6-2008

CSCI: A LEAP into the future

J. Abrantes
University of Wollongong, jo@uow.edu.au

A. Porter
University of Wollongong, alp@uow.edu.au

W. Meyers
University of Wollongong, meyers@uow.edu.au

Ray Stace
University of Wollongong, rstace@uow.edu.au

Willy Susilo
University of Wollongong, wsusilo@uow.edu.au

See next page for additional authors

CSCI: A LEAP into the future

Abstract

This paper outlines the development of a project which aims to improve the teaching and learning outcomes within the Computer Sciences. A major strategy being examined is the effectiveness of digital gamesbased learning. Utilising the Neverwinter Nights game engine the team have created a prototype to be trialled in the first half of 2008. The project forms part of a broader faculty based solution to address teaching and learning problems of first year students, known as QUALITY101.

Publication Details

This conference paper was originally published as Abrantes, J, Porter, A, Meyers, W, Stace, R, Susilo, W, Krishna, A, Zhou, ZQ, Judge, D, Franks, R and Xia, T, CSCI: A LEAP into the future, Proceedings of the Emerging Technologies Conference, University of Wollongong, 18-21 June 2008.

Authors

J. Abrantes, A. Porter, W. Meyers, Ray Stace, Willy Susilo, A. Krishna, Z. Q. Zhou, D. Judge, R. Franks, and Tianbing Xia

emerging technologies
CONFERENCE
Supporting a Learning Community

CSCI: A LEAP into the future

Jo Abrantes; Anne Porter; Wendy Meyers; Ray Stace; Willy Susilo; Aneesh Krishna; Zhi Quan Zhou; Daniel Judge; Ross Franks; Tianbing Xia
University of Wollongong

Abstract:
This paper outlines the development of a project which aims to improve the teaching and learning outcomes within the Computer Sciences. A major strategy being examined is the effectiveness of digital games-based learning. Utilising the Neverwinter Nights game engine the team have created a prototype to be trialled in the first half of 2008. The project forms part of a broader faculty based solution to address teaching and learning problems of first year students, known as QUALITY101.

Introduction

The LEAP project is examining the feasibility of using digital games-based learning within higher education. LEAP (Learning Experience And Problem solving) is a games-based learning project that emerged as one component of a broad strategy to address a teaching and learning problem within CSCI, first year Computer Science subjects at the University of Wollongong. QUALITY101, involving a team of academics, was charged with investigating the issues and developing a strategy. The research conducted by QUALITY101 revealed a number of factors including low student motivation, problems in delivery, activities and assessment. The games-based learning approach is one solution being investigated. It attempts to supplement traditional modes of delivery, allowing students an opportunity to practice skills within an engaging learning environment. The game development team is using the Neverwinter Nights game engine to develop an immersive environment enabling the deepening of generic problem solving skills, knowledge of programming concepts and strategies to solve coding problems. The LEAP project will trial the first prototype in the first semester of 2008.

Background to the project

In 2003 the Faculty of Informatics identified poor learning outcomes and high failure rates amongst first year Computer Science students. They formed a working party to investigate the issues and develop solutions in order to improve teaching and learning outcomes within the program. The working party became known as QUALITY101. They established an accountability cycle, action plan and evaluation framework to guide the investigators. The accountability cycle enabled the team to report directly to the Faculty Education Committee on a quarterly basis. The action plan targeted large core subjects with high failure rates. The evaluation framework utilises qualitative research methods including staff interviews and student questionnaires to inform the project. The LEAP project emerged from this process in order to investigate the feasibility of digital games to improve teaching and learning outcomes by focusing on conceptually difficult components of computing problems, problem solving in general and algorithms, and embedding these within a highly interactive digital environment.

The teaching of Computer Sciences

Historically and worldwide the teaching and learning of computer languages subjects has been fraught with difficulties and with poor educational outcomes. First year computer programming subjects are considered difficult by students (Smith and Webb, 1995; Garner, 2003; Costelloe, 2004; Yousoof et al., 2006) and have high withdrawal rates (Campell and Bolker, 2002; Robins et al. 2003). Academics teaching these subjects are often frustrated by poor student learning outcomes. A number of studies and approaches have been carried out around the world aiming to identify the problems and inform others attempting to raise students' understanding of subject contents and to make the studying and learning of these subjects more enjoyable (Jimenez-Peris et al., 1999; Gil, 2005).

Robins and colleagues (Robins et al., 2003) suggest that to be able to make teaching and learning more effective, the following questions must be addressed:

- Why is computer programming difficult to learn?
- What are the cognitive requirements of learning a first programming language?
- Do effective learning strategies exist?

Robins and colleagues (2003) draw on the work of Rogalski and Samurcay (1990) and du Boulay (1989), the latter identifying five possible sources of difficulty in learning a first year programming language. du Boulay suggests that students have difficulty because they must understand complex sets of ideas including:

1. General orientation: What kinds of problems there are and how programs can solve problems,
2. Notional machine: Models of the computer that allow the students to understand program execution,
3. Notation: Programming language syntax and semantics,
4. Structures: Building of schemas for operations such as loops and recursion,
5. Pragmatics: Skills involved in planning, designing, developing, testing and debugging programs.

Similarly, Rogalski and Samurcay (1990) identified the large number of cognitive activities involved in the complex task of programming. These extend from basic structuring operations such as loops and conditional statements to incorporating these into schemes and plans, program design, understanding, modification, debugging and documentation. Associated with the complex task of programming are the sophistication of analytical skills and problem solving abilities required.

In attempting to gain a deeper understanding of effective solutions, Costello (2004) uses six categories to classify approaches aimed at improving the teaching and learning outcomes within computer programming subjects. These approaches were classified as:

1. Lectures and labs;

2. Software visualisation including program visualization, algorithm animation, visual programming, programming by demonstration and computational visualisation;
3. Robots;
4. Problem-based learning;
5. Cognitive apprenticeship, and
6. Miscellaneous.

However, within these different classifications there is enormous scope for differences. Reinfelds and colleagues (2003) discuss many paradigms for programming including, procedural, functional, object-oriented, event driven, logic, concurrent, parallel, genetic and quantum; they attempt to identify and teach the common core of concepts that suit multiple paradigms. Within the lecture and laboratory classification many different strategies are employed. Campbell and Bolker (2002) emphasize the use of immersion, reading and writing code when teaching programming. Garner (2003) explored the educational impact of building programs from previously written modules and Vodounon (2006) found that using this approach there were improvements in high-performing students' ability to think logically and to divide problems into sub-problems as well as low-achieving students' improvements in ability to divide problems into sub-problems. Some academics have also explored the potential of more non-traditional solutions.

Examining the UOW situation

In order to institute fundamental and informed curriculum changes the QUALITY101 team drew on the work of Alexander and Hedberg (1994) and Reeves and Hedberg (2003) to develop an evaluation framework which was reflective and allowed feedback to inform the change process. The heart of this research process involved analysis, research, implementation, evaluation and feedback. The evaluation framework involves a four stage cyclic process:

- Stage 1: Design (including review and needs assessment),
- Stage 2: Development (formative evaluation),
- Stage 3: Implementation (effectiveness evaluation) and
- Stage 4: Institutionalisation (impact evaluation and maintenance)

From a Faculty perspective, the QUALITY101 team identified two key aspects of projects or innovations for improving teaching and learning outcomes that would lead to change in the educational culture. These key aspects involved the development of leadership and teamwork. Previous to the project, subject redesign involved single subjects and was not creating sustained improvement. As a process of the review it was considered desirable that several people owned innovations across a range of subjects. This alleviated a problem of vertical alignment, where in the past it was possible that lecturers, who have subjects that follow-on from one another, sometimes dismissed the improvements to the earlier subject. In some cases this resulted from a feeling that the changes allow too many students to progress thus forwarding the high failure rates onto the subsequent subject. Rarely was there seen the need to modify the follow-on subject as well. The

QUALITY101 framework alleviated this by including teams covering a sequence of subjects. The teams-based approach also informed parallel subject design. The team developed a collegial approach, encouraging Lecturers and Educational Developers to work together to look at ways to improve learning outcomes.

In 2003, the Faculty of Informatics analysed student results from 2000 to 2002. Data analysis revealed that within the introductory computing C++ programming subject there were significant problems marked by the annual failure rate of 26 to 39 per cent. On further investigation it became apparent that lecturers had been trialling and implementing strategies to improve learning outcomes. Many of these innovations were undocumented and happening in isolation from the program as a whole. Whilst done with excellent intentions the strategies failed to improve the situation. In 2003, the Faculty recommended a review and planned a restructure of the Degree. This was undertaken resulting in the addition of several new subjects. An existing subject CSCI111was divided into two subjects, CSCI103 and CSCI104, one focusing on algorithm design and one on C++ programming. These were developed as replacements for a single subject that attempted to teach both the algorithm design and coding. These changes were in accord with ideas that programmers firstly need to have good problem solving skills (Henderson, 1986; Linn and Clancy, 1992), analytical skills (Masheshwari (1997) or conceptual skills (Reinfelds et al, 2003) and that they must be able to implement their solutions in relevant language, executing and debugging strategies appropriate to the specific environment (Costelloe, 2004). Table 1 shows changes within the subjects from 2003 to 2006.

Table 1: Changes in operation and structure of subjects 2003-2006.

2003	Final implementations of the old curriculum, before the redesign of the single subject into CSCI103 and CSCI114. The two hour laboratory time is divided in a one-hour tutorial where the students are given a set of exercises and the tutors discuss these with the whole class. In the one hour laboratory, students work on their assignments.
2004	The existing subject was divided into two subjects one on algorithm design (CSCI103) and one on C++ programming (CSCI114)
2005	First implementation of a revised structure, the laboratory was meant to serve both purposes of lab and tutorial. The major changes were the successive introduction of structured exercises by way of a laboratory manual and the inclusion of tutorial time, enabling tutors to review these exercises with the whole class. This was refined in the second session and fully implemented in 2006.
2006	The laboratory program continues to be refined, but the assessment structure is also modified. Assessment is now eight short online tests, four assignments (instead of seven or eight) and a final examination.

Changes to subject design

As a result of the review two major changes were undertaken and trialled in 2005 within the program, namely the utilisation of Lab manuals and quizzes. Previously students' main form of activities and

formative assessment involved set assignments. In 2005 these were modified to include lab manuals, lab time (for students to work on selected exercises from their manual) and tutorials (enabling tutors to discuss and review exercises with the whole class). The exercises in the lab manual were aligned with topics covered in the lectures. Bloom's Taxonomy (Bloom & Krathwohl, 1956) was used to inform the development of the exercise covered in the lab manual. These included a weekly set of exercises of varying levels of complexity. This increased the students' skills in higher order thinking. Online quizzes, which utilised multiple-choice tests, were also implemented. The students undertook these, every second lab session, which provided, along with lecturer feedback, an opportunity for students to test their understanding of the fundamental concepts covered in the lectures. These tests counted towards a small proportion of the students' final marks.

In the autumn of 2006 grades showed an increase in high distinctions from 7.9 to 13 per cent and a lowering of fail grades from 28 to 22 per cent. However, the team felt there was more room for improvement. Student subject evaluations were undertaken as part of QUALITY101 in Autumn 2006. This provided important data to support the further development of the course. A student questionnaire was utilised and 29 students responded. Of those 65% of the students revealed that they spent less than 8 hours per week on their subject. This indicated there was a need to increase student engagement, participation and interaction. Students identified problems in a number of content areas specific to the subject including graphs, problem-solving strategies, programming logic errors and writing algorithms. When asked for feedback on improvements to the course, students said that there was a need for more time on more difficult tasks; increased exposure to relevant examples in particular worked solutions. The project team, having trialled a number of other educational strategies previously to increase student engagement argued for the development of a more interactive online tool such as digital games in line with the work of Robins and colleagues (2003).

An alternative solution to a traditional problem

Digital games-based learning is being identified as an effective teaching strategy within higher education, in general. Games enable a student to immerse themselves in their learning and increase their "fun", which increases student motivation and engagement (Ebner and Holzinger, 2007), and support problem based learning strategies (Kiili, 2007). Recent work within Computer Science education has also identified educational benefits of games-based learning.

The problems encountered by the academics responsible for the delivery of the Computer Science programs were not unique. Academics around the world have encountered similar problems. Some academics are turning to non-traditional methods of delivery in order to improve teaching and learning outcomes. Games-based learning is being examined as a viable part of the solution for education within the Computer Sciences.

In line with the work of Robins and colleagues (Robins et al., 2003), discussed previously, which identified groups of skills necessary for

successful learning within the computer sciences, preliminary work of academics utilising games-based learning appears promising (see Table 2). Games have pedagogical use in the teaching of computer programming by supporting the development of critical thinking and problem solving skills (Fasli M. and Michalakopoulos M., 2005; Neller et al, 2006; Rajaravivarma, 2005). Games also offer visual representation of abstract concepts, a component difficult to teach in programming subjects utilising traditional delivery method (Rajaravivarma, 2005). A well designed game appears to increase students' time "on-task", along with providing multiple opportunities for students to apply earlier learning to later problems (Gee, 2003; O'Neil et al., 2005).

Learning computer programming is recognized around world as difficult for many students, with students experiencing high levels of cognitive load (Costelloe, 2002; Garner, 2003; Yousoof et al, 2006). Students are required to learn the use of a program development environment, while learning the programming language syntax and developing logic design skills (Garner, 2001). The use of well-designed games can introduce students to these three aspects separately in an engaging and entertaining way.

Table 2: Alignment of Games based learning to identify problems in first year programming education

Benefits of Games Based Learning	Areas of concern for teaching programming
Visual representation of abstract concepts	1. General orientation; 2. Notional machine; 3. Notation
Critical thinking and problem solving skills	4. Structures; 5. Pragmatics

Evolution of a games based approach

In 2005 the QUALITY101 team successfully applied for funding to investigate the feasibility of games-based learning to improve the teaching and learning results within the first year Computing Science subjects. A project team was created consisting of content experts, learning designers, animators and computer programmers. The objectives guiding the project were to:

- Reduce failure rates in CSCI114 through encouraging practice
- Reduce failure rates in CSCI103 through improving students problem solving skills and understanding of key computer concepts
- Improve students' satisfaction with their learning outcomes in CSCI114 and CSCI103 through providing engaging and motivating activities and through developing a greater sense of competency
- Develop an engine (template) for gaming that is adaptable to other 100/200 level technical and hierarchical subjects e.g. Mathematics
- Develop a creative and energetic 100 level teaching team over the two parallel subjects CSCI103 and CSCI114
- Provide a more viable approach to improving faculty learning outcomes than the hitherto subject-by-subject approach

- Gather evidence for the QUALITY101 team regarding the impact of strategies implemented to improve learning outcomes in Informatics

If the teaching strategy proves feasible the LEAP project will seek external funding to expand its initial pilot via an ARC or Carrick grant.

Development of the LEAP project

The experience of the QUALITY101 academics, having implemented initial redevelopment of the first year subjects, identified the potential of games to provide an alternative opportunity for students to study within an environment in which they are intrinsically motivated to engage. Whilst the changes to delivery, including the lab manual, lab tests and modifications to tutorials had improved results, the team felt that there was more room for improvement. Students' motivation and weekly time engaged had still showed little improvement. The potential for games to not only support curriculum objectives but to increase student engagement appeared to offer a viable solution.

Specific outcomes driving the LEAP project are:

- Reduce failure rates in CSCI114 and CSCI103
- Shift the grade distribution of passing students to higher grades while maintaining or improving standards
- Provide an authentic experience for Bachelor of Computing Science students choosing to major in the currently growing Multimedia and Games major, enabling them to be responsible for components of game development
- Increase student satisfaction with learning resources as measured by Change Evaluation Survey and Lecturer ratings
- Develop a game suitable for teaching and learning in CSCI114, in particular C++
- Develop a game suitable for extending to advanced C++ topics introduced in CSCI124
- Evaluate and implement modern game engines, allowing them to be used as examples in later multimedia and gaming subjects
- Develop a gaming template suitable for adapting to other technical disciplines such as Mathematics (an additional learning tool for the Summertime Math (Porter 2007), Statistics, Science and Engineering.
- Present findings to the academic community via, publications and conferences
- Improve the project via external funding i.e. ARC or Carrick grants

The project involved a cross-university team including four academics responsible for three core subjects; two parallel and one vertical subject. The academics were supported by staff in CEDIR (The Centre for Educational Development and Interactive Resources) who facilitated the design and development of the game along with

examining its re-usability potential enabling the future expansion of the game approach to other sectors of the University.

Not only was the project geared to support student learning within first-year core subjects, but also final-year students were involved in design and implementation. The third year students consulted the development team in the initial stage and developed a parallel game design. They developed an alternative interface with a second game engine, including PlaneShift; an open source game engine enabling MMORPG (Massively Multiplayer Online Role-Playing Game). The game play included activities in which students' complete quests using their C++ programming knowledge. The students put their knowledge into practice via an in-game code editor. The code written by the student-player is then compiled and run by an integrated GNU g++ compiler that runs on the background in order to trigger game events used to complete various types of missions. The student game will be trialled with first year students in the first semester of 2008 (and compared with the suite of games developed by the CEDIR staff using Neverwinter Nights game engine).

Development of technical expertise

Digital games design is a specialist field. Whilst the production team from CEDIR is experienced in the design, animation, programming and development of educational resources, the development of the game required the adaptation of a range of skills. The LEAP project provided an excellent opportunity for the staff to increase their skills and knowledge within an applied context.

Graphical design began as an investigation of appropriate 3D design software, one was needed that would match existing skill sets of the design team and be appropriate for the game design. Initially the team explored open source software, one potential product being Blender. However, time became a limiting factor. The use of Blender would have required a large amount of valuable production time consumed with familiarising the team with the interface and 3D modelling methods of the tools.

A final decision was made to use Autodesk's 3Ds Max 9. It aligned with existing skills and knowledge of the team, came within budget, and the license terms were more suitable than the alternative 3D software, including Maya. The decision to choose Bioware's Neverwinter Nights as the game engine influenced the final decision. A number of resources, which enable models to be exported from 3Ds Max into Neverwinter Nights, were available, including the plug-in called NWmax by Joco. As an outcome, the Graphic Designer has developed strong skills in low polygon modelling, texturing and animation in 3Ds Max and has broadened skills into the gaming design world. The Graphic designer worked closely with the programmer as the game developed.

Initially the programmer investigated a range of freeware and other games engines suitable for the project. Financial considerations were important and many game engines are expensive. Initially the team was interested in using a free-ware engine called PlaneShift. While this is a free engine, the license for this product is restrictive, only allowing for the use of the underlying code. This necessitates the compiling of code into a runnable package, the development of all

interface including art, game assets, character and creature models, environments, textures, objects, and equip-able objects, etc. Had this been chosen as a final engine, it would have consumed valuable time and budget, prior to development of any in-game activities, on basics, including getting the game to compile and run properly, and creating art and interface assets. PlaneShift also lacked a helpful online community of users. The team decided against using PlaneShift for the teaching game, however the final year students who had different evaluation criteria, decided to use this engine. The team continued investigating "out of the box" options.

The team also examined Neverwinter Nights to develop the game. Neverwinter Nights is a commercial 3D Role-playing game. It includes a tool-set called the Aurora Tool-set, which encourages the use of the game engine and assets to create new modules. There is a wide community of helpful users upon whom a developer may seek support to resolve issues with the game. There have been a large number of third party tools developed for use with Neverwinter Nights. There are many script packages that have been developed by others, which can be freely used. The overall price is affordable, at under fifty dollars for a single user. The game is a few years old, hence it does not require the latest hardware to run. There is an extensive range of art, assets and models that are available for immediate use or customisation.

Neverwinter Nights also comes with a script programming language that is a customised form of C++, it can be used to customise many game functions. Also, Neverwinter Nights can be leveraged for multi-player options and the storing of information is persistent server databases. A disadvantage of Neverwinter Nights is the cost of fifty dollars for a single license, when implementing across a large number of students this can be cost prohibitive. Some 'hard-coded' aspects such as spell systems and character races/species are difficult to change and/or remove. Other limitations relate to the game engine itself, some activities need to be customised to make them workable in Neverwinter Nights, for example a usable keypad to enter a code. This cannot be readily made but instead the developer needs a "work around solution" such as large buttons on the floor, which the character can activate by walk over.

Despite some of these limitations, it was decided that the Neverwinter Nights engine using the Aurora Toolset would be the best option. Of particular relevance was its ability to utilise pre-existing art, assets and systems. This enabled the developer to concentrate on developing activities, dialogue, learning outcomes and storyline for the prototype. Should the project prove successful base mechanics and art creation will be enhanced in later versions. Essentially, Neverwinter Nights engine is being used as a form of rapid prototyping environment, enabling the team to produce a usable and appealing prototype without having to worry about the more complex tasks such as the creation of a physics or collision system. For the team, LEAP has proved a very valuable developmental experience, increasing skills in programming, game development and the use of interacting elements such as game servers, databases, differing script functions and art assets.

The story

A great war was underway in the 21st century, during which a new bio-technological weapon was developed. The hope was to harness a newly discovered power, something to do with anti-matter, superstrings and quantum mechanics. Using this power, the hope was not to destroy the technologies and equipment of the enemy but to be able to control them from afar. Alas, this new technology, when deployed on a large scale, tore a rift in the fabric of the universe - which unleashed bizarre inter-dimensional powers, reshaping the world and destroying much of civilised society.... (In-game narrative)

In order to engage students with the game idea, a narrative or scenario concept was developed to involve students from the beginning. It needed to be a flexible one that would allow many different types of activities to be encapsulated within it. The limitations imposed by the Neverwinter Nights game development engine had also to be taken into account.

With these factors in mind, a starting scenario was devised that was set some years in the future after a global catastrophe, the exact details of which were long lost to those that lived in this future world. A great war was underway in the early 21st century, when a new bio-technological weapon was developed. Alas, this new technology, when deployed on a large scale, tore a rift in the fabric of the universe and unleashed bizarre inter-dimensional powers, reshaping the world and destroying much of civilised society. These powers merged with modern technology and the technology, with varying levels of sentience, turned upon its human overseers. And thus began the age of descent into darkness, ending the age of technological development.

Humanity fought back, and barely survived. In doing so they reverted to a pre-industrial society, fearing complex technology and living in enclaves that fend off periodic attacks by the machines. Despite surviving like this for some centuries, humanity was slowly losing the battle and needed to find a new way to defeat the machines. The idea emerged to bring forth a person from the early 21st century, just before the wars began and use their knowledge to bring the machines under control. Assuming that almost everyone in the 21st century knew how to program computers - since ‘everyone back then used and controlled them’ - they have transport an undergraduate university student, from the past, to unwittingly become the hope for all humanity.

Player in a representation of a university car park, when they are suddenly “teleported” to a prison cell in what turns out to be a far-future post-apocalyptic world.

In-game activities

The game’s play involves the students engaged in a quest, as they journey through the game they encounter tasks and activities to perform. Most of the tasks are designed to support the teaching and learning objectives, including problem solving strategies and academic content. As the student progresses in the game they practice core skills. The academic team has identified five activities to be created for the first prototype based on identified weaknesses within student performance:

1. Pass Argument activity
2. Algorithm Design activity
3. Systematic Traversal Algorithms activity
4. Tree Traversal Algorithms activity
5. Sorting Algorithms activity

The Pass Argument activity introduces and illustrates the concepts of passing arguments to functions, namely "pass by value" and "pass by reference" (see Image 1). In this activity, the player is presented with two objects that will need to swap positions. To achieve this, the player will need to assemble several lines of code and choose to pass the argument to a function either by value or by reference. The objects will swap positions if and only if the code is correct, that is if "pass by reference" is used. If the player chooses "pass by value" each object is firstly "cloned" and then the clones swap positions instead of the objects.

Image 1:

The second activity requires the player to encrypt a given number to generate the secret code that will open a door. The player has access to a WWII enciphering machine that can be used to encrypt any number except the given number. The main aim of this activity is to derive the encryption algorithm from the relationships between the numbers the player selects and the respective encryptions produced by the machine. Once the player has derived the algorithm he/she can use it to encrypt the number given and obtain the code that will open the door.

The learning objective of a third activity is to help students practice systematic traversal algorithms (using stacks or recursion) by searching for the exit of a maze. Because of the large number of possible routes, the chance of success is very slim if the player attempts to find the exit route randomly. This activity can be extended to involve multiple players, either as competitors or as collaborators in searching for a route to escape.

A fourth activity enables students to practice tree traversal algorithms. In drive X of a computer (the operating system is command-line MS

DOS), there are many directories, sub-directories, and files. Each sub-directory may have further sub-directories. The player needs to find a file named “password.txt”, and this file contains the password that allows the player to open a door. In order to locate this file, the player can only use the command “dir” to list the files and sub-directories in current directory. All parameters to “dir”, such as “dir /s”, have been disabled so that the player can only see the contents of the current directory. However, the player can use the “cd” command to enter a parent or child directory, but limited to a certain number of times. This limit will force the player to apply a tree traversal algorithm to systematically search the file system.

The fifth activity enables students to practice sorting algorithms. The player is presented with several identical opaque boxes each containing an object. The player can use an x-ray machine to see what is inside the boxes; however only two boxes can be inspected simultaneously. The player needs to sort the boxes according to the heights of the objects inside these boxes by swapping two boxes at a time.

Outcomes and future directions

The LEAP project was intended as a mechanism for improving outcomes for students undertaking two first-year subjects. Early in the project there were other recognizable outcomes. These include:

- A change in the culture of education at first year level with the development of a team of academics across three core subjects, working together to improve learning outcomes.
- Development of technical expertise within the university community, allowing application of the gaming approach to other discipline areas.
- Development of frameworks for games and specification of activities within the games that expand students’ problem solving skills.
- The opportunities for third year students to become involved in the design and implementation of educational games.
- The opportunities for third year students to provide feedback to first year teaching, by including their past experience in the subjects that they undertook.

As an offshoot to the broader strategy of QUALITY101 the project is guided by its cycle of evaluation. Future directions will be informed by the first year student responses to the games trial, both in terms of their motivation to use the games, improvements to their problem solving skills, algorithm design and programming. Improvements in grades are also desired outcomes.

The games have a natural extension to increasingly include activities that encompass the thinking that is characteristic of the three core subjects that it targets. With planned activities including pass arguments, algorithm design, systematic traversal algorithms, tree traversal algorithms and sorting algorithms the project hopes to improve key skills.

From an educational culture perspective the project has the capacity to create a broader team, working together rather than in isolation to provide a motivational and coherent approach to problem solving. The natural extension is to include mathematicians working in mathematics fundamental to the computing discipline. Ensley and Crawley (2006) for example approach Mathematical reasoning and proof in Discrete Mathematics with, puzzles, patterns and games. The mathematical problems treated are clearly also the province of computing; binary trees, graphs and trees, recursion.

The game will be ready for trial with first year students in the first semester of 2008. This will be evaluated and modified. The academics will then re-examine the learning outcomes and create appropriate activities. The game play and game narrative will be further refined and integrated into the game. If initial evaluation proves positive the team will seek an external grant from ARC or Carrick to support further expansion and development.

Conclusion

The LEAP project represents an innovative technique to support teaching and learning within the Computer Science subjects, the incorporation of digital game-based learning. It has evolved from a Faculty initiative to improve learning outcomes amongst first year students, QUALITY101. Whilst, only in its early stages the project is providing a focus for a team based approach in which academics teaching across a number of subjects are collaborating to support the development of solutions. The LEAP project aims to develop an alternative teaching strategy which can be utilised throughout the University.

Acknowledgments

This work was funded by the University of Wollongong Educational Strategies Development Fund (ESDF)

References

Alexander, S. & Hedberg, J.G., (1994). Evaluating technology–based learning: Which model? In Beattie, K. McNaught, C. & Wills (Eds.), Interactive Multimedia in University Education: Designing for Change in Teaching and Learning, (pp. 233-244). Amsterdam: Elsevier Science.

Bloom, B. & Krathwohl, D., (1956). Taxonomy of educational objectives: The classification of educational goals, by a committee of college and university examiners. Handbook 1: Cognitive domain. New York , Longmans.

Campbell, W. & Bolker, E., (2002). Teaching and Programming by Immersion, Reading and Writing, Frontiers in Education Conference, 1, pp. T4G-23.

Costelloe, E., (2004). Teaching programming the state of the art. CRITE Technical Report, 2004. Retrieved 10 April, 2008 from https://www.cs.tcd.ie/crite/publications/sources/programmingv1.pdf.

du Boulay B., (1989), Some Difficulties of Learning to Program, In Studying the Novice Programmer, Eds E. Soloway and J. C. Spohrer, Lawrence Erlbaum Associates, Publishers, Hillsdale New Jersey, USA, pp. 283-299.

Ebner M. & Holzinger, A., (2007). Successful implementation of user-centered game based learning in higher education: An example from civil engineering, Computers & Education, (49)3.

Ensley, D. & Crawley, J., (2006) Introduction to Discrete Mathematics: Mathematical Reasoning with Puzzles, Patterns, and Games, John Wiley and Sons.

Fasli M. & Michalakopoulos M., (2005). Interactive Game-Based Learning. Proceedings of the International Conference of the Association for Learning Technology, ALT-C 2005, September, Manchester, UK.

Garner, S. K., (2001). A Tool to Support the Use of Part-Complete Solutions in the Learning of Programming, Proceedings of Informing Science 2001, Krakow, Poland

Garner, S. K., (2003). Learning to Program Using Part-Complete Solutions, Proceedings of International Conference on Computer Based Learning in Science 2003, Nicosia, Cyprus.

Gee, J. P., (2003). What video games have to teach us about learning and literacy, ACM Computers in Entertainment, (1)1 pp. 1-4.
Gill, T. G., (2005). Engaging Introductory Programming Students with CGI, Decision Sciences Journal of Innovative Education, (3)1, pp. 177-181.
Henderson, P., (1986). Modern Introductory Computer Science, Proceedings of the Eighteenth SIGCSE Technical Symposium on Computer Science Education, St. Louis, Missouri, USA, pp. 183-190.
Jiménez-Peris, R., Patiño-Martínez, M. & Pacios-Martínez, J. (1999). VisMod: A Beginner-Friendly Programming Environment, Proceedings of the 1999 ACM symposium on Applied Computing, San Antonio, Texas, USA, pp. 115 - 120.
Kiili, K., (2007). Foundation for problem-based gaming, British Journal of Educational Technology, (39)3, pp. 394-404.
Linn M. C., & Clancy M. J., (1992). The Case for Case Studies of Programming Problems, Communications of the ACM, (35)3, pp. 121-132.
Masheshwari, (1997). Teaching Programming Paradigms and Languages for Qualitative Learning, Proceedings of the 2nd Australasian Conference on Computer Science Education, Melbourne, Australia, pp. 32 - 39.
Neller, T., Presser, C., Russell, I., & Markov, Z. (2006). Pedagogical Possibilities for the Dice Game Pig, Journal of Computing Sciences in Colleges, (6) pp. 149-161.
O'Neil, H., Wainess, R., & Baker, E. (2005). Classification of Learning outcomes: evidence from the computer games literature. The Curriculum Journal, 16(4) 455-474.
Porter, A. (2007). Summertime Math. Symposium on Learning Support in Mathematics and Statistics QUT, 19 - 20 July 2007.
Rajaravivarma, R,. (2005). A Games based approach for teaching the Introductory programming course, The SIGCSE Bulletin, 37(4), pp. 98-101.
Reeves, T. C., & Hedberg, J., (2003). Interactive Learning Systems Evaluation. Educational Technology Publications, Englewood Cliffs, New Jersey.
Reinfelds, J., Bergin, J., Rasala, R., Van Roy, P., Bredin, J., & Scott, K. (2003). Concepts First in Introductory CS Courses: Working Group Report ITiCSE 2003, (Computing science and engineering department, Research Report, 293288).
Robins, A., Rountree, J., & Rountree, N. (2003). Learning and Teaching Programming: A Review and Discussion, Computer Science Education, (13)2, pp.137-172.
Rogalski, J., & Samurcay, R. (1990). Acquisition of programming knowledge and skills. In Hoc, J., Green, T., Samurcay, R., & Gillmore, D. (Eds.) Psychology of programming, London: Academic Press. pp. 157–174.
Smith, P., & Webb, G. (1995). Reinforcing a Generic Computer Model for Novice Programmers, Proceedings of the ASCILITE, Seventh Australian Society for Computers in Learning in Tertiary Education Conference, Melbourne, Australia.
Yousoof, M., Sapiyan, M., & Kamaliddin, K. (2006). Transactions on Engineering, Computing and Technology, (12), March 2006, pp. 259-262.

Contact: jo@uow.edu.au

Cite paper as: Abrantes, J., Porter, A., Meyers, W., Stace, R., Susilo, W. Krishna, Zhou, Z.Q., Judge, D., Franks, R., Xia, T. (2008). CSCI: A LEAP into the future. In I. Olney, G. Lefoe, J. Mantei, & J. Herrington (Eds.), *Proceedings of the Second Emerging Technologies Conference 2008* (pp. 1-14). Wollongong: University of Wollongong.