONCLUSION

The thread of topics described here has been implemented over the course of several years. While a formal assessment of the impact of this approach has not been attempted, several anecdotal observations can be made. First, the topics included either for discussion or as illustrations or assignments from alternate topics have a minimal time impact on the respective classes. Assignments must be explained in any case, so why not explain with topics of future interest where appropriate? Secondly, several positive, non-classroom, aspects have emerged. These include students becoming interested and ultimately pursuing both undergraduate and graduate research in areas including AI for medical imaging, bio-inspired computing for mammogram analysis, and GPU programming to accelerate such processing. These residual benefits include undergraduate research summer fellowships and several published articles. Finally, as a bonus of this approach, many illustrations developed for the CS classes can migrate to the computer literacy class, enriching the experience of the general education student.

In conclusion, we believe that we have formed an approach to distributing topics that preserves the depth afforded by individual classes while distributing foundational concepts across a subset of the curriculum, thus weaving a fabric for future studies.

REFERENCES

[1] ACM/IEEE-CS Joint Taskforce, "Computer Science Curricula 2013 Final Report 0.9, Pre-release version", *http://cs2013.org*, Oct, 2013

[2] Sheldon, M. and Turbak,F., "An Aspect-Oriented Approach to the Undergraduate Programming Language Curriculum", *SIGPLAN Programming Language Curriculum Workshop*, May, 2008.

[3] Prasad S. et al., "NSF/IEEE-TCPP Curriculum Initative on Parallel and Distributed Computing – Core Topics for Undergraduates", *http://www.cs.gsu.edu/~tcpp/curriculum/*, 2012.

[4] Minaie, A. and Sanati-Mehrizy,R., "Incorporating Parallel Computing in the Undergraduate Computer Science Curriculum", *Proceedings 2009 ASEE Annual Conference and Exposition*, 2009.

[5] Danner, A. and Newhall, T., "Integrating Parallel and Distributed Computing Topics into an Undergraduate CS Curriculum", *Proceedings Workshop on Parallel and Distributed Computing Education (EduPar-13)*, 2013.

[6] Mastascusa, E.j., Snyder, W. J., and Hoyt, B. S.,*Effective Instruction for STEM Disciplines: From Learning Theory to College Teaching*, Jossey-Bass, 2011.

[7] Ambrose, S. A. et al.,*How Learning Works: Seven Research-Based Principles for Smart Teaching*, Jossey-Bass, 2010.

[8] Bennett, W. R.,*Scientific and Engineering Problem-solving with the Computer*, Prentice Hall, 1976.

[9] IBM, "Watson Project", http://www.ibm.com/watson

[10] Google, "Ngram Viewer", http://books.google.com/ngrams

[11] OpenMp, http://openmp.org

[12] Nvidia,"CUDA", http://www.nvidia.com/object/cuda_home_new.html

[13] Khronos, "OpenCL", http://www.khronos.org/opencl

[14] OpenMPI, http://www.open-mpi.org