The Game's the Thing: Levelling up from Novice Status

Diane P. McCarthy
Christchurch Polytechnic, Institute of Technology
diane.mccarthy@cpit.ac.nz

Rob Oliver
Christchurch Polytechnic, Institute of Technology
robert.oliver@cpit.ac.nz

ABSTRACT

Quality computer engineering education is integral to the recruitment, retention, and employment of quality software engineers, as part of enabling a greater uptake of Science, Technology, Engineering and Mathematics (STEM) careers. The introductory programing course DICT440 uses Build Your Own Blocks (BYOB) and the team creation of a game, Theseus and the Minotaur, to teach introductory programing principles and skills. This paper argues that creativity is essential to innovation. Digital Games are being increasingly used in education and training internationally, as well as specifically in computer education. Aotearoa-New Zealand ITPs need to position themselves positively to leverage the creativity and motivation of software engineering students who are experienced gamers by developing games as part of teaching and learning software engineering. Computer game development courses can be developed collaboratively in a multi-disciplinary team, using appropriate learning theory, across ITPs in second and third year degree courses, in conjunction with regional game companies, alongside core business applications.

Keywords: novice software engineers, STEM, game based learning (GB-L) in software engineering, collaborative game development

1. INTRODUCTION

Effective pedagogy for novice software engineers has been the subject of much practitioner research this century. In Aotearoa New Zealand, an ongoing ITP collaborative international research project, BRACElet, researched effective curriculum delivery to novices for 13 years (Clear et al., 2011). By analysing naturally occurring data, it was concluded that novices think in concrete rather than abstract terms about algorithms. Ability to perform code tracing analysis of executed code foreshadows being able to reason about it. Explaining problems in plain English and writing suitable algorithms correlated positively. It was suggested that a possible hierarchy of tasks existed, with programing concepts, explaining in plain English, and code tracing analysis forming three important steps, from novice to intermediate level. Constructing algorithms requires a minimum level of code tracing skills. The better predictor of good performance is a combination of tracing and code explanation.

Victoria University of Wellington (Miliszewska, Venables & Tan, 2008) conducted research to stem the dropout rate of first year students in its computer science degree, which included computer programing as a compulsory course. A common problem in New Zealand universities, programing was seen as a very challenging skill set to master. Learning outcomes were improved by using the strategy of 'befriending'. Using analogy, collaboration, mentoring sessions and electronic support, the course became more user friendly and student centred, rather than content centred.

Similarly, Saito & Yamauara (2013), in a Japanese setting, compared three novice tertiary students exposed to a top down approach, with C sample code to copy (TDA) with three novices taught through the traditional bottom up approach (BUA). Findings show no noticeable improvement in algorithm construction, but TDA students exhibited greater understanding of the work of others, as they had to read and understand the samples and understood reuse. While the cohort is small, Hu, 2008, similarly explored this approach in a New Zealand ITP setting.

Drawing on constructivist and social learning theories, Kordaki (2010) used geometrical objects to teach C to high school level novice programers (LECGO: Learning Environment for programing using C using Geometrical Objects). This approach considered learning process, content, and the ways that students behaved when undertaking programing tasks. Many different kinds of external examples were used, such as problem solving activities through drawing simple geometrical shapes, using hands on experience and feedback to correct oneself, and multi layered multi-media to participate in these activities holistically. The pilot study had nine participants from 12 grade (18 year olds). Each student used pen and paper, TURBO C and LECGO to perform three sets of tasks, with the best results in the LECGO pedagogical approach.

2. GAME BASED LEARNING (GBL)

Parallel to this kind of practitioner research is the innovative use of Digital Games in education. Matthews (2014) challenges all educators to take stock of how to make teaching and learning more relevant in the 21st Century. Using the analogy of the paradigm shift in movie making from silent movies to talkies, and the innovations that followed in theory practice and technical requirements, Matthews challenges us to "Flip the Model." We need to look at better ways of integrating the interests and experiences of our students into our pedagogy as a powerful way of engaging with them.

One such approach is Game Based Learning (GBL). Digital Games are being researched and incorporated across a wide variety of fields and levels of education and training, including corporate and government settings and from early childhood education to tertiary education. For example, simulation games are being used to build mental models about entrepreneurship for novice business students (Palmunen, Pelto, Paalumaki, & Lainema, 2013). Explaining the rules of a game stimulates meaningful conversations in early childhood education, observing gender differences (Sorsana, Guizard & Trognon, 2013). Aboriginal youth have successfully learned through "Guitar Hero" in remote schools in Outback Australia (Jorgensen & Lowrie, 2013) and the US Military drills

This quality assured paper appeared at ITX 2014, incorporating the 5th annual conference of Computing and Information Technology Research and Education New Zealand (CITRENZ2014) and the 27th Annual Conference of the National Advisory Committee on Computing Qualifications, Auckland, New Zealand, October 8-10, 2014. Mike Lopez and Michael Verhaart, (Eds).

soldiers in 3D game environments in the use of Patriot Missiles (Sostak, 2012). Such uses can be applied in teaching programing.

GBL is commonly used to improve learner performance. Digital Games are prevalent in online learning. Tsai & Fan (2012) trace the increased incidence of social science research in this field, 2008 - 2012. Similarly, Hwang & Woo (2012) report increased educational research on digital games from 2001 -2010. Helpful advice is provided for policy makers to encourage governments to invest in these practices. Mayo (2007) suggests that the use of games to promote STEM careers from early childhood education through to senior high school, could overcome the stereotype that these kinds of careers are boring.

Games are being developed for instructional purposes. Westerna, Nadolski, Hummel & Wolpereis (2008) set out a framework for streamlining the development of a useful game based scenario in an educational setting. Serious games are inherently complex, requiring a high level of development. By simplifying the game concept, technical and practical requirements, projects are more cost effective and within the scope of computer education. Basic elements of the static game configuration are identified as well as the game dynamics, showing changes in various game components over time. The system architecture of building tools was defined, as well as a set of design principles, about game structure, feedback and representation, reducing complexity. This work foreshadowed many of the game engines which now underpin game development, such as Unity, based on the same principles.

3. SOFTWARE ENGINEERING & GBL

Games are being currently used to teach programing. Hu (2008) explored the use of games with novice programers in a New Zealand context. Hu's review of the literature suggested that while game programing was complex, integrating game scenarios into programing education "remained an uncharted area," Hu, 2008, p.28. Some practitioners assumed that programing games were too complex to be used at novice level, (Distasio & Way, 2006, in Hu, 2008). Klassen, 2006, in Hu, 2008, concluded after three semesters of using Alice, as an introductory language for novices, that building 3D games had required too much time and experience. However, others enabled novice students to program console games in Pascal and C++, (Cliburn, 2006; Feldgen & Clua, 2004, in Hu, 2008).

Hu (2008) developed simple games, such as Dice, in VB, where the learning facilitator shows sample algorithms, which students then modify and reuse to make variations. This is similar to Saito & Yamauara's TDA approach, which emphasizes reiterative reading, understanding and reuse of sample algorithms. Concepts were incrementally introduced, which increased the functionality of the game. Anecdotal evidence from class observation suggested that students were more engaged than before the intervention, and that this effect improved on task behaviour with business case scenarios (Hu, 2008).

GBL explores the programing of advanced computer architectures to execute applications, in high performance computing (HPC) and complex, inter-networked, distributed systems. These games provide examples of concurrent, parallel, and distributed manipulations and processing of sequences. For example, manipulation games include "pattern matching, recognition, alignment, transformation, and evolution of sequences," Amenyo, 2012, p. 351. Scrumia, a game to teach SCRUM as a software development strategy, has been well received by novice programers, who find it engaging, through evoking of real world situations, encouraging social interaction, and keeping students on task (von Wangenheim, Savi & Borgatto, 2013).

Wang & Chen (2013) researched the use of interactive games with 115 junior high school students in Taiwan, to teach introductory programing concepts. They found that while learning to write algorithms was difficult, engagement and performance were enhanced when students selected an experiential activity that best fitted their individual game play preference. Those who selected the challenging game strategy had higher flow experiences than those who selected the matching/challenging strategy. Those who mismatched their preference still performed better in the challenging strategy.

Game development frameworks facilitate learning in software engineering (Wang & Wu, 2009). Using such frameworks to teach novices has positive pedagogical outcomes. These include, greater participation of students, variety in how content is delivered, and stimulation of student interest in software engineering. By developing a computer game, students learn software architecture. This was successfully integrated into teaching and learning programing at Norwegian University of Science and Technology (NTNU).

Tinkering is another approach that can be used with novice programers, which is synonymous with "playing around" with programing. It is a concept that has arisen out of the field of learning analytics. Berlow, Martin, Benton, Petrick Smith and Davis (2013) used this learning concept to encourage novices to program. Novice students explore, tinker, and then refine their programs, which have been empirically proven to be a valuable means of acquiring these skills.

Uncertainty, a concept in games, and its effect on learning outcomes, has been explored by Ozcelik, Cagiltay & Ozcelik, (2013). A database concept was taught using a game-like tool, with one version incorporating uncertainty, and the other with no uncertainty. It was found that uncertainty increases learning and was positively associated with motivation.

Li (2012) explored enactivism, with teachers creating digital games for education purposes in a graduate course**.** Teachers were asked what they perceived as affordances and constraints as they designed, developed and shared games for delivering content. In particular, teachers found that their experiences of designing and building games helped them to rethink their teaching approaches and practices.

4. TIME FOR A CHANGE

Hu (2008) takes great care to emphasize that the games enable software engineering to be learned, rather than the other way around. Given that gaming has now been a field for over thirty years, perhaps it is timely to reverse this assumption, and enable experienced gamers, as novice programers, to create their own games, using software engineering as the means. Game engines such as Unity enable the complex game development tasks to be managed, and agile methods such as SCRUM lend themselves to developing game modules. Anecdotally, when given the opportunity, second year diploma and degree students at xxx have shown tenacity, creativity, and commitment to create games in BYOB, Minecraft, C and Unity, as their elected team project in professional practice. Half of these students had initial software engineering training as diploma students in DICT440: Introduction to Programing, which uses making a game as part of its pedagogical approach.

5. FESTINARE LENTE

However, incorporating games into novice computer courses is not a panacea, as Wouters, van Minwegen, van Oostendorp, and van der Spek (2013) warn. Little evidence had been shown to support the claims of greater motivation and skill as such. Using meta-analytical techniques, it was found that serious games are more effective in terms of learning ($d = 0.29$, $p < .01$) and retention ($d = 0.36$, $p < .01$), but not in terms of motivation ($d = 0.26$, $p > .05$) compared to traditional methods of instruction. What made the overall difference was the use of other instructional methods which supplemented the use of serious games, student/players learning in group settings, and being exposed to multiple training sessions.

Similarly, including gaming in a course does not necessarily diversify the uptake of computer programing by underrepresented groups such as women, black Americans, and Latino/Latinas. Where games are offered as part of a computing course, such as several universities in the United States, diversity was seldom mentioned and underrepresentation just as prevalent (McGill, Settle, and Decker, 2013). Could a game development project help us learn Te Reo, better understand our history and enable more meaningful partnerships with our local iwi? Be more inclusive for students with disabilities and cultural diversity- perhaps?

Finally, Wu, Hsaio, Wu, Lin and Huang (2012) warn that educators seldom consider the pedagogical approach and learning theory that they are applying when using GBL. Of the 658 projects surveyed, only 13.83% considered these aspects. Given that the way we see the world as educators shapes our assumptions, practices and student learning outcomes, it is suggested that learning theory is equally considered and underpins GBL in computer education.

6. A NEW ZEALAND GBL TALE

DICT 440, Introduction to Programing, was created in 2011, in response to the need to incorporate software engineering into the two year Level 5 diploma course at Christchurch Polytechnic, Institute of Technology. The course is taken in the second semester of the two year diploma programme. Like many novice courses, it does not have a specific pedagogical approach. Reflecting on the course delivery in Semester 1, 2014, the writers as an enrolled student and tutor delivering the course, have this tale to share and observations to make, within the context of literature review above.

In a class of 28 students, there were eight self- managing groups, ranging in size from three to four students. Each week, students are given samples of Build Your Own Blocks (BYOB) code, and are encouraged to write simple algorithms, for new business cases, based on those samples. Students may submit these and then feedback is given. In this way, students are encouraged to build up a library of code samples that can be referred to when making a game and in the final business case study two hour examination.

The course assignment, to replicate a simple online game, in BYOB, used the Program Development Life Cycle method of game development. In the first part, students set out the game specifications for the Theseus and the Minotaur Game. This included developing the avatars of the characters, the stage, as a maze of ten squares, and documenting the moves that needed to be replicated. This included written and visual representation of the moves.

Some groups had members undertake all of the assignment, and then combined the best parts to be submitted for a grade. Others had clear designation of various tasks such as written specifications, analysis of the game moves, and the multi-media aspects of crafting the avatars, maze/stage, and backgrounds (scenery). Another pattern of work flow that was observed was that expert gamers, with some considerable intuitive grasp of software engineering, tended to dominate the group and work outputs.

In both approaches, tinkering with the software became a major source of learning to write algorithms for the game. Students did not necessarily make a strong connection between the sample games that were mastered weekly, and the game in the assignment. Instead, they tinkered with the software, and through trial and error, wrote code which performed the moves of the game they were replicating. Thus, the pseudo code construction phase did not necessarily reflect the final structure of the game program.

Further, it was apparent in class workshops, where students demonstrated their works in progress, that they imitated code structure and shared sample code to assist each other. Where any students became uncertain of how to construct the code, leaders would offer assistance in the form of attending their weekly group meetings, and demonstrating their code and game moves, in the workshop, so that the other groups could imitate, tinker and create unique iterations of the samples they had observed. Thus collaboration happened within groups as well as between groups.

Scaffolding was present through the use of the original game to replicate, and the potential use of sample code, from weekly sessions. But it was mostly used through the sharing and observation of sample code being developed within the game. Peer support leaders, who had successfully completed the course in the previous semester, with A grades, demonstrated their games to current students, to provide examples of how the original game could be replicated.

Finally, many students began writing the code (or code segments) for the game right from the first phase of the assignment. They seldom, if ever, went online to the BYOB web site to find code samples. Students preferred to create their own. In some cases, the pseudo code was reverse engineered from the actual code. The third phase, where students were required to code, based on the pseudo code, was actually undertaken in many cases in the first and second phases of the four part assignment, and then tested and debugged for final submission. Students were able to give feedback on the contribution that each member made to the overall project in the form of a report in the fourth and final phase of the assignment.

Thus the most common method adopted by the novice software engineers in this course was to develop a prototype using an agile method intuitively, and work top down by tinkering, collaborating and overcoming uncertainty through peer support.

These reflections on our experiences and observations need to guide a formalised research project. An appropriate pedagogy needs to underpin this research, so that the student responses are informed by learning theory, contributing to the field of best practice in computer education.

7. CONCLUSION

Many of our students are expert level gamers, with 18-35 years of experience behind them, and yet are often novice programers, given the preoccupation of the compulsory sector of our education system to frame IT education as a tool. We need to couple this experience into GBL, alongside our other successful approaches and practices, to enable relevant, creative and entrepreneurial outcomes for these students.

Given the strengths of framing pedagogy in the diverse yet complementary strands of behaviourism, cognitivism,

humanism, and constructivism, perhaps we have been unjustifiably agnostic for too long. We can explore ways of embedding learning theory into the foundations of GBL.

And rather than maintaining our IT/ ITP silos, we can consider including specialists from other academic fields, as well as from within the game development industry. Finally, we can collaboratively develop GBL processes and serious game outputs for local, national and global use for the social, economic and educational wellbeing of Aotearoa New Zealand (Wu, et al, 2012).

8. REFERENCES

Amenyo, J.T. (2012). Playable Serious Games for Studying and Programing Computational Stem and Informatics Applications for Distributed and Parallel Computer Architectures. *Journal of Educational Conputing Research.* 47(4) 351-370.

Berlotti, F., Kapralos, B., Lee, K., Moreno-Ger, P., & Berta, R. (2013). Assessment in and of Serious Games. An Overview. *Advances in Human Computer Interaction.* P1-11.

Berland, M., Martin, T., Benton,T., Petrick Smith, C., & Davis, D. (2013). Using Learning Analytics to Understand the Learning Pathways of Novice Programmers. *The Journal of the Learning Sciences.* 22(4) 564-599 DOI:10 1080/10508406.2013.836655

Clear, A., Whalley, J., Robbins, P., Philpott, A., Eckerdal, A., Laakso, M-J., & Lister, R. (2011). Report on the Final BRACElet Workshop:Auckland University of Technology 2010. *Journal of Applied Computing and Information Technology.*15 (1) 1-7.

Hu,M. (2008). Using Game Scenarios for Teaching Novice Programmers. *NZ Journal of Applied Computing and Information Technology.* 12(1) 27-29.

Hwang, G-W., & Wu, P-H. (2012). Advances and trends in digital games based learning research: a review of publications in selected journals from 2001-2010. *British Journal of Educational Technology*. 43(1) pE6-E10. doi:10:1111/j1467-8535 2011.01242x.

Jorgensen,R., & Lowrie, T. (2013). Both ways strong: using digital games to engage Aboriginal learners. *International Journal of Inclusive Education.* 17 (2) 130-142. doi:10.1080/13603116.2011.605912

Kordaki, M. (2010). A drawing and multi-representational computer environment for beginners of programming using C: Design and pilot formative evaluation. Computers and Education. 54(1) 69-87. doi: 10.1016/jcompedu.2009.07.012.

Li, Q., (2012). Understanding enactivism: a study of affordances and constraints of engaging practising teachers as digital game designers. *Educational Technology Research and Development.* 60(5) 785-806. doi: 10.1007/s11423-012-9255-4

McGill, M.M., Settle, A., & Decker, A. (2013). Demographics of undergraduates studying games in the United States: a comparison of computer science students and the general population. *Computer Science Education.* 23(2) 158-185 doi: 10.1080/08993408.2013.769319

Matthew,B. (2014). Flip the Model: Strategies for Creating and Delivering Value. Journal of Academic Librarianship. 40(1) 16-24

Mayo, M.J. (2007). Games for Science and Engineering Education, *Communications of the ACM.* 50 (7) 30-35

Miliszewska, I., Venables, A., Tan, G. (2008). Improving Progression and Satisfaction Rates of Novice Computer Programming Students through ACME- Analogy, Collaboration, Mentoring and Electronic Support. *Issues in Science and Information Technology*. 5 311-323.

Oksansen, K. (2013). Subjective Experience and Sociability in a Collaborative Serious Game. *Simulation and Gaming*. 44(6) 767-793

Ozcelik, E., Cagitay, N.E., & Ozcelik , N.S. (2013). The effect of uncertainty on learning in game-like environments. *Computers and Education*. 67 12-20

Palmunen, L.M., Pelto, E., Paalumaki, A., & Lainema,T. (2013). Formation of Novice Business Students Mental Models through Simulation Gaming. *Simulations and Gaming.*44(6) 846-868. doi:10.1177/1046878113513532.

Saito,D., & Yamauara, T. (2013) A New Approach to Programming Language Education for Beginners with Top Down Learning. *International Journal of Engineering Pedagogy. Special Edition.* 16-21. doi: 10.3991/ijepv3i54.3216

Sorsana, C., Guizard, N., & Trognon, A. (2013). Preschool children's conversational skills for exploring game rules: communicative guidance strategies as a function of type of relationship and gender. *European Journal of Psychology in Education.* 28(4) 1453-1475. doi:10.1007/5 10212-013-0175-4.

Sostak, B.R. (2012). Applying Serious Game Principles to U.S. Military Training. *Journal of Applied Learning Technology.* 2(1) 19-22.

Tsai,C-W., & Fan,Y-T. (2013). Research Trends in Game Based Learning Research in Online Learning Environments. A review of studies published in SSCI-indexed journals from 2003 to 2012. *British Journal of Educational Technology*. 44(5) pE 115-E 119. doi:10.111/bjet1203.

vonWangenheim, C.G., Savi, R., Borgatto, A.F. (2013). Scrumia- An education game for teaching SCRUM in computing courses. *Journal of Systems and Software*. 86(10) 2675-2687.

Wang, L-C., & Chen, M-P. (2010). The effects of game strategy and preference matching on flow experience and programing performance in game based learning. *Innovations in Education and Teaching International.* 47(1) 39-52.

Wang, A.I., & Wu, B. (2009). An application of a Games Development Framework in Higher Education. *International Journal of Computer Games Technology*. Volume 2009 Article ID 693267.1-12 DOI:10.1155/2009/69327

Westerna,W., Nadolski, R.J., Hummel, H.G.K., Wolperis, I.G.J.H., (2008). Serious Games for higher education; a framework for reducing design complexity. *Journal of Computer Assisted Learning*. 24(5) 420-432. doi: 10.111/j.1365-2729.2008.00279x

Wouters,P., van Nimwegen, C., van Oostendorp, H., van der Spek, E.D. (2013). A Meta-Analysis of the Cognitive and Motivational Effects of Serious Games. Journal of Educational Psychology. 105 (2) 249-265.

W-H.Wu, H-C. Hsiao, P-L.Wu, C-H. Lin & S-H. Huang (2012). Investigating the learning-theory foundations of game-based learning: a meta-analysis. *Journal of Computer Assisted Learning*. 28, 265–279. doi: 10.1111/j.1365-2729.2011.00437.x