AAEE 2012 CONFERENCE
Melbourne, Australia
www.aaee.com.au/conferences/2012/

Creating project-based learning in software

Matthew Joordens
Deakin University, School of Engineering
majoor@deakin.edu.au

BACKGROUND

Project Based Learning (PBL) allows students to learn by doing hands on work and thus also again practical skills. Therefore going to software simulation may seem like a backwards step. However, the opportunity presented itself to create a unit on Artificial intelligence (AI). This lends itself it a software approach. An added benefit was the opportunity to create a PBL AI unit for both Engineering and IT students.

PURPOSE

Can Project Based Learning be improved with dedicated, specifically written, software design?

DESIGN/METHOD

After introducing a dedicated software package the student marks and student feedback for the AI unit for a few years before the package introduction until the most recent year was analysed. In the student surveys, 2 key questions where examined: unit material quality and feedback to students. Student comments were also studied.

RESULTS

Student surveys gave a consistently high mark for unit's material and student marks seems consistent as well but favourable comments increased. The most significant result was that the students' thoughts about the project feedback had increased significantly, probably due to the inbuilt feedback system in the dedicated software package.

CONCLUSIONS

With an appropriate subject matter a dedicated software package can help students in PBL. Student enjoyment increased which, in turn, increased their motivation.

KEYWORDS

Project Based Learning, Artificial Intelligence, Software Simulation.

Introduction

In 2004 it became apparent that a new unit was required to teach robotic artificial intelligence (AI) for engineering students. This unit was required to complete the robotic aspect of the electronics degree at that time. To this end the IT degree of gaming was examined to see the AI unit from that degree could be used. It turned out that the gaming degree was also lacking an AI unit, therefore due to the similarity between robotic and game AI it was decided to write a unit to suit both degrees.

. The unit was to be a year three unit, being the last year for gaming and the penultimate for engineering.

Artificial intelligence in one form or another has been around since the 1930s (Russell & Norvig, 2003). At that time AI was being designed for robotics. The robot engineering and scientists of this time were dreaming of a day when robots T able to replace humans in menial jobs. To do this it was realised that a sophisticated AI system was required and in 1951 Marvin Minsky designed and built SNARC, a neural net machine (McCorduck, 2004; Russell & Norvig, 2003). It was also 1951 that the first chest and drafts games were built, all with their own AI engines (Metropolis, Howlett, & Rota, 1980). From that time on AI has begun to grow in leaps and bounds and is now an important consideration for anyone in electronics/mechatronics engineering area or in game design. Whilst game AI grew out of robotic AI and there are some differences between the two they are, in fact, very similar. This makes the unit for both robotic and game AI very feasible.

The Problem

The new unit had to be created to teach AI. Since learning about the theory of AI and applying AI really two different things, it was decided to teach applied AI. To do this it was decided that unit should be as interactive as possible. It should also assist the students as much as possible whilst letting them learn in an as rapid a manner as feasible. To further assist students the feedback system also had to be as fast as possible. What was really required was for instantaneous feedback or as near to that as practicable. In order to motivate the students it should be both challenging and engaging. The students should be able to have fun with it. Finally, it had to appeal to both engineering and IT students.

In order to meet the requirements of this unit, it was decided to use project based learning (PBL) (Hosseinzadeh & Hesamzadeh, 2012; Shahbodin, Yusoff, & Mohd, 2011; Zhao, 2011). PBL either creates or simulates a "real world" project. This project must be able to create a world which, while matching a real or imaginary scenario, would concentrate the students' attention on the AI aspect. The scenario should be able to engage the student, and motivate the student to study AI. With this in place the student is enabled to apply his theoretical knowledge of AI to an engaging environment that allows the student to learn AI without being too lenient or too harsh.

To create a PBL one must consider four attributes: content, activity, scenario and results (Zhao, 2011). The content of this covers the core concept and should match the student's current educational level. The activity covers the learning strategy. That is, how the students are going to go about learning the content. The scenario creates a special learning environment in which the students will work on the project. The results include not only the results that the lecturer sees and assesses but the results and feedback that the students themselves see.

One possible PBL method to teach AI is to use actual robots and program the AI into them so that they perform some task (Parsons & Sklar, 2004). This approach is very hands-on but was not used as it was not as applicable to the IT student and it required the off campus students to either come on campus to do the work or have access to the robots at home.

The students could write their own applications that use AI. These could be their own version of popular games such as Mine Sweeper or Asteroids (Becker, 2001). This approach was not considered as most of the students are not at the right level to do all their own code and it would have meant that a large portion of the class time is spent on non-AI related code.

Dedicated software was written specifically for this learning content. Running a piece of dedicated software has several advantages. The first is that the software package can be designed to be as interactive as possible with the students. It is sometimes also possible to design the software so that it is able to give the students instantaneous feedback or as close to that as possible. The software package-has the flexibility of simulating many different environments which can be tailored to meet the needs of the project based learning in question. A possible method was to create a software package that allowed the students to use prewritten controllers to control a game character (Hingston, Combes, & Masek, 2006). This approach creates the game environment which removes the overhead cost of writing non-AI related code but it also removes most of the AI code as well.

A large consideration is that an AI is almost always developed in software. It will either be run on a laptop, a project dedicated computer, supercomputer or an embedded microcontroller. Whatever the AI is for it will run on some sort of computer, and hence will be written in software. Gone are the days where AI was designed in analogue and digital circuits alone. Whilst it is possible to write AI as a digital circuit in a field programmable gate array (FPGA), this is still essentially a software design. Since AI is almost always written as software, then a software package would seem to be the ideal medium to teach it. The final reason why a dedicated software package was chosen was because the lecturer and the programmer were one in the same person. It is vitally important when a software package is being written for teaching that the teacher and the programmer have a close affiliation. Normally this is because the teacher understands the teaching method to be used and the programmer understands how to write the code. If a teacher describes program to a software programmer and that leaves the programmer to it, the result will be a software package that does not meet the requirements of teaching it subject (Shiratuddin, 2011). Fortunately as both the teacher and the programmer were the same person the affiliation between them was very close indeed!

The software package was written in C# using MS Visual Studio. The basic code for a dynamic linked library (DLL) was provided to the student. The students create their AI in the DLL using c# in MS Visual Studio or MS Visual Studio Express.

Methodology

The Content

This PBL was an introductory level of robotic and game AI theory and application. The software package in this PBL would concentrate on the application and a few lectures covered the theory. The theory was covered in the first four to six weeks of the unit. The remaining time would be used in applying AI theory in the software package.

Activity

The software would simulate some sort of "real world" system. Students were able to operate the system on their own. That is, they would control the AI agent themselves. Once they have explored how the AI agent (either a robot or a game character) had to be controlled in order to complete a given goal they would write the AI code to control the agent itself. This allows students to try different strategies, methods and techniques, and see how they would reach the goal of the system. With that knowledge they can start to apply AI to do the same job. This carries two benefits: to be able to work out the best AI strategy, and to explore the difference between their own intelligence, and AI.

Scenario

In order to meet PBL requirements it was decided to create 2-D robot simulator but to disguise it as a game. And so the game Space Chase was born. (Figure 1)

A game used for teaching is not new. Games or ICT solutions have now been used in various educational programs (Dickinson & D., 2011; Engineers-Australia, 2012; Hirumi, 2008; Peixoto, Possa, Resende, & Padua, 2011; Shahbodin, et al., 2011; Ting & Hao, 2011). The scenario is that a scout spaceship (the AI agent) is caught in a bounded area of space and is surrounded by hazards, obstacles and resources. To add to its plight it is being hounded by drone spaceships that will head straight for it. If these drones touched a scout, they will both be destroyed. The scout has no weapons of its own, except for an Energy Shield.

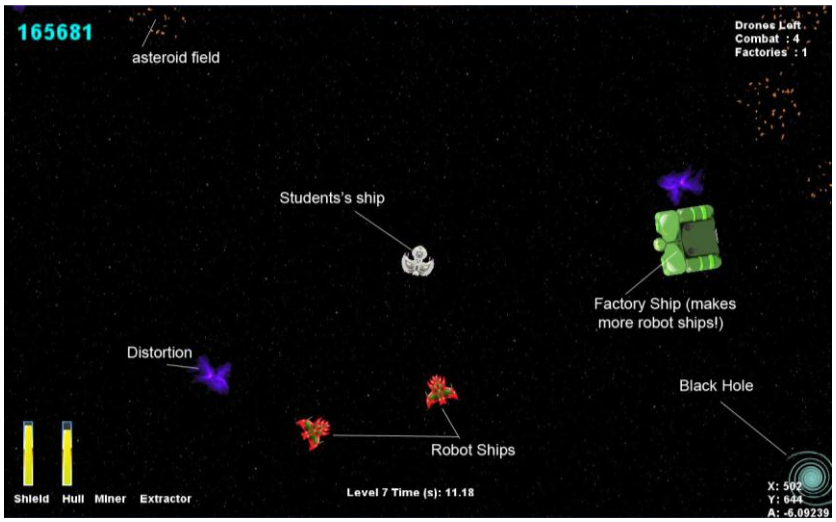

Figure 1 Space Chase game

This particular scenario is comparable to a robot in the real world. Terrestrial robots essentially work on a 2-D plane, whether that is a warehouse or a factory floor, or the open fields, be they as diverse as a farm or a battlefield. The AI agent (robot) was surrounded by obstacles, such as rocks, trees or animals. It could also be surrounded by hazards, such as ditches or mines and it could have resources such as a recharging socket or home base. In Space Chase these hazards, obstacles and resources are represented by space distortions, asteroid fields and black holes. Some hazards, such as on a battlefield, can be of a threatening nature. In the game these are the drone spaceships and are added not to concentrate on a military aspect, but to create a challenge for the game itself.

The object of the game is that all the drone spaceships must be destroyed to move onto the next level of the game. As levels progress more drones are added and less objects and resources are provided. The better the AI code the greater number of levels the scout spaceship will survive through. The scout spaceship must lure the drone spaceships to destruction without, of course, itself hitting a hazard or being hit by drone ship. The scout spaceship can hide behind a black hole to lure a drone spaceship into it, or it can lead the drone spaceship through asteroid fields until it has suffered enough damage to be destroyed. The scout spaceship will of course also be destroyed by a black hole and must therefore avoid them itself. It can also be damaged by moving through asteroid fields but the asteroid fields can be mined by the Scout spaceship to repair itself. The scout spaceship also has an energy shield that can be used as a last resort if a drone spaceship gets too close. It is able to re-energise its shield by extracting energy from space distortions.

Results

The final assessment of the students' achievements in this project examines how efficiently the AI performance is and how many levels the AI can survive in the game. But apart from

that, the students themselves must be able to see how well their AI is performing. One method is to break the operations the AI must perform into progressive tasks. Space Chase has six tasks to help the student gradually develop the AI code to play the final game.

These tasks are:

1. Locate a black hole. Since a black hole is able to attract a ship from a distance, and to enter them means sure destruction, that it is appropriate that as the Scout ship must move around its environment to locate any black holes in its vicinity so that it can avoid them. Hence this is an appropriate starting point.

2 Mine an asteroid field. Now that the AI can see black holes, this task concentrates on avoiding those black holes in a search for asteroid fields. Once it finds an asteroid field it must be able to get close enough to the field to mine it so that its hull may be repaired without actually entering the field to be damaged by it. This task works on the ability to avoid objects and move the ship precisely.

3. Extract energy from space distortions. This task completes the ability of the AI to navigate its environment without being damaged and to be able to use all the resources that are provided.

4. Evade a drone. This task introduces a drone spaceship. The drone's own AI is simply to head straight for the scout spaceship. With the introduction of the drone the AI must now be able to evade the drone so that scout spaceship is not destroyed.

5. Destroy a drone. This task now concentrates on either being able to lead a drone through asteroid fields or into a black hole without the scout spaceship being destroyed itself.

6. Destroy a factory drone. The factory drone is an added complication in the game. Whenever it collides with a drone spaceship it will create three new drone spaceships. Also it will not head for the Scout spaceship unless it is within a certain range of the scout. This task concentrates on having to find a factory drone before leading it, whilst remaining within its following range, to its destruction.

Once the student is able to write code to complete the six tasks, then the students AI, with a few more tweaks, should be able to play the game.

The assessment for the unit itself consisted of: A report of the AI methods possible and used, (testing the theoretical knowledge), the feedback score from the package and a final reflective report.

Feedback

While writing AI code to perform the different tasks, the student is able to see, as the game or task plays out, how well the AI code is performing. This is a good form of feedback for the student, however the software goes beyond this. The software has been designed to analyse the students AI performance. It uses a log of each task and the levels of the game to let the students know just where the AI is doing well and where it is falling down. An example is given in Table 1 for task 3, extracting energy from a space distortion.

Table 1 Assessment criteria for task 3

Sub-task	Performance Measure	Points
Mine some resources	Locate an asteroid field and mine resources from it to fully repair the hull	2
Locate an energy field	Appropriate sensor is active and detects presence of space-time distortion within sensor range	1
Collector	First activation of energy collector - out of range - In range	1 - 2

activation		
Energy collection	Collect enough energy to fully refill the Shields	2
Shield activation	Successfully activate shields when full	1
Survivability	Awarded if the ship survives all iterations of the task	2

One can compare this with a sample log output for task 3. The extract from the log for task 3 is in the Table 2.

The student is able to examine the log and to see where, and which activity, requires a more concentrated examination. The log extract seen here shows a perfect score, as it should since it results from the Lecturer's own AI solution! Not only is the student able to see how the AI code is progressing towards a task or goal, the student can see just what sort of assessment he should get as the log itself is used by the assessor.

In addition to this, the software package also allows the students to create their own log. They can create a comma separated variable (.CSV) log which contains any variables, such as, positions, velocities, the locations, as they desire and log them every 60th of a second. This allows the students to tailor make their own feedback system.

Table 2 Extract of feedback log

```
Task 3, iteration 0
Seed Used: 161423279
Repaired Hull          : 2
Detect Distortion      : 1
Extractor Activated    : 2
Shields Full           : 2
Shields Activated      : 1
Total for this iteration       : 8
Task 3, iteration 1
Seed Used: 136652495
Repaired Hull          : 2
Detect Distortion      : 1
Extractor Activated    : 2
Shields Full           : 2
Shields Activated      : 1
Total for this iteration       : 8
Survived all iterations        : 2
Task 3, Overall
Total for Task 3 (Sum of Score/Iterations): 10
```

Random but Fair

One concern was that all the environments of the tasks and game levels were created randomly. This could mean that some students would get easy environments and some more challenging environments. This would impact on assessing the students as there would not be an even playing field. To overcome this, a seed number is used. Random numbers in software packages are not really random but only pseudorandom. That is, they start with a seed number and generate the next random number from that seed using an algorithm. Therefore the series of seed numbers can seem random but are defined by the first number. Most software packages use the current time as the first seed number as the current time, to the millionth of a second, is not easily determined by the user. In this package the starting seed number can be defined. Therefore the assessor, when running each student's code, can enter the same seed number which is recorded by the log. This means that each student will have the same set of random environments used for

assessment. This also means that the students will not know what seed the assessor will use, and hence, will have to write their AI code for any random environment.

Results

In order to see the effectiveness of the software, the student surveys of this unit were examined since 2006. To examine the history a few more facts are required. In 2005, a final year engineering student was hired to write the first package. He was provided with art work for the game the rest was left to the final year student. Although program worked, it was not playable by the student and did not give feedback to the student. In 2007 a new software package was written by a colleague from IT who taught gaming. This package was still not playable by the student, but broke down the goal to the six tasks. The feedback system described herein was still non-existent. Whilst this code was written in the correct manner for a full-blown game, it was too complicated for the engineers who took this unit. This code was used in 2007 and 2008 and the unit was run by the colleague in 2007. In 2009 the package was rewritten in its current form to allow the students to concentrate on the AI aspect alone. This package was tweaked in the two following years to remove any bugs. The new software package was fully playable by the student, and this included music and sound effects. This should not be overlooked as sound is another level of engagement to the package (Ting & Hao, 2011).

To examine any possible improvements the software package provided, Student feedback data was examined. In the student survey analysis 2 questions were considered: the “quality of the course material” and “helpful feedback”. This data is available on the Universities website. The results are in Table 3.

Table 3 Student responses (scores out of 5)

Unit Survey Question	2006	2007	2008	2009	2010	2011
Course materials were of high quality	3.25	3.10	3.80	3.75	3.88	4.47
Teacher gave me helpful feedback	3.83	3.80	3.83	3.94	4.25	4.53

Looking at the course material quality we can see that from the table 2006 was a reasonable 3.25 out of 5 when the final year student written package was used. This score dropped in 2007 when the games lecturer took over for that year, possibly due to a more complicated software package for engineering students that greatly outnumbered the IT students in the unit. 2008 saw a rise of the material quality additional material was created to aid the engineering student. In 2009 when the final package was introduced we see a slight decrease in the score followed by two increases to 4.47 out of 5 in 2011. Combining this with student comments this increase is attributed to the new software package and the bugs being fixed over 2010 and 2011. Most of the bug fixes were to correct initial errors in the code. For example, the feedback on the scout ships rotation was incorrect and the students could not rotate the ship without problems.

The helpful feedback survey question shows a consistent feedback from 2006 to 2008. In this time the software itself did not give any feedback and so this feedback is due to the lecturer alone. From 2009 to 2011 we see a steady rise in the students’ evaluation of feedback. As the lecturer feedback methods were not changed from 2008 this increase can be attributed to the feedback the software package itself gives.

The student comments on this unit tell a similar tale. The following selected comments, which are all positive, shows how the students felt. It must be noted that there was not a single negative comment on the software.

“As a Software Engineer working professionally, I recognize the outstanding effort and high level of competency that has gone into the Software that Matthew created for use with this

subject. When I realized the scale of this exceptional contribution Matthew has made by creating this Unit and the Software used, I was overwhelmingly taken back."

"Of all my peers who develop software professionally, and who have sat a unit of AI in their CompSci degrees at other universities, this unit's content appears to be far more advanced."

"Having a project that was interesting made it easy to stay focused."

"Love the practical approach learnt heaps by writing code. Just disappointed that other subjects cut into the time to write code for this subject."

"Space chase is brilliant, an incredible piece of work from an obviously very smart person"

"[I liked] the astonishingly good software that has been created for this unit. The degree of hands on time involved. The relevance and quality of topics included."

"[I liked] its approach to learning."

The comments speak for themselves and the lecturer could not have asked for any more if he want to. The comment on the advanced nature of the unit is interesting as the unit is no more advanced than any other introductory AI units. The difference is that this unit applies the AI theory in PBL.

Conclusion

We see then that project based learning, in the area of AI, using dedicated software with built-in feedback allows students to learn whilst being both challenged and engaged. The feedback system allows the students to progress, almost independently of the lecture, at their own pace. At the same time the lecturer, with the software performing the assessment analysis has more time to spend with the student to guide student in any areas that are lacking. The game itself motivates the students to excel in a challenging area whilst at the same time letting the students enjoy themselves in a project that, while it is fanciful, can be related to real world scenarios.

References

Becker, Katrin. (2001). Teaching with games: the Minesweeper and Asteroids experience. *J. Comput. Small Coll., 17*(2), 23-33.

Dickinson, J.K., Woodard, P. R., Canas, R., Ahamed, S.S. & Lockston,, & D. (2011). Game-based trench safety education: development and lessons

learned. *Journal of Information Technology in Construction., 16*(Special Issue Use of Gaming Technology in Architecture, Engineering and

Construction.), 119-134.

Engineers-Australia. (2012, March 2012). Providing training through computer games. *Engineers Australia, 84*.

Hingston, Philip, Combes, Barbara, & Masek, Martin. (2006). Teaching an Undergraduate AI Course with Games and Simulation

Technologies for E-Learning and Digital Entertainment. In Zhigeng Pan, Ruth Aylett, Holger Diener, Xiaogang Jin, Stefan Göbel & Li Li (Eds.), (Vol. 3942, pp. 494-506): Springer Berlin / Heidelberg.

Hirumi, A. & Stapleton, C. (2008). Integrating Fundamental ID Tasks

with Game Development Processes to Optimize Game-Based Learning. In Miller C. (Ed.), *Games: Their Purpose and Potential in Education* (pp. 127-160). New York: Springer Publishing.

Hosseinzadeh, N., & Hesamzadeh, M. R. (2012). Application of Project-Based Learning (PBL) to the Teaching of Electrical Power Systems Engineering. *Education, IEEE Transactions on, PP*(99), 1-1. doi: 10.1109/te.2012.2191588

McCorduck, Pamela. (2004). *Machines Who Think* (2nd ed.): A. K. Peters, Ltd.

Metropolis, N., Howlett, J., & Rota, G.C. (1980). *A History of Computing in the Twentieth Century* New York: Academic Press

Parsons, Simon, & Sklar, Elisabeth. (2004). *Teaching AI using LEGO Mindstorms*. Paper presented at the AAAI Spring Symposium.

Peixoto, D. C. C., Possa, R. M., Resende, R. F., & Padua, C. I. P. S. (2011, 22-24 May 2011). *An overview of the main design characteristics of simulation games in Software Engineering education.* Paper presented at the Software Engineering Education and Training (CSEE&T), 2011 24th IEEE-CS Conference on.

Russell, Stuart J., & Norvig, Peter. (2003). *Artificial Intelligence: A Modern Approach* (2nd ed.): Prentice Hall.

Shahbodin, F., Yusoff, M., & Mohd, C. K. N. C. K. (2011, 26-28 Sept. 2011). *ICT + PBL = holistic learning solution: UTeM's experience.* Paper presented at the Digital Information Management (ICDIM), 2011 Sixth International Conference on.

Shiratuddin, M. F. (2011, 14-16 Nov. 2011). *Integrating computer game-based learning into construction education.* Paper presented at the Information Technology and Multimedia (ICIM), 2011 International Conference on.

Ting, Wei, & Hao, Zheng. (2011, 24-25 Sept. 2011). *Sound Effect of Physical Engine in Game Design.* Paper presented at the Information Technology, Computer Engineering and Management Sciences (ICM), 2011 International Conference on.

Zhao, Yan-yan. (2011, 3-5 Aug. 2011). *Design and development based on PBL online course.* Paper presented at the Computer Science & Education (ICCSE), 2011 6th International Conference on.

Copyright statement