Teaching programming using computer games: a program language agnostic approach

Law, Bobby

Published in:
Proceedings of 16th European Conference on eLearning (ECEL 2017)

Publication date:
2017

Document Version
Publisher's PDF, also known as Version of record

Link to publication in ResearchOnline

Citation for published version (Harvard):
Law, B 2017, Teaching programming using computer games: a program language agnostic approach. in *Proceedings of 16th European Conference on eLearning (ECEL 2017).* Academic Conferences and Publishing International Limited, Reading, pp. 368-376.

Teaching Programming Using Computer Games: A Program Language Agnostic Approach

Robert Law
Glasgow Caledonian University, UK
robert.law@gcu.ac.uk

Abstract: Programming can be a challenging subject for beginners to comprehend but a very satisfying and enjoyable one for those who apply the time and effort required to master it. Students can approach programming with a fixed mindset, perceiving the subject to be difficult before they even start. Such a mindset will result in the student procrastinating, being less motivated to participate and possibly placing metaphorical barriers to their learning where there are none. Certain programming concepts can appear, to the student, abstract in nature and intrinsically linked to the programming language being used. Dispelling this thought is a challenge requiring an approach that decouples the programming language from the fundamental programming building blocks. The fundamental programming building blocks of sequence, selection and iteration can be illustrated and enhanced through the use of computer games designed to teach programming. Such games present the player with a challenge to solve that requires a level of problem solving and computational thinking to achieve the desired solution. The purpose of these programs is to give the student a platform to visualise the abstract concepts of programming allowing them to build on their ability to solve problems and create algorithmic solutions. A genre of such games is the "program your robot" which offer a maze style approach where the student must use the correct programming building blocks in a suitable order to achieve the goal of safely navigating the robot from the start of the maze to the designated finish. The visual nature of these games allows the students to visualise the problem enabling them to discern the steps required to complete the task. Students are asked to write down the steps needed to solve the problem in a basic algorithm and implement their algorithm. This subsequently leads to the students "debugging" and amending their initial algorithms thus developing another valuable skill. With each level the nature of the puzzle gets more complex introducing new concepts and reinforcing the fundamental building blocks. The students are also building up a portfolio of problem solving skills and algorithm designs coupled with valuable debugging skills. This paper will discuss an ongoing attempt to introduce problem solving, algorithm development and elementary debugging with second year Game Design students undertaking a C++ game development module.

Keywords: programming, problem solving, algorithms, programming building blocks

1. Introduction

Teaching programming can be fraught with difficulty from both the perspective of the Lecturer and the student Learner. From the point of view of the Lecturer; what language should be taught? what programming paradigm should be taught? Leutenegger & Edgington (2007) advocate that the issue of programming paradigm, procedural or object oriented (OO) approach, to programming is of lesser importance than the engaging nature of the examples used when teaching.

If the programming paradigm is not of major concern, then it is possible that to teach the fundamental building blocks of programming does not require the Lecturer to choose a specific language but instead can be language independent. Rajaravivarma (2005) emphasises that introductory programming courses should develop the students programming skills through the use of problem solving and logical thinking and as such suggests a games-based approach for both engaging the students and developing this skillset.

Removing the language and programming paradigm from the teaching equation allows the teaching to concentrate on the fundamental building blocks of sequence, selection and iteration. Without the syntax of a programming language acting as a constraint a parallel can be drawn between sequence, selection and iteration and everyday life thus reinforcing the concepts and their relationship to problem solving.

Thus, computer games specifically designed to teach programming can be used to help students with both problem solving and learning programming concepts as stated by Rajaravivarma (2005) that problems can be abstract in nature thus game playing offers a "visual representation" of the problem.

The remainder of this paper is organized as follows: Section 2 gives information about pedagogical issues related to learning to program, Section 3 introduces the online programming game used with the students and the subsequent task undertaken with the students. Section 4 discusses the results gleaned from surveying the students with regard to the tasks introduced in Section 3. Section 5 offers a summary of the student experience of using the selected programming game, and concludes the paper giving proposals for future work.

2. Literature review

Ibrahim et al. (2010) proffer that programming can prove difficult for students' due to the abstract nature of some of the underlying concepts coupled with the logical thinking and problem solving required and as such suggest that students' perceive programming to be testing thus lack the required drive to "learn it". Lau & Yuen (2009) suggest that learning to program is a “highly cognitively demanding task for students”. Vahldick et al. (2014) also observe that students do not find programming a simple task suggesting that a complex skillset needs to be developed which requires a level of "high motivation". To this end Ibrahim et al. (2010) suggest that the use of educational games can contribute to the improvement of the students' "motivation and learning perceptions towards Programming." Preliminary research undertaken by Ibrahim et al. (2010) has identified a number of reasons for the students' lack of motivation including: perception of the subject as being "boring"; difficulty understanding explanations; lack of engaging teaching methods and, possibly surprisingly, "not enough exercises".

Robins et al. (2003) advocate that the norm for introductory programming textbooks is to allot a considerable proportion of their volume to familiarising the reader with a particular programming language and, in their view, this mirrors what happens with beginners programming courses.

Leutenegger & Edgington (2007) suggest that the examples need to be alluring enough to engage the student otherwise interest will wain thus negating the argument of procedural or OO approach. Regardless of whether students are taught procedural programming or object oriented programming students have to still learn the fundamental programming building blocks (Lahtinen et al. 2005). For this reason Leutenegger & Edgington (2007), promote a "Game First" approach believing that game programming piques the interest and curiosity of the majority of beginning programmers.

Rajaravivarma (2005) proposes that the most effective approach to master programming is "by actually doing it." This approach relies on the student actively engaging with the learning process and showing a willingness to program outwith their nominated class times. Vahldick et al. (2014) note the fact that learning to program requires plenty of practice and engagement outwith scheduled class time, however, students can lack the required motivation to engage with this need. Garris et al. (2002) propose the concept of the "motivated learner" which they state are "enthusiastic, focused, and engaged." Characteristics that they attribute to the "motivated learner" include: interested in and enjoy what they are doing, try hard, persist over time, self-determined, and driven by their own volition.

Coller & Scott (2009) undertook a study comparing their game based delivery of their module against their normal delivery of their module revealing that students undertaking the game based delivery mechanism appeared to apply approximately twice as much time outwith their normal class time to their course work.

Rajaravivarma (2005) uses a game based approach to complement the programming with the problem solving and critical thinking skills by examining the problem in the shape of a game which they play in order to resolve the problem. Robins et al. (2003) also recognise problem solving as a base for teaching programming. The use of games reinforces the need to clearly identify the problem and produce a design solution thus design can be accentuated through the use of games that need a plan or algorithmic approach to succeed (Rajaravivarma 2005).

Rajaravivarma (2005) views programming as "problem solving using a computer language." suggesting after problem identification (Analysis) programming consists of three stages: design, development and testing. Robins et al. (2003) suggest that the available literature tends to view learning to program from a psychological/educational aspect but they do concede that "the basics of good software engineering practice" is important and should also be considered. Interestingly Rajaravivarma (2005) alludes to the students' ability to identify the problem and in some instances when presented with the problem definition their inability to properly understand the presented instructions. Winslow (1996) identifies problem solving as a key skill comprising 4 steps, the second of which is "determine how to solve the problem". Winslow (1996) further subdivides this step to include not just solving the problem in any form but in particular a "computer compatible form". Fincher (1999) suggests that problem solving is a "short-hand encapsulation" of the programming's non-coding skills.

Lahtinen et al. (2005) conducted an international survey of approximately 500 students and teachers to help determine the areas of difficulty students face when learning to program. Their survey comprised three parts; the second part was further split into two covering program construction issues and programming concept issues. The top two issues being program design and identifying tasks that can be converted to procedures.

Lahtinen et al. (2005) survey was mirrored somewhat by Tan et al. (2009) survey of 182 Malaysian undergraduate students to determine their attitude towards and view of learning to program. Their survey covered a number of different areas but of interest is the students' thoughts on what they regard as the difficulties of learning programming. A number of difficulties were identified but two of the top ones were; program design and division of functionality into methods/functions/procedures (Tan et al. 2009). Their result correlates well with Lahtinen et al. (2005) original study.

An interesting observation that Lahtinen et al. (2005) have made from the study of the literature is that students view programming on a "line by line" basis instead of applying "meaningful program structures". Milne & Rowe (2002) suggest that students can sometimes think they understand a topic but when asked to explain the topic their explanation can underline the fact they do not understand the topic as well as they think they do.

When selecting an appropriate game an attempt should be made to match the game against Coller & Scott (2009) suggested three learning principles that are embedded in games: can achieve initial success quickly but difficulty increasing continuously; inspire active and critical learning with relevant knowledge and skills "discovered"; offer escapism/fantasy inspiring "imagination with a sense of unlimited possibilities." This was a good basis to help select a game that would be both educational and fun for the student.

A useful source for selecting an appropriate game was the work undertaken by Vahldick et al. (2014) in their review of games specifically designed for teaching programming concepts. Their work used the Computer Science Curricula 2013 Curriculum Guidelines for Undergraduate Degree Programs in Computer Science (ACM & IEEE 2013) a joint project between the ACM and IEEE. Of particular interest is the fundamental programming concepts unit and its associated topics as shown in Table 1 below.

Table 1: ACM & IEEE fundamental programming concepts

Unit	Topics
1. Fundamental Programming Concepts	1.1 Basic syntax and semantics of a higher-level language 1.2 Variables and primitive data types (e.g., numbers, characters, Booleans) 1.3 Expressions and assignments 1.4 Simple I/O including file I/O 1.5 Conditional and iterative control structures 1.6 Functions and parameter passing 1.7 The concept of recursion

Vahldick et al. (2014) identified three fundamental proficiencies that the beginning programmer should possess: "comprehension, writing and debugging." These are a subset of Shneiderman & Mayer (1979) "five basic programming tasks" of composition, comprehension, debugging, modification and learning.

Vahldick et al. (2014) identified 40 games categorising them based on the following criteria: type, platform, competence, topics and language.

3. Overview of pilot study

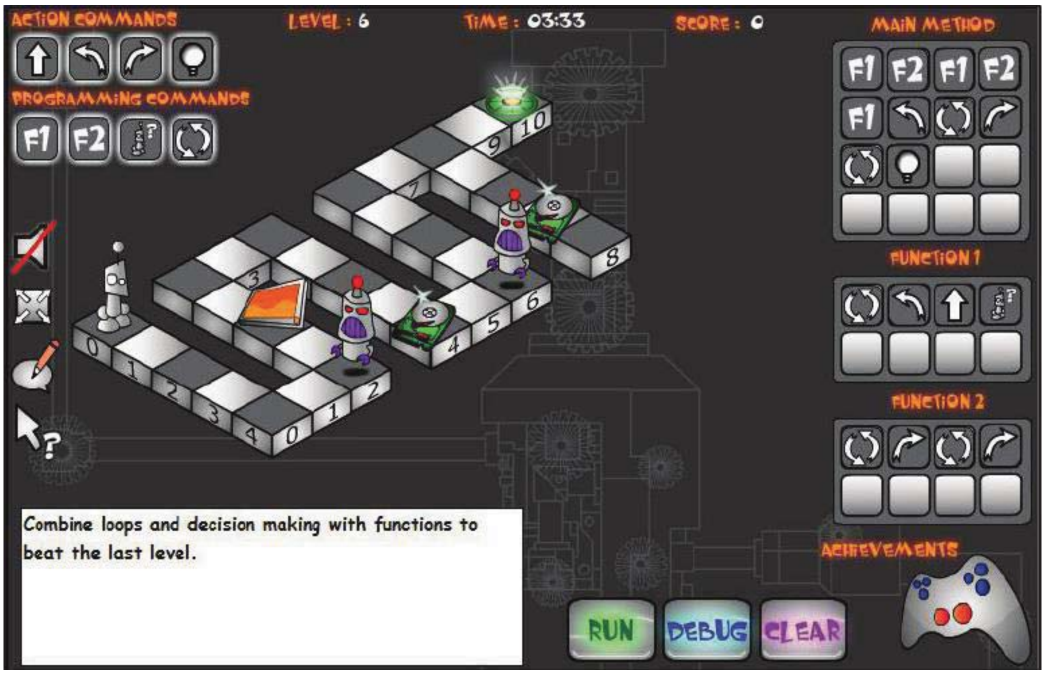
The literature review offers credence to the assumption that games specifically designed to teach programming can be used to teach students, on an introductory programming course, problem solving skills and the programming concepts sequence, selection and iteration. To this end, a suitable computer game was required that would teach the aforementioned skills.

3.1 Selecting an appropriate game

The basis for selecting an appropriate game centred on the 40 games listed by Vahldick et al. (2014). The criteria for selecting a game also included: web-based, matched a number of the criteria in Table 1 and were primarily free. These criteria would allow students to use the game outwith the University campus. A number of games were reviewed; Program Your Robot, BotLogic, Light-Bot2 and Gidget. However, it was decided to use Program Your Robot which is a web based game that covers the topics 1.5, 1.6 and 1.7 from Table 1. Using Program Your Robot follows on from the proposed work of Law (2016). Program Your Robot is a web based platform developed by Kazimoglu et al. (2012) and is available at http://www.programyourrobot.org/ This game will help introduce and reinforce the concepts of sequence, selection and iteration. Program your Robot is classified by Kazimoglu et al. (2012) as a "serious game" where the learner can engage in "algorithm building, debugging and simulation.", enhancing and expanding their computational thinking skills.

The advantage of this environment for the learner is the emphasis on both problem solving and the introduction of the fundamental building blocks of programming without the learner being burdened by any specific programming language and its inherent syntax (Law 2016). Kazimoglu et al. (2012) web based Program Your Robot has been specifically designed to teach introductory programming constructs through the control of a robot to solve puzzles. The game gradually introduces the learner to the programming concepts of sequence, selection, iteration and functions/methods. The learner constructs a "solution algorithm" comprising the correct combination of "action commands" to enable the Robot to reach the designated point of each platform (Law 2016).

The game requires the learner to build a solution algorithm from two categories of commands: action commands and programming commands. Programming commands are used to implement the programming constructs of sequence, selection and iteration. The learner is also exposed to the concept of functions/methods through the ability to "create repeatable patterns". This fits well with Winslow (1996) who proposes the use of algorithms or patterns that can be used to help the student to decompose a problem into workable chunks. These patterns, once learnt, can be applied again and again to successfully transform problem solutions into computer based solutions. Figure 1 shows the programming learning environment for the game. The right hand side of the screen has a main method, the primary control mechanism for controlling the robot, and two functions. The left hand side of the screen has the command categories of action and programming. The interface allows drag and drop of any of the action and or programming commands to fill the empty slots within the main method or the two functions. Run or Debug buttons, located at the bottom left of the screen, are used to execute the built algorithm. If the learner uses the debug button to test their algorithm they will receive feedback in the form of error/warning messages (Kazimoglu et al. 2012).



Combine loops and decision making with functions to beat the last level.

Figure 1: Program Your Robot

Progression through the levels brings into sharp reflection the need to build reusable solution algorithms, hence the need for functions/methods, as the number of free slots in the main method is not enough to successfully complete the level (Law 2016).

From a pedagogical perspective Winslow (1996) generalises Linn and Dalbey's "ideal chain for learning computer programming" (Linn & Dalbey 1985) to

> *"learn one new feature or item at a time,*
>
> *learn design skills combining old strategies and new ones, and*
>
> *learn general problem solving skills plus those unique to this material."*

Program Your Robot appears to fit Winslow (1996) generalisation relatively well.

3.2 Student task

The students undertaking the pilot study consist of two cohorts, both second year undergraduates, the larger of the two are part of the Game Design (GD) programme and the smaller of the groups are part of the Game Software Development (GSD) programme totalling 42 students.

The task presented to the students was quite open ended; they were given the url for Program Your Robot and simply asked to play the game and note how they solved the tasks the game presented to them. On completion of the game the students were asked to complete an online questionnaire, hosted using Google Forms. The students did this exercise as part of a one-hour lab based session.

4. Pilot study results

This section will present the findings from the survey, of which 37 of the 42 students enrolled on the module completed, giving a yield of 88%. All responses to the survey were anonymous.

The first question on the survey asked the students to determine the purpose of the game. Remembering that they had been told nothing about the game a sample of responses is listed below:

> *"In my opinion, the game was trying to teach us the importance of sequence and the uses of loops and functions."*
>
> *"To break down the methods used in programming so people can understand it before they move on to using code themselves."*
>
> *"To teach basics of functions, loops and IF statements as well as basic order of execution"*
>
> *"to help people understand the basics of programming using the game"*
>
> *"To get us thinking about the basic programming functions in a game context"*
>
> *"To help us understand how to efficiently use loops and functions"*
>
> *"To learn the basic principles of programming"*

In general, the students did appear to determine the purpose of the game.

Students were subsequently asked if they found the game enjoyable to play. The students had to select a value on a scale of 1 (Not Very) to 5 (Very). Results are presented in the graph displayed in figure 2.

Some anecdotal comments from students, with regard to enjoyability of the game, are listed below.

> *"Overall I did enjoy the game and would recommend to people in the future."*
>
> *"The game was fun engaging and easy to understand."*
>
> *"I thought it was fun."*

Again, the overall consensus appeared to indicate that the game was enjoyable to play.

In order to gain an insight into how easy the game was to play, the students were asked directly if they found the game easy to play. Again, the students had to select a value on a scale of 1 (Not Very) to 5 (Very).

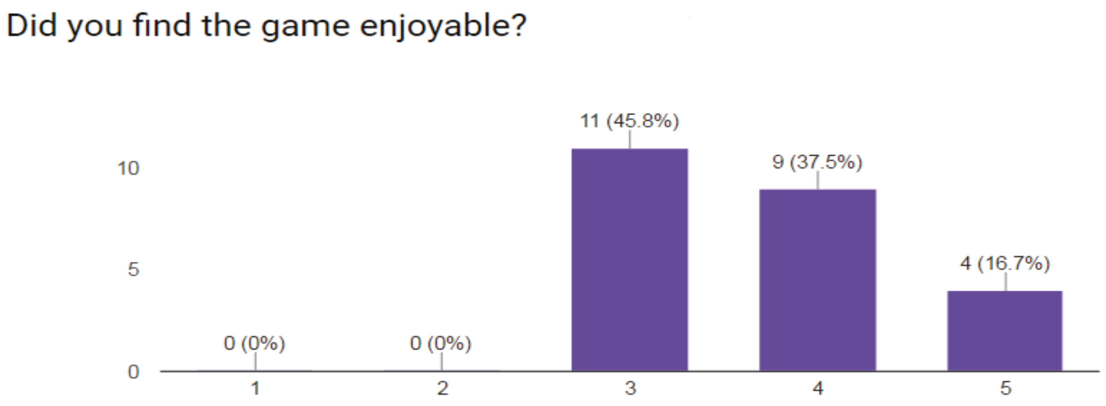


Figure 2: Response to the question "Did you find the game enjoyable?"

Results are presented in the graph displayed in figure 3.

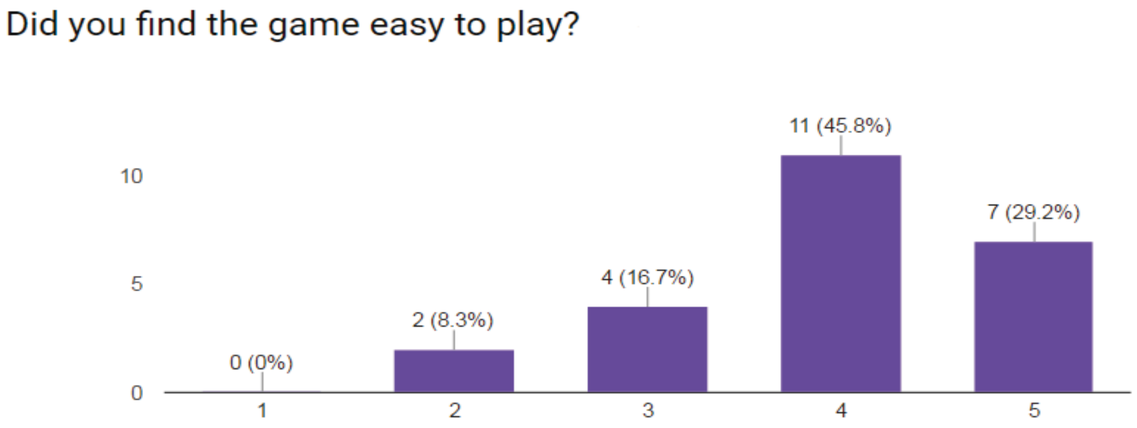


Figure 3: Response to the question "Did you find the game easy to play?"

On the whole, most people found it easy to play but, as can be seen from the graph in figure 3, two respondents indicated that they did not find the game easy to play. Unfortunately, no anecdotal responses were collected for this question and therefore it is not known the reasoning for the two responses. This is obviously an oversight on the data collection abilities of the questionnaire. Speculating, as to why two students did not find the game easy to play, the obvious thought would be that they did not watch the short accompanying explanatory videos or they did not make use of the help feature.

Students were asked if they completed the game. The response to this question was a simple yes or no. The results a presented in figure 4 as a pie chart.

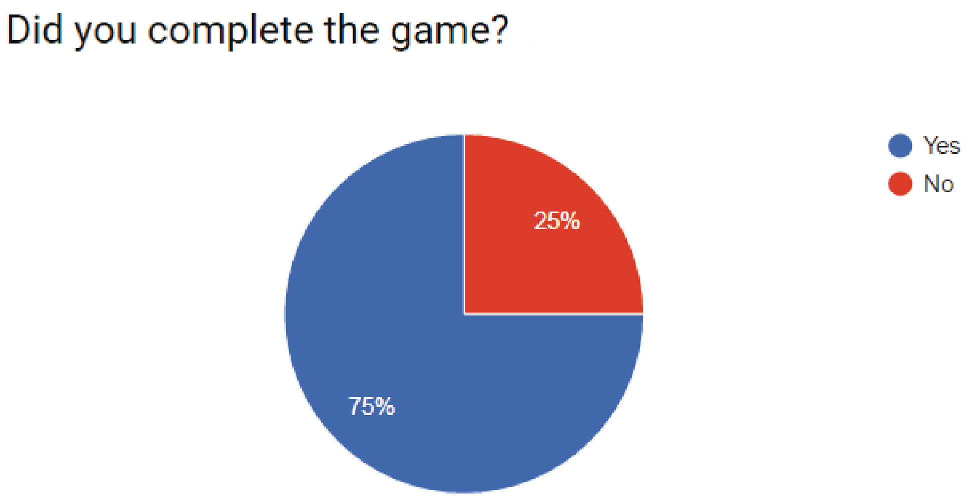


Figure 4: Response to the question "Did you complete the game?"

Rather surprisingly a quarter of the respondents did not manage to complete the game. This was unexpected and again, a failure of the survey to collect a textual reasoning for the failure, leaves the reason for this open to surmise. This may have been due to some of the students repeating levels to attain higher level scores and not completing the game within the allocated lab time.

The game comes with short video tutorials and the students were asked how helpful they felt these tutorials to be. Again, the students had to select a value on a scale of 1 (Not Very) to 5 (Very). Results are presented in the graph displayed in figure 5.

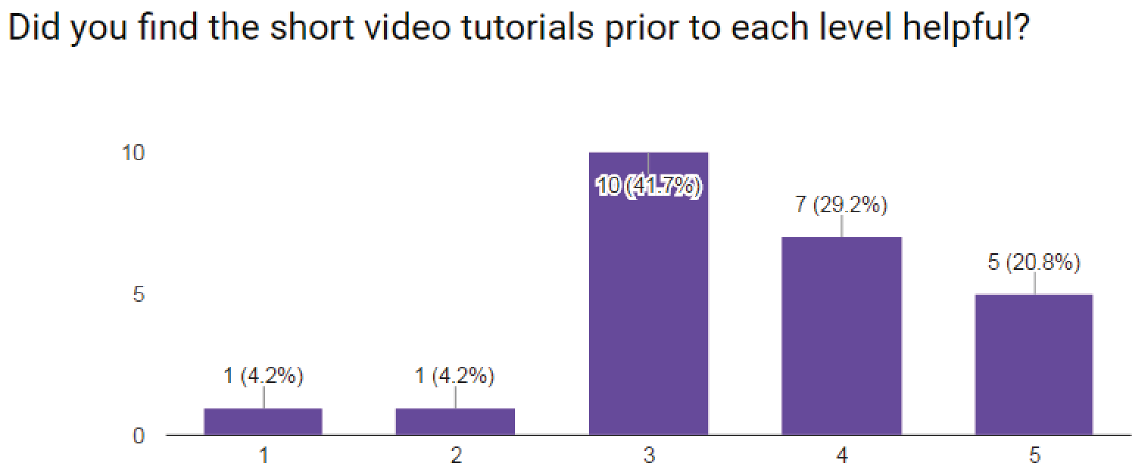


Figure 5: Response to the question "Did you find the short video tutorials prior to each level helpful?"

It’s fair to say that the majority of the respondents felt the video tutorials were useful but it was interesting that two respondents were not happy with the video tutorials. Again, the survey did not permit for the capture of textual input for the explicit reasoning of the respondents as to their unhappiness with the video tutorials. This result ties in with the two students, in figure 3, who found the game difficult to play; as to why they didn’t find the videos helpful it may actually be because they didn’t realise the videos were available until they had reached the later stages of the game.

Students were asked to comment on the ease of use of the games user interface. Results are presented in the graph displayed in figure 6. Again, the students had to select a value on a scale of 1 (Not Very) to 5 (Very).

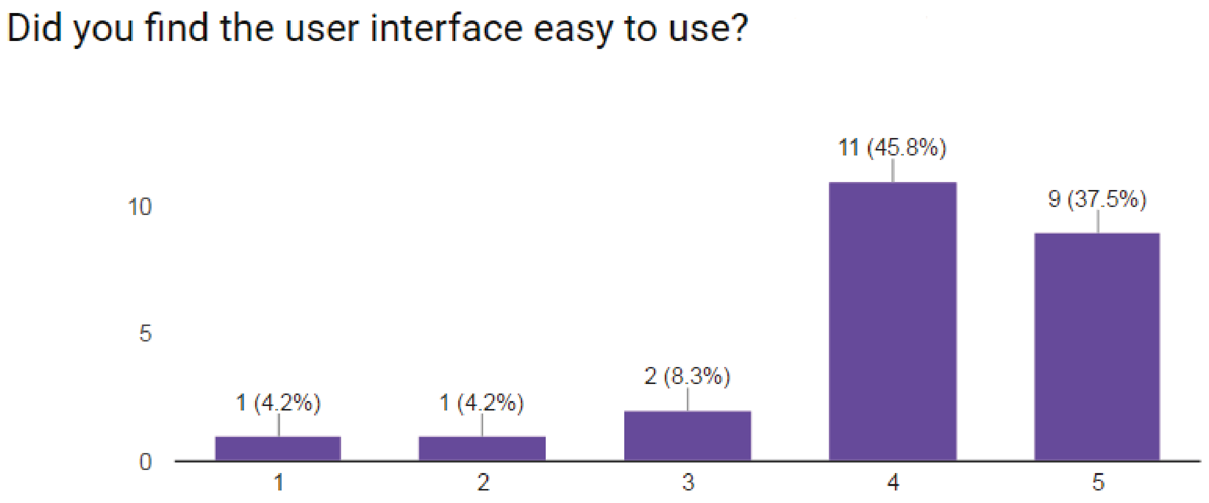


Figure 6: Response to the question "Did you find the user interface easy to use?"

On the whole, about 83% of the respondents did find the interface for the game easy to use. Taking into account the previous two questions, finding three respondents who did not find the user interface easy to use is not a big surprise. With a better design to the questionnaire then valuable data on why these three respondents did not find the user interface easy could have been captured. Again the two respondents who didn’t find it easy to use may well be the same two participants highlighted in figures 3 and 5.

5. Conclusion

Although, this is a limited pilot study with only 42 students, the results, coupled with the anecdotal comments from the students, does provide confidence that there is merit in this approach and suggests that it would be worth expanding the study to a large cohort undertaking a wider range of computing programmes. Ideally, this study would be replicated with the entire computing department first year cohort to ascertain the validity of the premise that teaching programming without using a programming language will help towards enhancing the student's problem solving skills and their understanding of the fundamental programming building blocks.

One comment in particular stood out as highlighting what the game and the study was intending to do which was to get the students to approach programming not from a language oriented perspective but a problem solving and pseudocode/algorithm building perspective.

> *"The Game forced me to visualise the steps the robot was taking and think what steps and rotations it needed to navigate."*

As can be seen from the student comments below, the students appreciated the purpose of the study and understood what the game was trying to teach them.

> *"This is a very intuitive and useful little game, that should be used to introduce programming to people."*
>
> *"I think this is a great way to showcase coding from a different perspective."*
>
> *"I believe the game can be very helpful to new programmers as it is a fun and easy way to explain loops and if statements in a non-complicated way."*
>
> *"I do believe it succeeds in doing what it is supposed to and exactly to show these very key features of programming in a fun and engaging way."*

From the brief pilot study undertaken there are clear signs that the students have engaged with the chosen game and the underlying reasoning behind it. The implementation of the study leaves analysis of the results subjective, therefore, open to interpretation. However, based on anecdotal evidence, i.e. the comments made by students, and from interactions with the students during the lab, this would suggest that the students had a better understanding of the fundamental programming building blocks of sequence, selection and iteration. Hence, the initial hypothesis was achieved to some extent.

Repeating the study in the next academic year with a tighter focus on the data gathering survey should allow a qualitative analysis to take place providing a clearer picture of the effectiveness of the chosen game at relaying the concepts of sequence, selection and iteration and the premise that programming can be introduced, initially, without the use of any formal programming language.

References

ACM & IEEE, 2013. Computer Science Curricula 2013.

Coller, B.D. & Scott, M.J., 2009. Effectiveness of using a video game to teach a course in mechanical engineering. *Computers \& Education*, 53(3), pp.900–912.

Fincher, S., 1999. What are we doing when we teach programming? In *Frontiers in Education Conference, 1999. FIE'99. 29th Annual*. pp. 12A4–1.

Garris, R., Ahlers, R. & Driskell, J.E., 2002. Games, motivation, and learning: A research and practice model. *Simulation \& gaming*, 33(4), pp.441–467.

Ibrahim, R. et al., 2010. Students Perceptions of Using Educational Games to Learn Introductory Programming. *Computer and Information Science*, 4(1), p.p205.

Kazimoglu, C. et al., 2012. Learning programming at the computational thinking level via digital game-play. *Procedia Computer Science*, 9, pp.522–531.

Lahtinen, E., Ala-Mutka, K. & Järvinen, H.-M., 2005. A study of the difficulties of novice programmers. In *Acm Sigcse Bulletin*. pp. 14–18.

Lau, W.W. & Yuen, A.H., 2009. Exploring the effects of gender and learning styles on computer programming performance: implications for programming pedagogy. *British Journal of Educational Technology*, 40(4), pp.696–712.

Law, B., 2016. Puzzle Games: A Metaphor for Computational Thinking. In *10th European Conference on Games Based Learning: ECGBL 2016*. p. 344.

Leutenegger, S. & Edgington, J., 2007. A games first approach to teaching introductory programming. *ACM SIGCSE Bulletin*, 39(1), pp.115–118.

Linn, M.C. & Dalbey, J., 1985. Cognitive consequences of programming instruction: Instruction, access, and ability. *Educational Psychologist*, 20(4), pp.191–206.

Milne, I. & Rowe, G., 2002. Difficulties in learning and teaching programming—views of students and tutors. *Education and Information technologies*, 7(1), pp.55–66.

Rajaravivarma, R., 2005. A games-based approach for teaching the introductory programming course. *ACM SIGCSE Bulletin*, 37(4), pp.98–102.

Robins, A., Rountree, J. & Rountree, N., 2003. Learning and teaching programming: A review and discussion. *Computer science education*, 13(2), pp.137–172.

Shneiderman, B. & Mayer, R., 1979. Syntactic/semantic interactions in programmer behavior: A model and experimental results. *International Journal of Parallel Programming*, 8(3), pp.219–238.

Tan, P.-H., Ting, C.-Y. & Ling, S.-W., 2009. Learning difficulties in programming courses: Undergraduates' perspective and perception. In *Computer Technology and Development, 2009. ICCTD'09. International Conference on*. pp. 42–46.

Vahldick, A., Mendes, A.J. & Marcelino, M.J., 2014. A review of games designed to improve introductory computer programming competencies. In *Frontiers in Education Conference (FIE), 2014 IEEE*. pp. 1–7.

Winslow, L.E., 1996. Programming pedagogy—a psychological overview. *ACM Sigcse Bulletin*, 28(3), pp.17–22.