Integrating Games and Machine Learning in the Undergraduate Computer Science Classroom

Scott A. Wallace
Washington State University Vancouver
School of Eng. and Computer Science
Vancouver, Washington 98616
1-360-546-9112
wallaces@vancouver.wsu.edu

Ingrid Russell
University of Hartford
Department of Computer Science
West Hartford, Connecticut 06117
1-860-768-4191
irussell@hartford.edu

Zdravko Markov
Central Connecticut State University
Department of Computer Science
New Britain, Connecticut 06050
1-860-832-2711
markovz@ccsu.edu

ABSTRACT

A student will be more likely motivated to pursue a field of study if they encounter relevant and interesting challenges early in their studies. The authors are PIs on two NSF funded course curriculum development projects (CCLI). Each project seeks to provide compelling curricular modules for use in the Computer Science classroom starting as soon as CS 1. In this paper, we describe one curriculum module which is the synergistic result of these two projects. This module provides a series of challenges for undergraduate students by using a game environment to teach machine learning and classic Artificial Intelligence concepts.

Categories and Subject Descriptors

K.3.2 [**Computers and Education**]: Computer and Information Science Education – *computer science education, curriculum*

General Terms

Design, Experimentation, Theory

Keywords

Artificial Intelligence, Machine Learning, Computer Games, CS1, CS2

1. INTRODUCTION

In Dale Carnegie's motivational classic *How to Win Friends & Influence People* [4], the reader is urged to "throw down a challenge". Noting the work of behavioral scientist Frederic Herzberg, he observes "The one major factor that motivated people was the work itself. If the work was exciting and interesting, the worker looked forward to doing it and was motivated to do a good job." Similarly, and along with many other educators [2], we believe that students will be more engaged in their studies and more likely to continue studying computer science if they find their course projects relevant, interesting and appropriately challenging.

GDCSE'08, February, 2008, Miami, Florida, USA.

Creating curriculum that students find relevant and interesting is a time consuming creative task. Unfortunately, for many instructors, the creative process is often cut short due to a host of competing obligations. The net result is that many programming exercises, especially those provided to lower division students, often seem contrived and fail to illustrate the challenges faced by professional computer scientists. Thus, students may get a skewed perspective of the discipline and become disinterested before the full range of problem solving possibilities is exposed by their course work.

We seek to provide instructors a core set of curricular modules that bring interesting and challenging problems into traditional computer science courses. Robotics and games are just two areas with demonstrable attraction for high-school students and non-majors. We believe that this attraction will extend into the undergraduate classroom when traditional computer science course objectives are taught using artificial intelligence or computer games as a vehicle.

In this paper, we describe two approaches to teaching computer science goals: one using a machine-learning theme; the other a computer game based theme. We then describe one curricular module currently under development that integrates both approaches to create a sequence of game-based student projects to teach core topics from Artificial Intelligence.

2. A MACHINE-LEARNING THEME

An introductory Artificial Intelligence (AI) course provides students with basic knowledge of the theory and practice of AI as a discipline concerned with the methodology and technology for approaching difficult problems. Recently, work has been done to address the diversity of topics typically covered in an introductory course and to create a theme-based approach. Russell and Norvig present an agent-centered approach [16]. Some courses seek to use robotics to unify diverse AI topics [6]. However, in an introductory course, one often wishes to impart a wide variety of topics efficiently, indexing the major areas of the field. This requires that an experiential education allow for rapid prototyping which unfortunately robotics does not allow. In effect, the engaging application limits the breadth of such a course. Robotics teaches through hard experience that a simulated world is often much easier to handle than the real one. By contrast, a machine learning application can be rapidly prototyped, allowing

learning to be grounded in engaging experience without limiting the important breadth of an introductory course.

Our project, Machine Learning Laboratory Experiences for Introducing Undergraduates to Artificial Intelligence (MLExAI), incorporates machine learning as a unifying theme across multiple AI topic areas to address the issues raised above and to enhance student learning experiences in the introductory AI course. Machine learning is inherently connected with the AI core topics and provides methodology and technology to enhance many real-world applications. Machine learning also provides a bridge between AI technology and modern software engineering. As Mitchell [11] points out, machine learning is now considered as a technology for both software development (especially suitable for difficult-to-program applications or for customizing software) and building intelligent software (i.e., a tool for AI programming).

Machine learning is listed as #1 in a Computerworld article "12 IT skills that employers can't say no to" [2]. The article quotes Kevin Scott, a senior engineering manager at Google: "As companies work to build software such as collaborative filtering, spam filtering and fraud-detection applications that seek patterns in jumbo-size data sets, some observers are seeing a rapid increase in the need for people with machine-learning knowledge. … There are lots of applications that have big, big, big data sizes, which create a fundamental problem of how you organize the data and present it to users. Demand for these applications is expanding the need for data mining…companies are snapping up these skills as fast as they can grab them."

Project MLExAI involves the development, implementation, and testing of a suite of projects that can be closely integrated into a one-term AI course. Using machine learning as a unifying theme is an effective way to tie together the various AI concepts while at the same time emphasizing AI's strong tie to computer science. We focus on fundamental algorithms and knowledge representation [7,14,15].

Each project involves the design and implementation of a learning system which enhances a particular commonly-deployed AI application. In addition, the projects provide students with an opportunity to address not only core AI topics but also many of the issues central to computer science, including algorithmic complexity and scalability problems. The rich set of applications that students can choose from spans several areas including network security, recommender systems, computational chemistry, robotics, conversational systems, cryptography, web document classification, vision, data integration in databases, bioinformatics, pattern recognition, data mining, and games.

3. A COMPUTER GAMES THEME

In our recruiting efforts and classes we have noticed a significant number of students citing computer games as a motivator for their selection of majors, and expressing great interest in computer game design projects. Surveys from our own institutions as well as others (e.g., [12,18]) strongly suggest that these are not isolated observations.

The benefits of game design extend far beyond simply catering to popular culture. Computer games require the skillful integration of a range of computer science techniques such as graphics, networking, and artificial intelligence. As a result, games can serve as ideal projects for an upper level capstone course. At the lower level, games may help make projects more interesting and relevant to students [1,5,13], and may help with both recruitment and student retention [1,12]. Together, these attributes suggest that games related curriculum may be a powerful tool for creating exceptional computer scientists.

The primary goal of commercial games is to entertain. As a result, AI techniques are useful only if they improve the game play or can otherwise make the game more compelling to customers. As a result, the techniques applied in industry are often relatively shallow, providing only enough depth to give the illusion of intelligence. This approach tends to give the greatest returns as resources can then be spent on more visible features (such as art).

At least from a theoretical level, any computer game that generates character behavior (e.g., stimulus response reactions, or path planning) may potentially benefit from the AI techniques that are often discussed in an introductory undergraduate class. Classic board games like checkers are natural candidates for adversarial search techniques and often form the motivating examples in textbooks. Indeed, many AI programming projects are built around a version of Checkers, Chess, or even Clue. A host of purpose-built AI research environments such as the classic Wumpus World also resemble games.

Real time games, on the other hand, are less often used in the AI classroom. In part this is probably due to the fact that they are more complicated to implement and thus they impose a higher burden on the instructor. Nonetheless, real-time games can make very interesting test beds and today form the basis of a variety of academic and military AI research and simulation environments (e.g., [10]).

4. SYNERGISTIC CURRICULUM

Studies have shown student motivation, and in turn quality of learning experience, is dramatically affected by the problem domains an instructor chooses for examples and assignments [3,19]. A problem domain that a student relates to and finds relevant leads to deeper understanding and hence smoother transfer to other domains, something that assessment of our work supported. This is particularly true when computer games are used as the domain. In our experiences with the machine learning projects, we have noted that students often express an interest in games when looking for machine learning applications; similarly other educators have noted that blending AI and games can help to attract a larger body of students than AI alone [17]. Modern computer games have increasingly been using AI and machine learning techniques and as Tom Mitchell points out [11] machine learning is now considered an important technology for developing and customizing complex software applications, which also include computer games. All these factors contribute to the synergy between machine learning and computer games themes in the AI curriculum.

Originally proposed in [18], the Java Instructional Gaming (JIG) Project is a collaborative effort between Washington State University Vancouver and the University of Puget Sound. The JIG Project has three main aims: to build the software infrastructure to help bring game-based projects to students at all levels in the CS curriculum; to create a set of educational resources to foster the use of Java based game projects especially

at small, resource-limited, schools; and to develop a community of educators that use and help improve these resources.

In addition to providing a software infrastructure for student built games, the JIG project is also developing a set of curricular modules that can be used throughout the CS curriculum as early as CS1. The goal of these modules is to provide a compelling vehicle for teaching core CS topics without undue burden on the instructor. The project described below, which is the synergistic result of both the MLExAI and JIG projects realizes this goal.

In this project, students build a variety of AI components to control the behavior of non-player characters in a real-time strategy game. Students improve the game incrementally, by implementing AI algorithms: simple, default behavior, is replaced by AI algorithms capable of generating robust behavior. Thus the project described here contains a number of individual assignments. Although the intent is for these assignments to be used together throughout a semester long course, it is also possible for instructors to pick and choose among the assignments, selecting only those that fit the needs of their individual course.

The assignments in this project cover three core areas of Artificial Intelligence: Search, Knowledge Representation, and Machine Learning. Search and Machine Learning are covered in projects 1 and 3 respectively; all projects require some discussion of knowledge representation.

4.1 The Game: Robot Defense

Robot Defense takes place in the future, when a strange variety of insects threatens the future of the world's food supply. Due only to good fortune, the insects have thus far been isolated to a small island and scientists must race against time to protect the world. The player's goal, as the operator of the island's scientific outpost is to gather insect samples for the scientists to evaluate before the insects find a way off the island.

The game bears similarity to the classic arcade game Lemmings, and the modern flash game Desktop Tower Defense. It takes place on a small, two dimensional map. The initial environment can be modified by the player who can place small structures in a grid-like pattern. Each structure serves to gather insect samples or keep the insects from spreading off the map. Well placed structures serve both purposes.

Robot Defense contains a variety of structures, a brief description of each follows:

Vacuum Tower: Vacuum towers capture nearby insects with a powerful suction. The suction is not continuous, but can be turned on and off and directed in any desired direction from the tower. While the tower's suction always produces some force, the magnitude of that force and its direction is a complex function of the local environment.

Particle Filter: A particle filter provides a barrier for robots and insects alike, although this barrier is fragile and degrades over time. Filters can be used to direct the insects across the map, but they cannot be used without restraint.

Fan: Unlike vacuum towers, fans cannot capture insects. They can however, help to redirect their movement. Thus, fans can be placed to slow down the insect movement or to redirect them towards vacuum towers.

Robot Control Center: The robot control center is the heart of the island operation. Each control center can govern four robots. Robots collect resources from the environment to keep the fans and vacuum towers operating. Resources appear at arbitrary locations on the map over the course of the game. Their appearance can be detected by each control center which must then determine which (if any) robot to assign to the new collection task. Periodically, robots must also carry out tasks specified by mission headquarters. These tasks may require travel to arbitrary locations on the island, and therefore constrain the placement of structures as described in the following section.

4.2 Initial Game Play

The unimproved game can be played by a student, and some experimentation is important to help develop an understanding of how the environment works. Play begins with a setup phase in which the map is completely empty. The player places structures of various types onto the map so as to capture as many insects as possible. The placement of structures is limited by two factors. The first is cost: the player has a fixed number of credits to begin with, and each new structure built reduces this value. The second constraint is due to the robots which must periodically leave the map to carry out tasks for mission headquarters. Thus, at all times a path must exist that will allow an arbitrarily selected robot to exit a specified side of the map.

Once structures have been placed on the map, real-time play begins. From this point onward, insects trickle through the map, attempting to cross from the left side to the right (west to east). Insects are simple creatures and will always take the most direct path across the screen (such a path must always exist due to the building constraints mentioned above).

The player engages in the game by performing three types of actions:

1. directing robots to acquire resources
2. directing the flow of the vacuum towers
3. adjusting the direction of the fans

By directing robots to acquire resources from the environment, the player ensures that their structures (fans and vacuums) continue to operate for the duration of the game.

By directing the vacuum towers and adjusting the directions of fans, the players change the aerodynamics of the local environment which in turn affects how well towers capture insects. That is, the probability an insect is captured by a particular vacuum tower is a function of the type of insect to which suction is being applied, the location and types of other nearby structures and insects, and the direction and magnitude of other air currents (due to fans or other vacuums operating in the area). The player's task is simply to learn an approximation of this function so they can set vacuum towers and fans to capture a large number of insects. The game is designed so that this function can be set to one of several pre-determined values, or selected randomly. Random selection means that each instance of the game requires the player to relearn their strategy.

4.3 Assignments

Assignments are designed such that the game's functionality becomes increasingly sophisticated during the course of the semester. A sequence of three projects is outlined below.

4.3.1 Project 1: Robot Route Planning

Robots have two tasks that require path planning: gathering resources and performing the periodic off-map tasks assigned by control towers. In this assignment, students implement route finding approaches appropriate to both tasks.

Resource gathering requires robots move to arbitrary locations on the map (wherever the resources appear). Search is the best approach to apply to this situation, and students implement A* search using a graph representation of the map provided by our software infrastructure. By experimenting with maps of various sizes, students see first hand the impact of exponential growth on computation time. This provides a good opportunity to discuss practical methods that tradeoff optimality for solution speed.

The second route finding task differs from the first in that the destinations (edges of the map) are static and do not change over time. Because of this, search is not required during game play. Instead students can precompute the optimal path from any location on the map using Best-First Search and cache the information for fast retrieval later in the game.

4.3.2 Project 2: Rule Based Approaches

After playing the game on their own, students design a simple set of rules to control how robots are assigned to different resource gathering tasks, and to control the settings of fans and vacuums. The goal of this assignment is not to build a perfect solution, but rather for students to distill their tactics into well defined rules.

Rules are built in Soar [9], an open-source agent architecture built on a variant of the RETE matching algorithm. Soar has been in active use since the 1980s. It has been used in conjunction with a number of commercial computer games such as Quake [8] and Unreal Tournament [10] to create research test beds. Under typical conditions, Soar can run in soft real time with such games. Deployed systems have been demonstrated to maintain efficiency with upwards of tens of thousands of rules.

This assignment provides a good backdrop for introducing concepts of knowledge representation. We expect that many of the rule sets submitted by students will be relatively brittle, in that they will be designed for one particular aerodynamic function, but will not be robust enough to perform well if the function is selected randomly.

4.3.3 Project 3: Learning a Vacuum Policy

In this project, students implement and use reinforcement learning instead of hand coded rules to identify the value of actions in each particular state. Students are given a preselected series of levels which define the map size, a set of structures and locations as well as the number and types of insects that will appear during the game.

Students implement Q-Learning to create a policy identifying the value of actions in each particular state. An important discussion centers on different representations of the game state, as not all information is relevant for learning. Domain knowledge based on the rules of the game can be used to significantly improve learning rate.

Once their learning algorithm has been implemented, students use the game as a test bed for running experiments. In their first experiment, students explore how state representation impacts learning rate. Next, students explore the robustness of learned policies by comparing their learning system with their hand built rules for a variety of aerodynamic functions. Both experiments highlight the relevance of previous discussions on knowledge representation and the potential brittleness of hand crafted rules. In addition, the exercise also serves to broaden a student's perception of games. By the end of the project, we hope that students will view games not only as entertainment products, but also as potential research vehicles. A short lecture on serious games will further highlight research applications.

5. PREDECESSOR ASSESSMENT

The Robot Defense module described in this paper is part of MLExAI's second phase in which twenty new projects are being implemented and assessed over the next twelve months. Robot Defense will be used for the first time in the Spring semester of 2008. Although formal results have yet to be obtained for the game module, six projects from the first phase of the MLExAI project have been tested and assessed in the classroom. We believe that these results provide insight on what to expect from future curricular modules such as Robot Defense. Below, we briefly describe the Phase 1 assessment, and follow this with a discussion some special considerations for our upcoming assessment of the Robot Defense curriculum.

Phase 1 curriculum was implemented and tested eight times over a three year period at three diverse institutions: Gettysburg College, a small liberal arts college; University of Hartford, a private comprehensive university; and Central Connecticut State University, a large public university. Led by the project evaluator, an independent evaluation of this work involved students, faculty, internal and external evaluators and a distinguished advisory board. Students responded to a written survey and a sample of students was interviewed from each of the three locations using either a direct (group) interview or a group conference call.

The survey was comprised of 22 ranked questions and two open-ended questions. Ranked questions were scored by students from 1 (strongly disagree) to 5 (strongly agree). The survey was issued to all students in the eight introductory AI courses. Course size varied from a minimum of 12 students to a maximum of 25. The total number of respondents was approximately 75.

The majority of students was enthusiastic about the projects and felt that they were an effective way to introduce Machine Learning and fundamental AI concepts. Their responses indicated a high level of confidence in their ability to apply these problem-solving techniques in other situations. In summarizing their experience, students stated that they had a very positive learning experience in the course and many indicated the desire to pursue additional study or research in AI and Machine Learning. These sentiments were reinforced by their responses to an open-ended question that asked what they liked best about the course. Table 1 includes the range of percentages over all evaluation locations and periods of those who replied with 'Agree' or 'Strongly Agree' to 4 selected questions:

Question	%A & SA
The student project contributed to my overall understanding of the material in the course.	88-100%
After taking this course I feel that I have a good understanding of the fundamental concepts in AI.	83-100%
Based on my experience with this course, I would like to learn more about the field of AI.	71-100%
I had a positive learning experience in this course.	75-100%

Table 1: Range of Percentages

These results indicate that students felt MLExAI curriculum significantly enhanced their learning experience, and we expect similar results following the assessment of the Robot Defense curricular module. Because Robot Defense is a product of both the MLExAI and JIG projects, assessment will yield data that can be analyzed within that global scope. That is, after assessment we will have a better understanding of how game-based projects impact learning in the AI classroom (by comparing against standard AI curriculum and against other non-games based projects from MLExAI). In addition, we will have a better understanding of how students view game-based projects by examining the assessment results of all JIG related curriculum.

6. CONCLUSIONS AND FUTURE WORK

We presented a curricular module that includes a set of game-based student projects designed to teach students core Artificial Intelligence (AI) topics typically covered in an introductory AI course. The module is a synergistic result of two curricular approaches funded by NSF, a machine learning based theme, project MLExAI, and a computer game based theme, project JIG.

The module described here is one of twenty projects being implemented and assessed as part of phase 2 of project MLExAI. Assessment results of six projects developed under phase 1 of MLExAI are presented here and demonstrate that this environment significantly enhanced their learning experience. We believe that these results provide insight on what to expect from future curricular modules such as the one presented here, Robot Defense. Six games-based projects, including Robot Defense, are currently under development within the MLExAI project. Together, these projects will help us to assess how game based projects in general impact learning in the AI classroom.

7. ACKNOWLEDGMENT

This work is supported in part by NSF DUE-0409497, DUE-0716338, and DUE-0633726.

8. REFERENCES

[1] Bayliss, J. D., The Effects of Games in CS1-3, *Proceedings of the Microsoft Academic Days on Game Development in Computer Science Education*, 59–63, 2007.

[2] Brandel, M., 12 IT Skills Employers Cannot Say No To", ComputerWorld, July, 2007.

[3] Bruer, J. T., Schools for thought: A Science of Learning in the Classroom. MIT Press, 1993.

[4] Carnegie, D., How to Win Friends and Influence People, revised edition, Simon & Shuster, New York, NY, 1981.

[5] Decker, A., Haydanek, S., Egret, C., When objects collide: abstractions over common physics problems for capstone projects in CS1, Journal of Computing Sciences in Colleges, 21(2), 12–18, 2005.

[6] Dodds, Z. et al. (eds.), Robots and Robotics in Undergraduate AI Education, Special Issue of AI Magazine, 27(1), AAAI Press, 2006.

[7] Kumar, A., et al., Non-Traditional Projects in the Undergraduate AI Course. *Proceedings of the 37th Annual SIGCSE Technical Symposium on Computer Science Education*, ACM Press, March 2006.

[8] Laird, J. E., Using a Computer Game to Develop Advanced AI, Computer, 34 (7), 70–27, July 2001.

[9] Laird, J. E., Newell, A., Rosenbloom, P. S., SOAR: An architecture for general intelligence. Artificial Intelligence, 33(1), 1–64, September 1987.

[10] Magerko, B., Laird, J. E., Assanie, M., Kerfoot, A., Stokes, D., AI Characters and Directors for Interactive Computer Games, *Proceedings of the 2004 Innovative Applications of Artificial Intelligence Conference*, San Jose, CA, July 2004. AAAI Press.

[11] Mitchell, T., The Discipline of Machine Learning, CMU-ML-06-108, July 2006.

[12] Parberry, I., Roden, T., Kazemzadeh, M. B., Experience with an industry-driven capstone course on game programming, *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education*, 91–95, 2005.

[13] Rankin, Y., Gooch, B., Gooch, A., Interweaving Game Design into Core CS Curriculum. *Proceedings of the Microsoft Academic Days on Game Development in Computer Science Education*, 22–27, 2007.

[14] Russell, I., Markov, Z., Neller, T., "Teaching AI through Machine Learning Projects", *Proceedings of the 11th Annual Conference on Innovation and Technology in Computer Science Education*, ACM Press, June 2006.

[15] Russell, I., Markov, Z., Neller, T., Enhancing Undergraduate AI Courses through Machine Learning Projects, *Proceedings of the 35th Frontiers in Education Conference*, IEEE Press, October 2005.

[16] Russell, S. and Norvig, P., Artificial Intelligence: a modern approach, Upper Saddle River, NJ: Prentice-Hall, 2003.

[17] Youngblood, G. M., Using XNA-GSE game segments to Engage Students in Advanced Computer Science Education. *Proceedings of the Microsoft Academic Days on Game Development in Computer Science Education*, 22–27, 2007.

[18] Wallace, S. A., Nierman, A., Addressing the need for a Java based game curriculum, Journal of Computing Sciences in Colleges, 22(2), 20–26, 2006.

[19] Wilensky, U.. Abstract Meditations on the Concrete and Concrete Implications for Mathematics Education. In I. Harel & S. Papert (Eds.), Constructionism. Ablex: Norwood, NJ. 193-203, 1991.