Shallow Blue: Lego-Based Embodied AI as a Platform for Cross-Curricular Project Based Learning

Robert I. Selkowitz[*] and Debra T. Burhans[+]
Canisius College Departments of Physics[*] and Computer Science[+]
2001 Main Street, Buffalo, NY 14208
{selkowir,burhansd}@canisius.edu

Abstract

We report on Shallow Blue (SB), an autonomous chess agent constructed by a small group of faculty and undergraduate students at Canisius College. In addition to pushing the limits of consumer grade components at low cost, SB is a focal point for interdisciplinary student projects spanning computer science, engineering, and physics. We demonstrate that undergraduate students can engage in rich, long-term robotic design and applied Artificial Intelligence (AI) from both hardware and software perspectives. Student outcomes of SB include senior theses, conference presentations, peer-reviewed publications, and admission to graduate programs. Students who participated also report substantial development in skills and knowledge applicable to their post-undergraduate education and careers.

Introduction and Background

Canisius College is a small primarily undergraduate institution located in upstate New York whose quantitative science degree programs include computer science, physics and pre-engineering. The latter is a two or three year program targeted toward transfer to a university with a full engineering program. The general education curriculum at Canisius is extensive, reflecting the liberal arts and Jesuit traditions of the institution. All three programs have small enrollments, with approximately 5 students per year in physics and engineering and 10-15 in computer science. These factors constrain STEM programs in terms of the breadth of courses and upper division electives that can be offered. Nonetheless, through the combination of a required senior project in physics, the strong interest of graduate school bound students across these disciplines, and faculty desire to meld meaningful student mentoring with research, it has been possible to create a thriving interdisciplinary AI group working on low-cost solutions to embodied AI problems. To date, the group has worked on two robotic platforms: Shallow Blue (SB), an embodied chess player, with full vision and assembly cost below $800; and Jim, a puppet-like humanoid head with capabilities for gestural and spoken emotional output. We report primarily on SB and outcomes for its student development team.

There are two goals for this project, one technical the other pedagogical. The technical goal is the design of a novel, low-cost embodied AI platform for both classroom and research use. The pedagogical goal is to provide a set of significant design and implementation tasks for students interested in careers in engineering, AI, and robotics. Engaging all three groups of students allows for a broader range of project themes than traditional AI approaches. This style of collaborative project is possible, if not preferable, at institutions with smaller CS, physics, and engineering groups.

The remainder of this section discusses the SB project in relationship to other undergraduate AI and robotics projects and courses, and addresses the reasons for and ramifications of the decision to use Lego as a primary medium for hardware development. In the following section, we provide an overview of SB, examining the design relative to other embodied chess agents in terms of cost, hardware and software sub-systems, and the composition of the design team. Following this overview, details of ongoing system upgrades and ancillary projects are given. The paper concludes with a discussion of student outcomes, directions for development of additional AI systems, and the extension of the collaboration beyond traditional STEM fields.

Related Work

Touretzky (2013) cites the need for robotics education that gets at real problems rather than simply providing an embodied platform for implementing existing algorithms and covering old ground. He highlights the importance of

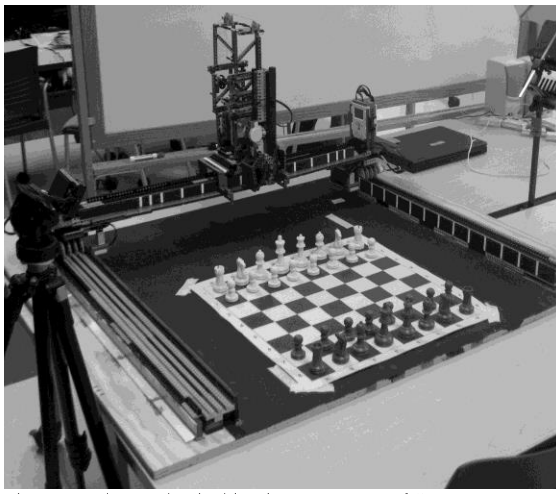


Figure 1. The mechanical hardware systems of SB.

connecting mathematics to real world problem solutions in a way that is often neglected in the CS curriculum. However, he feels that Lego is not sufficient in terms of support and hardware for deep college-level innovation. On the other hand, Klassner and McNally (2007) argue for the sufficiency of the Lego Mindstorms platform in AI education, citing their use of MatLab code that enables robust machine learning using the NXT platform. Sklar et al. (2007) employ robots in a variety of classes including AI, but switch from Lego to the AIBO for complex AI applications. Panadero et al. (2010) discuss a number of different classroom experiences with Mindstorms. Most relevant to this paper are observations regarding the positive impact of robotics experiences on student abilities in the areas of group work and problem solving. Specifically, they observed that the real world interactions and issues that arise with robot projects enhance problem solving skills, both individual and group; that students learn to carry out more robust testing; and that students are able to work iteratively on improving the performance of their computational artifacts. Akin et al. (2013) describe the use of Lego Mindstorms in a robotics course in a way that echoes the reasons we use the platform: it allows students to pursue fairly sophisticated explorations of hardware and software interactions in addition to teaching them about real world constraints and noise without requiring machining and electronics skills. Klassner et al. (2012) use sophisticated software and algorithms in a machine learning class with all programs running directly on the Lego brick. We made a decision to use Bluetooth for communication in order to avoid running complex software directly onto the brick, choosing instead to run the system on a laptop that communicated simple commands to the bricks. Tradeoffs such as these abound once a decision to use Lego is made. Kitts (2007) discusses projects that are very close to ours in spirit and practice, specifically undergraduate capstone projects on multi-robot systems that have involved many of the same features, including entry into competitions and creation of enduring artifacts. Departments with a capstone project requirement such as those within engineering schools already have a mechanism for building skills and experiences that may be less readily-available to a liberal arts CS major.

Why use Lego?

The decision to use Lego as the mechanical basis for SB was driven by three related factors. First, we had a pre-existing stockpile of Lego robotics equipment and a student population with some Lego robotics experience. Our introduction to engineering course and one of our CS0 courses utilize Lego robotics as a core curricular tool, as does an honors course and the upper-level AI course. In the engineering class, students engage in design and development of a single robot for a competition, while the computer science course uses robot programming as a means to motivate learning of control structures and algorithm design. Leveraging student familiarity with and enjoyment of Lego robotics, as well as the pre-existing parts collection greatly eased startup. Second, Lego provides a reasonably flexible means for hardware design with a limited investment in equipment and rapid learning curve. Students using Lego do not need to develop machining skills, do complex strength calculations, or wait for expensive custom parts. There is an obvious tradeoff: the strength of Lego is limited, as is the ultimate range of options for both actuation and structural elements. However, it is suitable as a bridge for students who might later learn the engineering or machining skills needed to engage in professional-grade robotics. Third, Lego provides sufficient onboard computing power, particularly ample for slaved mechanical control, which can be coupled with an off-board master to provide both higher level AI capability and opportunities to learn inter-device communication techniques. In particular, we discuss the latter two advantages in terms of claw development and Bluetooth communication between the NXT control bricks and PC.

The SB Platform and Chess

Compared with the other autonomous agents described in Table 1, SB is both the least expensive and the only one designed by a team reliant upon undergraduate students. The industrial arms, KukaMonster and ChessKA were designed to showcase industrial robotics. Maxwell, Gambit, Chiara, and SB were all designed for the AAAI small object manipulation challenge (2010, 2011). SB performed reasonably well at 2011 AAAI, with significant bugs limiting performance during the actual competition. By the end of the conference, however, thanks to intensive work on the code and physical system issues by the students and faculty, SB was capable of playing reliably, albeit without operational

Robot	Morphology	Cost	Speed	Perception System	Design team info
The Turk	Humanoid with arm	High	Standard	Magnetic board and hidden human operator	Wolfgang von Kempelen
Chiara	Hexapod walker	~$2.5k	Standard	Onboard camera for occupancy change	Tekkotsu lab/chess as a master's thesis
Gambit	Arm	~$18k	Standard	Kinect. Piece recognition functional	U of Washington and Intel collaboration
Maxwell	Mobile manipulator	Mid to low	Standard	Kinect. Piece recognition under development	Faculty and Masters' students
Kuka CHESSka	Industrial arms	High	Blitz	Full vision mechanism unavailable	Industrial robotics firms
SB	Cartesian translator	~$800	Standard	Still image comparisons for occupancy change.	2 Faculty and 2 undergraduate students

Table 1. A partial list of chess-playing robots. Only the industrial arms are capable of blitz play. SB is the only agent to rely solely on faculty and undergraduate students. Two platforms were adapted for chess as Masters' degree projects; one is academic-industry collaboration.

vision, vs. Maxwell on a single board and in a robot vs. human demonstration match. Performance the following year at the 2012 AAAI Robotics Fair was robust, with the robot fully operational as discussed below.

The SB platform can be roughly considered as five coordinated subsystems: the physical infrastructure (Hardware), the Master Control Program (MCP), the image processing system (Vision), the motion controllers (Movement), and chess decision making (Houdini). Of these, the first four were student-designed and coded. Houdini required a custom interface which meant students had to handle I/O issues such as parsing and string formatting as well as inter-process communication.

The SB physical architecture is tightly constrained by three concerns: Lego structural and motor limitations, the geometry of chess, and difficulty of motion control. To best satisfy these constraints, the students chose a Cartesian crawler design, as shown in Figure 1. For the purposes of this paper, hardware can be thought of primarily in terms of the bridge and rail system for translation across the board and the tower and claw system for lifting and manipulating pieces. Vision related equipment is external to the board assembly. A complete description of SB's design and operation can be found in (Lanighan et al. (2011), Burhans et al. (2014)).

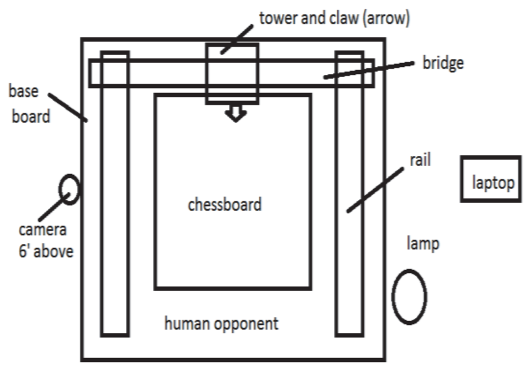


Figure 2. SB hardware schematic.

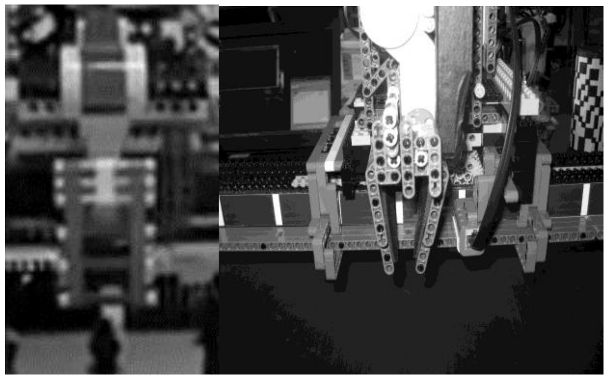
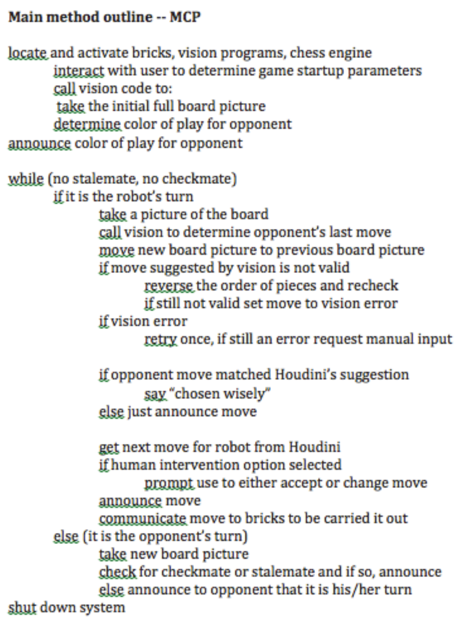


Figure 3. Two versions of the claw. The early prototype (left) was replaced with an improved claw (right). Note that the improved claw opens left-right, while the prototype opened front-back.

A bridge rolls across two fixed rails to reach the rows of the chessboard. Atop the bridge, a tower rolls across the columns and raises and lowers a gripping claw to manipulate chess pieces. Motion of the bridge is enabled by motorized gear wheels traversing Lego gear racks mounted on the side rails. While the initial plan was to utilize wheel-mounted rotation sensors to track position across the grid, it became readily apparent that accumulated sensor error would lead to large motional inaccuracies after a small number of chess moves. Instead, a series of white position marks were affixed along the bridge and rails at desired stop points, and color sensors used to count stripes to track position. The tower rides along the rail, and carries a retractable claw, which is raised and lowered via pulley. A guide shaft is used to keep it in alignment with the tower. Pieces must be lifted high enough to pass over the tallest impediments on the board (kings). The height constraint and position mark system combine to require the claw shaft to sit alongside the main body of the tower, pushing the center of mass to the lead end of the bridge, and resulting in the oscillation problem discussed below. The position mark system also results in a lack of freedom to fine tune the pickup and dropoff points for a chess piece; the claw must cover a full square on pickup, and return the piece to a near centered position on drop off.



```
Main method outline -- MCP

locate and activate bricks, vision programs, chess engine
        interact with user to determine game startup parameters
        call vision code to:
         take the initial full board picture
        determine color of play for opponent
announce color of play for opponent

while (no stalemate, no checkmate)
        if it is the robot's turn
                take a picture of the board
                call vision to determine opponent's last move
                move new board picture to previous board picture
                if move suggested by vision is not valid
                        reverse the order of pieces and recheck
                        if still not valid set move to vision error
                if vision error
                        retry once, if still an error request manual input

                if opponent move matched Houdini's suggestion
                        say "chosen wisely"
                else just announce move

                get next move for robot from Houdini
                if human intervention option selected
                        prompt use to either accept or change move
                announce move
                communicate move to bricks to be carried it out
        else (it is the opponent's turn)
                take new board picture
                check for checkmate or stalemate and if so, announce
                else announce to opponent that it is his/her turn
shut down system
```

Figure 4. Outline of the MCP algorithm.

The claw and iterative hardware design

SB's piece manipulator is a two-fingered claw, with only one of the two fingers capable of motion. It represents the endpoint of an iterative hardware development cycle that was clearly enabled by the Lego platform, indicating the deeper engineering process allowable within this framework. Initial ideas for the claw involved a three or four-fingered pincer with each pincer finger controlled by a single motor and a set of gears. The students prototyped a few gear systems for control and found that the gearing was too cumbersome for the space requirements of SB's tower. The second and third versions were each more successful, and are shown in Figure 3. Both of these claws use two fingers to clamp a piece in a single direction. This way, the piece can only be centered along one dimension of the chess square, a compromise relative to the fully centering pincer. Claw version two had two movable fingers, which allowed for a wider grip; however play within the gears did not allow for reliable gripping. Shortly before the competition, the students redesigned the claw to use a single mobile finger and added rubberized pads to increase friction. This claw is robust.

The claw design process required students to carefully weigh and balance tradeoffs in design optimization: centering capability and dexterity were exchanged for reliable closure and grip strength. Furthermore, the students were able to engage in part prototyping and manufacture. Here it is instructive to compare the process to three alternative approaches to embodied AI instruction. In case 1, a mostly predesigned platform is used such as the Finch, Scribbler, or Create (www.finchrobot.com, parallax.com, www.irobot.com). While excellent for projects with a predominantly software focus, a prefabricated platform does not allow students to address hardware issues in as much detail as can be accomplished with Lego. In case 2, component parts are purchased and assembled, as with the ER-1 (Evolution Robotics, no longer manufactured). In this mode, students have some freedom to solve hardware problems, but mostly in the context of assembling component sub-systems. Case 3 requires prototype design and machining or 3-d printing of parts. For example, this approach was used by Maxwell (Ferguson et al. 2011). In terms of design practice, it is the richest and most powerful paradigm, however, the skill and cost investment, even for 3-D printing, exceeds that which is often available for small group undergraduate projects. Thus, Lego provides an intermediate case: students can experience task specific hardware design and refinement without investing more time than is available during a typical undergraduate schedule.

Software development: The MCP and communication

The MCP carries out control and coordination functions for SB, issuing motion orders to Movement, querying vision for updates on changes in chessboard state, and accessing Houdini for chess decisions. An outline of the MCP algorithm is presented in Figure 4.

The vision subsystem is written in C++ and uses OpenCV to process board images. Initially, a blank board image is taken in order to set corner points for board squares and to store color histogram information about central regions of interest (ROIs) in board squares. After the initialization step, still images of the board following each opponent move are analyzed and compared to previous board images using OpenCV. A detailed description of the vision code and algorithms can be found in (Burhans et al. 2014). Both top-down rules about chess play and bottom-up image information are combined to determine the best possible candidate for the move made by the opponent. This move is checked for validity with Houdini.

Communication with Houdini is accomplished through the use of several methods, including `isValidMove`, `bestMove`, `suggMove`, and `isMate`. All of these methods take as a parameter `movesFromStart`, a string containing all of the completed moves of the game encoded in the format specified by the Uniform Chess Interface (UCI). The vision system calls `isValidMove` with its proposed move for the opponent appended to `movesFromStart` to see whether the completed moves followed by this new move represent a valid game. If so, the vision system concludes that this move is in fact the move made by the opponent.

An abstract class, `ChessEngine`, specifies what is needed to interact with a chess engine in general, and contains both abstract and non-abstract methods including a constructor. In particular, the methods for sending and receiving messages to and from a chess engine along with commands to execute that chess engine are included here. For SB this class is extended by `HoudiniEngine`, which contains methods and data structures that pertain specifically to Houdini. Note that Houdini was not our initial choice for a chess engine due to the fact that it is a Windows-based program. We were using Linux for other robotics projects and planned initially to use it for SB. With this in mind we also did initial system development with Stockfish 2.0.1, the other top-rated chess engine that was UCI-based. Houdini was selected primarily because it was the strongest chess engine available at the time. We parameterized Houdini to play at the level of approximately 2987 (Grand Master). The desire to plan for using other chess engines and other platforms led to the flexible, hierarchical design of the software. Using a different UCI-based chess engine would require the creation of a new subclass of `ChessEngin`e that provides engine-specific functionality, but the rest of the system would remain unchanged. Our code is also capable of using and parsing FEN (Forsyth-Edwards Notion) enabling upgrades to allow a mid-game start.

An important and interesting aspect of the development of the system architecture involved communication with and computation on board the two Lego bricks. As in many real world projects, there are important tradeoffs about where best to locate processing tasks. The bricks run Lejos programs that receive start and stop signals from the laptop. Initially, the bricks ran complex programs that handled the tasks of translating chess moves to motions as well as tracking sensor data and motor control. During design, an alternative arose: to treat the bricks as slaves that would return information about the world through their associated sensors and carry out simple movement commands sent by the laptop. After several iterations of design it was decided to opt for the latter system.

This decision required the development of a hardware independent communication protocol to transmit signals from the laptop to the motion controllers. For SB this meant creating mapping behaviors onto a set of integer codes that could be easily transmitted and quickly interpreted by the bricks. Grappling with these possibilities developed new skills for the students.

To enable Bluetooth communication it is necessary to reset the bricks each time the MCP is restarted to run a send/receive initialization routine, but once a game is in process communication proceeds smoothly. Finding information about using Bluetooth with the bricks was a challenge which required the students to seek and modify preexisting software in a manner exceeding typical classroom projects.

Motion and hybrid Software/Hardware solutions

Motion control problems with early versions of SB illustrate a third strength of the Lego approach to embodied AI: coupling of hardware and software solutions to a design problem. Much professional embodied AI work involves mechanical systems and their control. In order to best achieve reliable motion, it is often necessary to iteratively co-develop the software and mechanical hardware used for an agent as challenges arise. In the case of SB, one challenge arose during long distance motion of the bridge across the rails. Due to the high center of mass and off-center weighting of the tower, it routinely oscillated front-to-back across the bridge, resulting in torsion of the mechanical systems and motion of the color sensors relative to the position stripes. This in turn caused misreading of position, and failure of the tower and bridge to stop at appropriate points. Under traditional pre-packaged approaches to undergraduate robotics, the problem would have been solved prior to student engagement, or addressed by students solely through software. Our use of Lego allowed for modes of problem solving more akin to those encountered in research and industrial settings.

The students identified three major causes of the oscillation problem: rapid acceleration and deceleration of the tower, an unbalanced load on the tower due to the offset claw, and the overall high center of mass of the tower relative to its width. The latter of these three problems could not be addressed, as range of travel of the claw was required to be slightly larger than the height of a king. The tower thus needed to be tall enough to support that motion. Instead, the students struck upon solutions addressing the other two causes of oscillation. First, the claw was counterbalanced by 250g of mass hanging from the opposite side of the tower. This placed the center of mass closer to the horizontal center of the tower. Second, a set of guide rails were added to the bridge and a hook added to the tower. The hook engages the rails, limiting tipping motions. Finally, a stepped deceleration routine was added to the control software: a few position markers short of the final location, the bridge motors begin reducing speed, producing a gentler final stop. The hardware and software solutions were each insufficient on their own to resolve the oscillation problem. It was only by coupling partial solutions that used both software and hardware that students were able to correct the problem.

Long-term development experiences

An additional benefit of the SB project was the opportunity for students to engage in relatively long term development and redevelopment of a system. Project-based learning within a course allows for initial design and implementation, but projects within a course usually conclude at the end of the semester. Similarly, the physics senior thesis project is a one-semester activity, carried out at the end of the curriculum. SB, through the touchstone of AAAI robotics activities, provided a venue for ongoing student engagement with an extended project.

Participation in this kind of activity has better prepared students for post-graduation work and educational opportunities. SB was presented twice by students at AAAI: version 1 was brought to AAAI 2011 and version 2 to AAAI 2012. Four major issues surfaced in 2011 that necessitated revision of the system, including the following: vision failed due to the extremely dim lighting conditions; the paper marking strips that indicated the locations of the rows and columns of the chess board wore off and tore; the rails required realignment at each set up, namely, they needed to be clamped to a table surface with exacting and difficult spacing requirements; and the interface from the MCP to Houdini periodically failed. In addition, during the competition the system had problems with piece movement and placement. By the end of the 2011 conference the bugs in the MCP/Houdini interface were eliminated, the movement problems were fixed, the strips were temporarily improved, and the rails were effectively placed. Unfortunately, the vision system would not function under sub-optimal lighting conditions.

Following AAAI 2011 the vision problem was solved by adding a studio light system with diffusing umbrella to the set up. Additionally, the vision routines were upgraded and fine-tuned, as discussed in Burhans et al. (2014). The stripes were found to wear out from passage of the bridge and tower, resulting both in degradation of position detection and the presence of adhesive in the racks used to guide motion. Both resulted in misread of final position and cases of the bridge twisting and binding as it moved. New stripes were printed on vinyl and installed. Finally, it had been necessary each time SB version 1 was set up to align the rails parallel to each other and clamp them in place. This process was time consuming and prone to error. Even slight misalignment could result in torsion and binding of the bridge as it moved across the rails. The rails were subsequently permanently mounted on a single plywood board, obviating the need for on-site realignment. As a result, set up time was reduced to less than 15 minutes, with most of the effort involved in unpacking and placing the vision equipment.

When SB was brought to AAAI 2012, it participated in two full-length exhibition chess matches against Dr. Kenneth Regan, an international master rated chess player and known computer scientist and commentator on tournament-level chess. Against Dr. Regan, SB played autonomously, with full vision, chess reasoning, and motional capabilities. SB played additional matches against passersby, with human opponents often making a few moves and then turning play over to the next opponent. The upgraded version was able to complete multiple games during the conference. Importantly, both students attended this conference and were able to see the results from their significant investments in SB.

Ongoing Project Development

While the students involved in the creation of SB are no longer undergraduates, we continue to pursue areas of development and applied research using the SB platform with new undergraduates and faculty. These include upgrades to the vision system necessary for implementation of detection of a mid-board game state, which would allow SB to begin play from a midgame position or to solve chess puzzles; and an exploration of the experience of playing against SB, which involves collaboration with social scientists.

To date, SB has played against a small number of human opponents. Primary human opposition has been members of the project team (Sikorskyj, Selkowitz) and Dr. Regan. It is noteworthy that Dr. Regan actively dislikes playing onscreen chess against computers, as well as online chess against human opponents.

All three players report a sense of connection to the robot during play, which approximates that between two human players more closely than does play versus an onscreen computer opponent. Notably, Dr. Regan found himself talking to and about SB as if it were human, referring to “his creativity” and complementing “him” on good plays. Dr. Regan does not express this same humanization of Houdini (“it”) onscreen, and this humanization gives rise to a distinct pleasure in playing against SB. We suspect that the experience of playing versus a robot can be a closer approximation of in-person human-vs-human chess than is human-vs.-human onscreen play, and are currently undertaking a detailed study of the phenomenon in collaboration with social science faculty and students. This sort of problem suggests one way in which AI can be more closely related to the Liberal Arts goal of holistic, humanistic education.

Student Outcomes

Student outcomes from the SB project fall into two related categories: closure of curricular gaps and development of critical design and problem solving skills. In particular, the computer science curriculum at Canisius College is constrained by institutional requirements and a small faculty and student body. Aspects of the SB project, including programming for simultaneous usage of multiple computing devices, management of projects requiring multiple programming languages, and scripting for input/output devices are rarely covered in the curriculum in small CS departments, yet these are valuable knowledge areas for computing professionals. The students who have worked on the project have benefited in many ways from their participation.

One of the original team members is currently employed full time working in IT Services at a major university. He is also engaged in a startup that has already garnered considerable attention from major players in the field. His thoughts on the project (private communication) are as follows:

> *"Working on SB improved my ability to work on larger projects, particularly the kind that have a well-defined goal, but with very little details or plan on how to get there. Coupled with a high degree of freedom to plan the roadmap, it was good preparation for functioning as a project manager and technical lead in my current work as an IT developer. I found it inspiring to work on a project in which the espirit de corps was "it had to get done". This aspect in particular, was foundational for my own thinking and cultural formation of my current social video startup company."*

The other original team member is currently a PhD student working in a perceptual robotics group. He received an Honorable Mention in the NSF Graduate Research Fellowship Competition, in part due to his background working on SB. His work with SB provided him with several advantages. First, he gained sufficient robotics experience to decide to pursue the field in graduate school. He also gained a sense of the scope and depth of work required for a practical robotics project. Finally, the experience of attending and publishing at AAAI enhanced his interest in and desire to join the culture of academic AI and robotics.

There are currently three students engaged in the upgrades to SB discussed above. A second group of students is working on Jim, another Lego-based project in the area of affective robotics. Both projects will be deployed in the AI course going forward, with each serving as a platform for project-based lab exercises with the potential to become longer term projects for interested students.

Conclusion

The students and faculty working on SB and other Lego-based projects have been drawn from Computer Science and Physics/Engineering programs. It is interesting to note that all of the students and faculty involved in the hardware design portions of these projects have taken calculus-based physics and at least two semesters of calculus. This sets them apart from most Computer Science students at our institution. We believe that engaging with physics or engineering students and faculty is a key part of this style of embodied AI project. While this approach may not be necessary at institutions with large CS departments, it can be invaluable for smaller programs.

In addition to these projects we participate in an interdisciplinary electronics-oriented research group with other students from CS and Physics as well as faculty from Music and Digital Media. Our most vigorous interdisciplinary connections in our liberal arts setting have a definite hardware orientation, and the mix of faculty and projects continues to attract students. Our goal is to continue to provide rich experiences to undergraduates that build upon their classroom projects and allow them to develop critical skills for moving on to graduate school or into the work force. These types of projects, involving both hardware and software, have led to an increase in the number of physics students choosing to major or minor in CS and a large increase in the popularity of our introductory programming sequence.

Acknowledgements

The authors acknowledge the hard work and dedication of Jerod Sikorskyj and Michael Lanighan in designing and implementing the SB platform.

References

Akin, H. L., Meriçli, C., Meriçli, T. 2013. Introduction to Autonomous Robotics using Lego Mindstorms NXT, *Computer Science Education*, 23(4), 368-386.

Burhans, D., Selkowitz, R., Sikorskyj, J., and Lanighan, M. 2014. Shallow Blue: A low-cost platform for embodied chess play. *Proceedings of IEEE 2014 TePRA,* IEEE, 138-143.

Ferguson, M., Gero, K., Salles, J., and Weis, J. 2011. Playing chess with a human-scale mobile manipulator, *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, AAAI Press.

Kitts, C. 2007. Undergraduate Capstone Projects on Multi-Robot Systems. Robots and Robot Venues: Resources for AI Education, Papers from the AAAI Spring Symposium (Technical Report SS-07-09), AAAI Press.

Klassner, F., and McNally, M. 2007. Demonstrating the capabilities of Mindstorms NXT for the AI curriculum. *Proceedings of the American Association for Artificial Intelligence Conference*. AAAI Press.

Klassner, F., Peyton-Jones, J.C., Lehmer, K. 2012. Genetic Algorithms with Lego Mindstorms and Matlab. Proceedings of the Twenty-Fifth International Florida Artificial Intelligence Research Society Conference.

Lanighan, M., Sikorskyj, J., Burhans, D., and Selkowitz, R. 2011. Lego plays chess: a low-cost, low-complexity approach to intelligent robotics, *Proceedings of the American Association for Artificial Intelligence Conference*. AAAI Press.

Panadero, C. F., Román, J.V., Kloos, C. D. 2010. Impact of Learning Experiences Using LEGO Mindstorms® in Engineering Courses. *Proceedings of EDUCON 2010* (Madrid), IEEE, 503-512.

Sklar, E., Parsons, S., and Azhar, M.Q. 2007. Robotics Across the Curriculum. *AAAI Spring Symposium: Semantic Scientific Knowledge Integration*. AAAI Press.

Touretzky, D. S., 2013. Robotics for Computer Scientists: What's the Big Idea? *Computer Science Education*, 23(4), 349-367.