Programming Education via Microcontroller Game Engines

Literature Review

Mitch Myburgh
University of Cape Town
Rondebosch
Cape Town, South Africa
mybmit001@myuct.ac.za

ABSTRACT

Computer Science is an important field in the modern world, and engaging students in first year programming courses is paramount for producing the modern high-tech workforce that South Africa desperately needs. Furthermore with the rise of connected and 'smart' devices, hardware programming is also becoming an important feature of a modern Computer Science education. Engaging students using games and easy-to-use microcontrollers, such as the Arduino, is an important step towards motivating students and increasing interest in the discipline of Computer Science.

First year Computer Science can be difficult for many students and engaging and motivating them to learn outside the classroom is important for them to pick up the skills necessary to compete in the modern world. The use of game programming assignments and Arduino programming has been shown to provide students with extra motivation, to not only learn more but also to extend their knowledge into more complex areas.

Integrating games (in the form of game-based programming assignments) and Arduino programming into a single assignment could provide a solution to flagging enrollment numbers and reengage students in the study of Computer Science.

CCS Concepts

•Applied computing → Interactive learning environments; •Hardware → Sensor applications and deployments; Sensor devices and platforms; *Displays and imagers; Networking hardware;* **•Human-centered computing →** Handheld game consoles;

Keywords

Education; Microcontrollers; Arduino; Programming; Game Framework

1. INTRODUCTION

WOODSTOCK '97 El Paso, Texas USA
© 2016 ACM. ISBN 123-4567-24-567/08/06...$15.00
DOI: 10.475/123_4

Computer Science and programming is a fast growing field, with computer programming being among the industries with the fastest growth in demand[30] from 1965 to 1994 in South Africa. Despite the large demand South Africa has a lack of qualified Computer Scientists, *Daniels and Reza* list computer-related professionals in their list of scarce skills[9] and the South African Government list *Information and Communication Technology* and *Information Technology* on their 2014 *List of Occupations in High Demand*[24].

Hardware devices such as microcontrollers and even computers have reduced in price in recent years, making developing hardware systems and infrastructure accessible to more people than ever before. Devices such as the Arduino, an open-source, low-cost, and easy to use microcontroller board, have enabled anyone to own and build microcontroller systems. With the rise of the Internet of Things[19], hardware and micrcontroller programming is becoming more and more important.

Having a robust knowledge of not only computer programming, but programming on dedicated hardware (such as microcontrollers) is an essential skill in the modern world, one that is needed in South Africa. Equipping students with these skills will allow them to compete in the knowledge-based economies of the 21^{st} Century[28].

Futhermore engaging often apathetic students in Computer Science is an important task. In this review, the problems with programming education are looked at, followed by a look at potential solution in the form of games and microcontroller education.

The project is an investigation into the use of a game framework running on microcontrollers (namely the Arduino system) in order to teach first year students programming, and engage and interest them in expanding their knowledge. The programming will be done in the low-level C language, which is a procedural language so is relatively easy for students to pick up after their introduction in Python.

2. PROGRAMMING EDUCATION

Learning programming is difficult[11][15][18]. From information technology courses in South African schools[18] to introductory Computer Science courses at universitys around the world[15] the literature agrees that teaching and learning programming skills are both difficult tasks that require refinement in order to equip more students with the basic coding and problem solving skills learned in introductory Computer Science courses.

Coupled with this difficulty computer science courses are seeing declining enrollment[21][28][20][8][2] and high dropout rates[2].

2.1 Problems in Programming Education

A report by the *ITiCSE 2001 Working Group*[22] in which the programming skills of first year computer science students were evaluated it was found that many of the students do not have sufficient programming and problem solving skills.

The *ITiCSE 2001 Working Group* was formed due to the belief among educators that their students weren't learning and sought to evaluate this claim through a common test administered to 216 students at 4 universities[22]. The test involved 3 programming assignments where the students were required to design programs to evaluate mathematical expression (postfix, infix and infix with bracket precedence in order of difficulty). The students preformed poorly on this task achieving just an average score of 22.89 out of a possible 110, and many of the solutions did not even compile[22].

The *ITiCSE 2001 Working Group*'s report only evaluated a small subsection of the approximately 2 million students who enroll in Computer Science every year[3], however it provides additional evidence to the claims made in other papers[11][15] that programming is difficult for students to learn. It also illustrates the claim[11] that students don't understand algorithms, and cannot develop, test or fix their own algorithms.

Another paper by Jens Bennedsen and Michael E. Caspersen[3] investigates failure rates in first year computer science courses. In a survey of 63 institutions it was found that 67% of first year Computer Science students pass the course, which means that of the 2 million students who in enroll in such courses worldwide, 650,000 fail to pass the course[3].

Bennedsen et al.[3] acknowledge that their data pool is small however their data does align with the information presented in other papers[11][15].

In parallel to this lack of knowledge and high failure rates, Leutenegger and Edgington[21] report that from 2000 to 2005 the number of incoming computer science majors dropped by 60% to 70%, while Bayliss[2] reports a more conservative 50% drop from 2000 to 2007 and a dropout rate of 30% to 40%, which is in line with Bennedsen et al.'s [3] first year drop out rate of 33%.

This reduction in incoming students is potentially due to the perceived difficulty of the course while the high drop out rate could be attributed to both the actual difficulty of the course and the lack of motivation for students.

2.2 Possible Reasons for Problems in Programming Education

The issues put forward by the *ITiCSE 2001 Working Group*[22] and Bennedsen et al.[3] regarding the inability of students to program and their failure rates has a number of reasons, chief among them the difficulty (and perceived difficulty) of introductory programming courses.

Papers by Anabela Gomes and António José Mendes [11], and Tony Jenkins[15] discuss this issue, suggesting reasons why the courses are considered difficult and possibly what steps can be taken to remedy the situation and more effectively teach students programming skills.

One of the problems suggested by Gomes et al.[11] is that while introductory programming and Computer Science courses are designed to teach problem solving skills[11] in the context of programming, many students lack the basis for these skills coming in to the course and are left behind due to the pace[15] necessary to complete the syllabus. The student also usually needs rudimentary mathematical ability[15] coming in to the course and Mathematics is another subject considered to be extremely difficult.

The programming language used in the introductory courses are also considered an issue for students[11][15][22]. They are often very complex[11] due to being designed for use in industry and not for teaching[11][15]. While languages such as *Pascal* and *LOGO* are designed for teaching[15] many courses favour languages used in the 'real world' such as *Java* for recruitment purposes[15]. The *ITiCSE 2001 Working Group*[22] also mentions that complex languages (such as Object-oriented ones) take more time to teach, leaving less time in the syllabus for more advanced topics.

Another issue put forward are a lack of motivation, or external motivations for entering into the field[15] that are unable to sustain the student's interest throughout the degree. Coupled with Jenkins' assertion that the content and assignments could be considered boring[15] and the statement by Gomes et al.[11] that there are growing negative connotations surrounding Computer Science, this results in students often losing interest in the course.

Other issues for poor performance in introductory programming courses are the need for multiple learning styles[11][15] for different students[11] and within different section of the content[15] (i.e. syntax must be memorised, but a given algorithm is better to be understood than memorised); abstract concepts are considered hard[11]; and the fact that many students will learn programming for the first time in university.

2.3 Possible Solutions for Problems in Programming Education

Gomes et al.[11] and Jenkins[15] put forward potential solutions to the problems highlighted in the above section.

Among the potential solutions highlighted by Gomes et al.[11] two stand out as relevant, namely including games in the syllabus (This will be discussed in depth in the section on *Programming Education Through Games*), and using programming patterns to assist in development. Gomes et al.[11] suggest that incomplete patterns be presented to students that they then fill out with code this can assist students struggling to create the structure of the program as well as to provide examples of best practices and reinforce these in the student's mind.

Other solutions put forward by Gomes et al.[11] are continuous assessment and the modification of the content to focus on the student's needs; and focusing on the student's learning style and tailoring content to that style.

Jenkins[15] states that some fundamental changes need to happen in order for programming education to become more effective, including choosing suitable education languages (*Pascal* and *LOGO*[15]) over those popular in industry; and making the course flexible in order to accommodate different learning styles.

3. PROGRAMMING EDUCATION THROUGH GAMES

Games are probably the most obvious tools in increasing engagement and learning in the Computer Science classroom, as most students entering the field of computer science are familiar with games[20][28][21]. Games have captured the attention of young people and become a multibillion dollar[26] industry in the process, they hold the attention of players for often long stretches of time[26], during which they learn from the game world, it would be advantageous if this could be applied to the study of Computer Science.

One of the criticisms for using games is that they alienate students who don't play games and women. The first issue can be solved by giving students multiple assignment choices[8][7], the second issue can be considered moot as women make up 45% of people who play games[21], with Bayliss[2] reporting that 65% of women play games.

Sung[28] breaks the integration of computer games into Computer Science courses into three categories: *Game development classes*, classes where the goal of the course is to make a game; *Game programming classes*, classes where the technical aspects of game development are discussed, such as path-finding algorithms; and *Game development client* which is the integration of games into the curriculum, as assignments and learning aides. This review will will discuss *Using Games to Teach Computer Science* which will look at games that teach programming topics and *Using Games as Programming Assignments* which will look at using games in programming assignments in order to increase student motivation. The section on *Using Games as Programming Assignments* is of the most relevance to the proposed project and so will be discussed in the most depth.

3.1 Using Games to Teach Computer Science

Due to their interactive and self paced nature, computer games can be useful for teaching students new and difficult concepts. Games also capture players' attention and can also be used as a motivational tool in the classroom through gamification[25].

Kahn[16] describes a game called *Toontalk* which teaches children programming skills by building simple programs within a game world. The abstractions make the 'code' easy for children to understand and the game fun to play, with robots representing functions (they perform a set operation given an input) and boxes representing data structures (they can be given to robots to manipulate). Kahn's work illustrates how programming can be taught in an engaging way, and while simplistic and for children, his ideas could be applied to a more advanced game for older students.

Zapušek et al.[31] also describe a game for children to teach concepts related to variables.

Muratet[23] describes a RTS game aimed at University level students in order to teach them programming.

Bayliss[2] descibes a UNIX scavenger hunt that was used to teach students the often cryptic (when coming from GUI focused systems such as Windows) UNIX command line.

3.2 Using Games as Programming Assignments

Another interesting development is the use of games as small programming assignments that replace or complement traditional programming assignments. Game assignments can provide students with additional motivation[21][8], can be implemented without sacrificing content[21][28], and the motivation from games can encourage students to look into more complex topics in Computer Science[20].

Students are often interested in making games, with many choosing a computer science major based on their love of games, and desire to make their own[20]. These students may become disillusioned by difficult, boring or seemingly irrelevant assignments[20][21]. However integrating game based assignments could increase motivation and keep students interested in the course.

Two papers by Cliburn[7][8] discusses experiments in which he provided students with a choice between game-based and traditional assignments in the first experiment[7] and a choice between game-based, traditional and interactive story assignments in the second experiment[8]. In both experiments the students overwhelmingly chose the game based assignments with 79% of students choosing games in the first experiment and 71% of students choosing the game based assignment in the second experiment.

In his first study Cliburn[7] found that game assignments had a small impact on marks, with higher marks coming from the non-game assignments (although both the means were within one standard deviation of each other). One reason considered by Cliburn for this was that stronger students gravitated towards assignments that were harder in order to take on a larger perceived challenge. What Cliburn did find was that game-based assignments increased student motivation, with the majority of students surveyed commenting that it did increase their motivation. Students also considered the game based assignments as their favourites, and often chose them over traditional assignments that were seen as more difficult (despite the fact that the content under examination was identical). Cliburn also found that while women chose game-based assignments less, they still overwhelmingly preferred them to traditional assignments.

Cliburn's second study[8] backed up the results of the first one, where again game-based assignments were far more popular than either story-based ones (the worst performer of the three) or traditional ones. One of the notes he made was that while highly motivating and interesting to students, lecturers would have to undertake additional effort to create such assignments.

Both Cliburn's[7][8] studies focused on a single class, and as such may not be representative of the larger population of computer science students.

In their paper Leutenegger et al.[21] discuss the use of games in an introductory Computer Science course at the University of Denver. Having recently introduced a game development degree, they integrated games into the standard Computer Science curriculum. These games allowed students to get immediate visual feedback as to issues and bugs in their code, and Leutenegger et al. assert they were able to achieve this without sacrificing content.

Sung et al.[29] produced a series of game themed assignments that could be dropped into existing courses with minimal effort, taking the technical topics that needed to be tested and integrating them into games in order to provide additional motivation for students. Sung et al. also used the approach suggested by Gomes et al.[11] in which they provided students with a program skeleton including complete graphics and interaction functionality so that the student can focus on developing the required algorithm, while still being able to see the visual results of their programming.

In another article Sung[28] further discusses the use of games in programming assignments. Sung notes that in elective Computer Science courses, such as artificial intelligence, software engineering and computer graphics, game-based assignments fit particularly well given the close links these topics have to the game industry.

However in both his papers Sung[29][28] mentions that it is difficult to get buy in from lecturers with little or no game or graphics experience, however this can be solved by providing already complete assignments with scaffolding. Sung also produced a library in his first paper so that lecturers with the required skill and interest could create their own game-based assignments.

Kurkovsky[20] discusses using mobile game assignments as a method to teach his students, with the games providing motivation to students when studying more advanced topics such as databases, networking, artificial intelligence and software engineering.

Bayliss's[2] game-based Compuer Science course showed a 93% retention rate throughout the degree, showing the power that integrating games into the Computer Science curriculum could have on retaining and engaging students in the content. She also makes use of game scaffolding which enables students to focus on the section of the code that is required while still having the motivation of working on a game. She also mentions another potential benefit of scaffolding, when reading the code students can learn better code styles, as well as the build their skill level in reading code.

4. EDUCATION THROUGH MICROCONTROLLERS

Microcontrollers have become a pervasive part of the modern world, used in industry, cars, consumer electronics and others[13] to create connected and 'smart' devices. As a result many attempts have been made to engage students in the programming and creation of microcontroller enabled devices from using e-textiles[5][6], to designing graphical programming languages to help students to learn to use the devices[4][12][17].

The Arduino is a popular kit for many microcontroller based courses, due to its open-source nature, ease of use, low cost, and strong community[14].

Rubio et al.[27] made use of Arduinos in a course to teach traditional programming. The students were exposed to a traditional lecture, with integration of an Arduino example to illustrate concepts such as loops, conditional statements and arrays. The students were then given the opportunity to build their own Arduino based systems in lab sessions. 95% of students found the lab sessions interesting and 85% enjoyed the lecture demos. This increased interest translated directly into a 32% increase in the number of students achieving a good programming level, and a 21% increase in their confidence in programming over previous courses that did not use Arduinos. The investigation has only been applied to one course of one degree, however the results are promising for the usefulness and integration of Arduinos into other courses.

Jamieson[14] investigated the use of Arduinos in an Engineering course designed to teach programming and microcontrollers to students. The students had to present a midterm and final project based on a microcontroller with 90% of the class choosing Arduinos. In his study Jamieson found that the projects produced with Arduinos were far more complicated than those produced in previous years where Arduinos were not available. This shows that due to their ease of use, Arduinos are perfect for quickly protoyping advanced systems, and that students can learn electronics concepts faster with them.

The LilyPad Arduino[5][6] is a project that sought to engage the imagination of students through the design and implementation of wearable computing devices and e-textiles, based on a customised Arduino board. Buechley et al.[5][6] designed the board and a summer camp style programming and development course. They were able to not only engage students (including a large proportion of females), but were also able to stimulate interest in computing and programming, with some students expressing interest in taking further courses in wearable computing and 3 out of 8 students expressing interest in programming and using electronics at home.

Use of robots based on the Arduino platform have also been used to teach programming to design students[1]. Alers and Hu built the AdMoVeo platform in an attempt to teach designers programming skills in order to equip them with the skills necessary to build interactive protoypes and products. The Arduinos were mounted on a robot and the students programmed them using Processing, a popular language for design and prototyping. The 140 students in the study were able to produce complex robotic behavior including automated racers and maze escaping robots.

Dziabenko et al.[10] also provided an interface for learning to program where students coded a microcontroller robot over the internet to complete a game. The students did not build the robot themselves, but were engaged in microntroller programming, without having to own an Arduino board themselves.

Many projects have been undertaken in order to make use of visual programming languages such as Scratch on the Arduino[4][12][17]. While the project proposed will be designed to teach students text based programming, it is interesting to look at the results of studies investigating graphical programming languages on the Arduino.

Booth and Stumpf[4] found that while students found text based coding to be difficult due to the range of syntaxes and confusing console output, many found programming an Arduino board with a graphical language to be easier. Gupta et al.[12] conducted a similar study in Indian high schools and found a 27% to 42% increase in programming skills of those that coded an Arduino using Scratch over those that coded it traditionally. Gupta et al.[12] chose the open-source Arduino project in order to investigate the teaching of hardware at Indian schools, so as to prepare students for jobs in the marketplace.

5. DISCUSSION

Programming skills are fast becoming a necessity in the modern workplace, from designers developing interactive prototypes to developers developing the software on which many businesses are now based, because of this those with programming experience are in high demand[24]. But despite the prospect of lucrative careers, many are turned off by the prospect of programming finding it difficult and lacking the motivation to pursue a career in Computer Science.

Two possible solutions which have been shown to capture the attention and interest of students are the use of games (whether to teach concepts or used as assignments) and the use of Microcontrollers such as the Arduino.

Integrating games in the curriculum increases students' motivation, and when given a choice they overwhelmingly choose game-based assignments over traditional assignments, considering them to be more fun, and even less hard (despite examining the same content)[7][8]. It is clear that integrating games into Computer Science courses is an important step towards engaging modern students, many of whom play games regularly. Integrating games into the curriculum could have some problems, namely alienating of people who don't play games (and in particular women), however by providing multiple assignment choices[7][8] this problem can be mitigated. Furthermore, many women today play games[21] and so will not be alienated by game-based content in courses. Developing two (or more) types of assignments for each assignment can however be taxing on the lecturers, who often don't have much time.

Another development in motivating students towards Computer Science education is the use of hardware in the course. Hardware such as microcontrollers and in particular the Arduino have become highly popular in recent years, thanks to the demand for connected and smart devices comprising the Internet of Things[19]. Being able to see physical results motivates students when learning to program, and students find the use of Arduinos in lab sessions interesting[27]. Furthermore Arduinos can be used to introduce programming to students who have never programmed before, such as in the LilyPad projects[5][6]. Despite being extremely attractive as an educational aide and being low cost, buying a large number of microcontrollers can be an expensive exercise for Computer Science departments to undertake.

A new potential solution to providing motivation to students in Computer Science would be to integrate both games and microcontroller programming. A game framework, that provides a scaffold for students to fill in, that links to an Arduino controlled game board could provide the best features of game-based assignments and microcontroller programming. Giving students an opportunity to build their own games, as well as program on the Arduino, will help them learn skills that are valuable in the modern workplace.

6. CONCLUSIONS

With the high demand for Computer Science graduates in the modern workplace it is imperative that students become engaged and interested in Computer Science. The perceived and actual difficulty, high drop-out rates and lack of motivation of students need to be addressed in order to stem the tide of decreasing Computer Science enrollments.

A number of possible solutions exist, the two most promising being the integration of games into the curriculum, which has been shown to vastly increase student involvement and interest, even reducing the perceived difficulty of the course, as students consider game-based assignments to be easier than traditional assignments; and the use of microcontrollers, in particular the open-source, low cost, Arduino boards, which also have piqued the interest of students and allow them to see physical results from their 'virtual' code, while also giving them experience in the design of electronics.

The project that will be presented will integrate these two facets to attempt to create an interesting programming assignment that not only teaches students programming in C, and the use of Arduinos, but also engages them and increases their interest and motivation in Computer Science.

7. REFERENCES

[1] S. Alers and J. Hu. Admoveo: A robotic platform for teaching creative programming to designers. In *Learning by playing. Game-based education system design and development*, pages 410–421. Springer, 2009.

[2] J. D. Bayliss. Using games in introductory courses: Tips from the trenches. In *ACM SIGCSE Bulletin*, volume 41, pages 337–341. ACM, 2009.

[3] J. Bennedsen and M. E. Caspersen. Failure rates in introductory programming. *ACM SIGCSE Bulletin*, 39(2):32–36, 2007.

[4] T. Booth and S. Stumpf. End-user experiences of visual and textual programming environments for arduino. In *End-User Development*, pages 25–39. Springer, 2013.

[5] L. Buechley and M. Eisenberg. The lilypad arduino: Toward wearable engineering for everyone. *Pervasive Computing, IEEE*, 7(2):12–15, 2008.

[6] L. Buechley, M. Eisenberg, J. Catchen, and A. Crockett. The lilypad arduino: using computational textiles to investigate engagement, aesthetics, and diversity in computer science education. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 423–432. ACM, 2008.

[7] D. C. Cliburn. The effectiveness of games as assignments in an introductory programming course. In *Frontiers in Education Conference, 36th Annual*, pages 6–10. IEEE, 2006.

[8] D. C. Cliburn and S. Miller. Games, stories, or something more traditional: the types of assignments college students prefer. *ACM SIGCSE Bulletin*, 40(1):138–142, 2008.

[9] R. Daniels. Skills shortages in south africa: A literature review. *DPRU Working Paper*, (07/121), 2007.

[10] O. Dziabenko, J. García-Zubia, and I. Angulo. Time to play with a microcontroller managed mobile bot. In *Global Engineering Education Conference (EDUCON), 2012 IEEE*, pages 1–5. IEEE, 2012.

[11] A. Gomes and A. J. Mendes. An environment to improve programming education. In *Proceedings of the 2007 International Conference on Computer Systems and Technologies*, page 88. ACM, 2007.

[12] N. Gupta, N. Tejovanth, and P. Murthy. Learning by creating: Interactive programming for indian high schools. In *Technology Enhanced Education (ICTEE), 2012 IEEE International Conference on*, pages 1–3. IEEE, 2012.

[13] D. J. Jackson and P. Caspi. Embedded systems education: future directions, initiatives, and cooperation. *ACM SIGBED Review*, 2(4):1–4, 2005.

[14] P. Jamieson. Arduino for teaching embedded systems. are computer scientists and engineering educators missing the boat? *Proc. FECS*, pages 289–294, 2010.

[15] T. Jenkins. On the difficulty of learning to program. In *Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences*, volume 4, pages 53–58, 2002.

[16] K. Kahn. A computer game to teach programming. In *Spotlight on the Future, NECC '99. National Educational Computing Conference Proceedings.* ERIC, 1999.

[17] Y. Kato. Splish: a visual programming environment

for arduino to accelerate physical computing experiences. In *Creating Connecting and Collaborating through Computing (C5), 2010 Eighth International Conference on*, pages 3–10. IEEE, 2010.

[18] M. Koorsse, C. B. Cilliers, and A. P. Calitz. Motivation and learning preferences of information technology learners in south african secondary schools. In *Proceedings of the 2010 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists*, pages 144–152. ACM, 2010.

[19] H. Kopetz. Internet of things. In *Real-time systems*, pages 307–323. Springer, 2011.

[20] S. Kurkovsky. Engaging students through mobile game development. In *ACM SIGCSE Bulletin*, volume 41, pages 44–48. ACM, 2009.

[21] S. Leutenegger and J. Edgington. A games first approach to teaching introductory programming. In *ACM SIGCSE Bulletin*, volume 39, pages 115–118. ACM, 2007.

[22] M. McCracken, V. Almstrum, D. Diaz, M. Guzdial, D. Hagan, Y. B.-D. Kolikant, C. Laxer, L. Thomas, I. Utting, and T. Wilusz. A multi-national, multi-institutional study of assessment of programming skills of first-year cs students. *ACM SIGCSE Bulletin*, 33(4):125–180, 2001.

[23] M. Muratet, P. Torguet, J.-P. Jessel, and F. Viallet. Towards a serious game to help students learn computer programming. *International Journal of Computer Games Technology*, 2009:3, 2009.

[24] B. E. Nzimande. List of occupations in high demand: 2014. *Department of Higher Education and Training*, 2014.

[25] S. O'Donovan, J. Gain, and P. Marais. A case study in the gamification of a university-level games development course. In *Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference*, pages 242–251. ACM, 2013.

[26] M. Prensky. Digital game-based learning. *Computers in Entertainment (CIE)*, 1(1):21–21, 2003.

[27] M. A. Rubio, C. M. Hierro, and Á. Pablo. Using arduino to enhance computer programming courses in science and engineering. In *Proceedings of the EDULEARN13 Conference*, pages 5127–5133, 2013.

[28] K. Sung. Computer games and traditional cs courses. *Communications of the ACM*, 52(12):74–78, 2009.

[29] K. Sung, M. Panitz, S. Wallace, R. Anderson, and J. Nordlinger. Game-themed programming assignments: the faculty perspective. In *ACM SIGCSE Bulletin*, volume 40, pages 300–304. ACM, 2008.

[30] I. Woolard, P. Kneebone, and D. Lee. Forecasting the demand for scarce skills, 2001–2006. *Human Resources Development Review*, pages 458–474, 2003.

[31] M. Zapušek and J. Rugelj. Learning programming with serious games. *Game-Based Learning*, 1:13, 2013.