Computers & Education: X Reality 4 (2024) 100061

Contents lists available at ScienceDirect

Computers & Education: X Reality

journal homepage: www.journals.elsevier.com/computers-and-education-x-reality

A Serious game for programming in higher education

Thomas Hainey *, Gavin Baxter

University of the West of Scotland, United Kingdom

ARTICLE INFO

Keywords:
Formative assessment
Revision
Computer game
Serious game
Evaluation
Programming education

ABSTRACT

Programming is a highly difficult skill which is a constituent of many undergraduate programmes at Higher Education (HE) level. With the advancement of games technology there is an increasing opportunity for educators to provide innovative assessment tools for students on their courses which are highly immersive and graphically indicative of the times. This could potentially be in a supplementary capacity or to a greater extent inextricably linked to the learning outcomes and assessment outcomes. Notably serious games and Games-Based Learning (GBL) have received high levels of attention from educationalists due to being motivational, novel learning approaches. This paper will outline two empirical studies conducted to develop a game to teach programming at HE level. The first study will gauge the acceptability of a computer game for teaching programming and formulating content integration development requirements. The second study will outline the evaluation of the developed game being placed in a module as a formative assessment tool to assist learners to revise for their formal class test. Study one showed that acceptability of the game was high with 61 participants completing an acceptability/content integration questionnaire. The game was designed to consolidate knowledge on rudimentary and advanced programming concepts, data structures and algorithms. 48 participants evaluated the game in study two with the results generally indicating that they enjoyed playing the game as a revision alternative with 14% of participants rating it as very effective and 51% of participants as effective for allowing them to prepare for their class test. The majority of participants also believed that games could be utilised in a formative and summative assessment capacity on courses for independent study.

1. Introduction

Computer Programming is a fundamental skill across many Computing Programmes at HE including: Computer Science, Software Engineering and Computer Games Development/Technology. Programming is a highly useful skill that can lead to a very rewarding career, but two problematic issues exist: Firstly, programming is perceived as a difficult, challenging skill to learn where lifetime development and commitment is required taking almost a decade to turn a novice programmer into an expert. Secondly, programming courses have some of the highest dropout rates where it is suggested that traditional teaching approaches such as lecturing, role-playing, lab tutorials and paper-based case-studies are not sufficiently effective in suitably preparing graduates for employment. Serious games have been empirically evaluated qualitatively or quantitatively in several areas including: business, computing, social issues and health (Boyle et al., 2016; Connolly, Boyle, MacArthur, Hainey, & Boyle, 2012) at various levels of education including Primary Education (Hainey, Connolly, Boyle, & Wilson, 2016). There is however a lack of specific empirical evidence in the serious games field in terms of game types/genres being applied to different groups requiring particular pedagogical content (Hays, 2005). Serious games are a potential solution to some of these issues as they are regarded as a new innovative teaching approach capable of engaging learners and revitalising interest in a difficult field of study via immersive experiential learning. It has been noted that research is lacking in terms of how computer games are used in teaching with regards to acceptance by learners and specifics regarding referred curricula and pedagogical content in terms of content integration (Backland & Hendrix, 2013). Integration of content and a lack of development frameworks and models is a challenge for serious games/GBL in terms of validity. Nevertheless, educational institutions are encouraged to utilise novel, innovative, technological approaches to engage learners, enhance the educational experience and improve retention and pedagogical effectiveness.

The academic literature has many examples of games for the purposes of teaching programming with a large number of GBL applications

* Corresponding author.
E-mail address: thomas.hainey@uws.ac.uk (T. Hainey).

https://doi.org/10.1016/j.cexr.2024.100061
Received 3 October 2023; Received in revised form 3 February 2024; Accepted 18 March 2024
Available online 18 April 2024

and games for the purposes of teaching aspects of Computer Science and Software Engineering. A rich and diverse collection of games exist including board games (Jordaan, 2018), online interactive tools (Berkling & Thomas, 2013), mobile phone applications (Boticki, Barisic, Martin, & Drljevic, 2012), console games (Morsi, Richards, & Rizvi, 2010), multiplayer online games (Xiao & Miller, 2014), games-based construction environments (Doss, Juarez, Vincent, Doerschuk, & Liu, 2011), PC games (Connolly, Stansfield, & Hainey, 2007), collaborative or competitive gamification approaches (Sheth, Bell, & Kaiser, 2013), games in collaborative virtual environments such as Second Life (Yap, 2011) and MMORPGs (Malliarakis, Satratzemi, & Xinogalos, 2014).

While many of the games that are customised and specially implemented for teaching a highly specific topic, there are examples of extremely popular commercial games being adapted and utilised for teaching. For example: *Super Mario Brothers* to teach AVL Trees, rotations, balancing, searching, insertion and deletion of nodes (Wassila & Tahar, 2012) and *Bomberman* for teaching procedural programming in C (Chang, Chou, & Chen, 2010). While games in the literature cover a broad and diverse range of topics in software engineering and computer science. Some of the games specifically relating to programming education covered the following topics:

- Object Oriented Programming Concepts - classes, inheritance and polymorphism (Lotfi & Mohammed, 2018; Alhazbi & Ismail, 2010; Wong, Yatim, & Tan, 2015; Malliarakis et al., 2014).
- Syntax, data types and algorithm design, variables, if-statements and loops (Anderson & McLoughlin, 2007; Masso & Grace, 2011; Malliarakis et al., 2014).
- Sequence, selection, iteration and functions/methods (Cagin, 2013; Law, 2012)
- Programming logic and simple coding (Chang, 2010; Jiau, Chen, & SSu, 2009; Finkelstein, Nickel, Harrison, Suma, & Barnes, 2009).
- Algorithms and data structures (Browne & Anand, 2013; Chaffin, Doran, Hicks, & Barnes, 2009; Lawrence, 2004; Shabana & Chen, 2009; Wei, Chen, & Doong, 2009).
- Debugging code (Deitz & Buy, 2016; Miljanovic & Bradbury, 2017).
- Software and code visualisation (Marques, Levitt, & Nixon, 2012).
- Pointers (Karapinar, Zavarak, Senturk, Kara, & Erdogmus, 2012).
- Array and multi-dimensional array visualisation (Baker, Zhang, & Caldwell, 2012).
- Decomposition of complex programming tasks (Khenissi, Essalmi, & Jemni, 2013).

There are examples of a number of game-based construction and GBL applications for the purposes of teaching programming indicating great graphical diversity and fidelity. Malliarakis et al. (2014) performed a review of a number of games including Catacombs (a 3D multiplayer game to teach variable declaration, nested conditionals, structures and loops), Saving Sera (a 2D game to teach control structures and recursion), EleMental (a 3D game to teach recursion), Gidget (a web-based game to teach algorithm design and analysis) and Wu's Castle (a 2D game for teaching loops, arrays and parameters). Vahldick, Mendes, and Marcelino (2014) in their review of games to improve introductory computer programming competencies identified 40 games categorising them based on the following criteria: type, platform, competence, topics and language.

Teaching practical computer programming concepts in Higher Education (HE) can be a very challenging task for educationalists. There is a diverse range of learners with regards to proficiency, confidence and programming experience. These learners can come from Secondary Education (SE), Further Education (FE) institutions, be international or mature learners. This problem is exacerbated by increasing pressures placed on universities in Scotland to move articulation levels from 49% to 75% in the next decade (Scottish Funding Council, 2016) meaning that students can gain access to realistically the first 3 years of HE from a college level route. This can lead to computer programming classes at Universities having a range of learners with vastly different previous knowledge and experience. One additional problem is that traditional teaching approaches such as lecturing, practical labs, paper-based course works and role playing may not be sufficient to cater for every type of learner in terms of preparation for careers in the computer games/software industry. The Scottish Government (2016) published a digital and learning teaching strategy report with the objectives of ensuring the consideration of digital technology in curriculum and assessment delivery areas and to drive innovation in digital technology for learning and teaching. Additionally, the Scottish Qualifications Authority (SQA) (2018) published a report written by young people across Scotland setting out ideas and recommendations on the future of how assessment of qualifications take place where an identified key theme was involving learners in the design of assessment and assessment systems. Lecturers are therefore encouraged at tertiary education level to use innovative digital technologies in teaching, learning and assessment in the hope of improving student retention, engagement, pedagogical effectiveness and to maintain excellent and contemporary professional practice. This study will investigate the acceptability and design stages of integration of games-based learning (GBL) or serious games (a particularly interactive digital technology) into a HE context for computer programming education. The study will produce a framework to address the issues of integrating a computer game for programming education into a real classroom context. This will be formulated by taking into account the problems associated with programming education and other applications that have been developed to address move motivational learning utilising games and novel approaches. The requirements for the application will be derived from a survey methodology and the existing academic literature. A game will then be designed and developed from the first survey study and will be evaluated in a classroom setting in an HE programming class to ascertain if it can be a useful formative assessment tool in a supplementary learning capacity.

2. Research rationale, questions, methodologies and developmental design

2.1. Research rationale

University computing courses generally have at least one or perhaps more programming modules and students tend to find these modules to be challenging as programming has a reputation of being a difficult skill to learn. Year 1 and 2 of computing courses at university have the highest levels of attrition (Chaffin et al., 2009; Jantan & Aljunid, 2012) with computer programming courses being a major contributory factor. Learners can struggle to grasp the most basic of programming concepts and skills (Deitz and Buy (2016, p. 37). In an attempt to address this, this study is a preliminary step in attempting to design and integrate a novel digital teaching approach into a computer programming course based on user acceptability and requirements at HE level and will be of interest to the following parties for the following reasons:

- Educationalists in general as a framework will be presented on integration of a new novel teaching and learning approach taking into account content and assessment integration.
- Computer scientists as the study investigates computer programming specifically identifying problematic areas of learning and where a digital technology can be best applied for these areas.
- GBL and serious game researchers as empirical evidence will be presented in a practical application in terms of ascertaining requirements from learners and evaluating the serious game in terms of learning effectiveness.
- Government and educational authorities specifically the Scottish Government and the Scottish Qualifications Authority (SQA) as the study will base the requirements for the GBL application on learner preferences.

- Higher Education Institutions (HEIs) can use this study as a pedagogical blueprint for the integration of GBL into computer programming education and to guide the incorporation of digital technologies into other courses potentially.

This paper will outline the full development cycle of a game to teach programming consisting of two empirical studies. Study one will gauge overall acceptability of a programming game and identify content and general requirements. Study two will allow participants to utilise the game as a formative assessment tool and evaluate the game ascertaining if it can be an effective formative assessment supplementary tool at HE level.

The objectives of the study are the following:

- Perform a literature review to ascertain what empirical evidence exists associated with serious games and programming education and also to identify problems associated with traditional teaching approaches for programming.
- Perform a survey (empirical study 1) to ascertain if a serious game for programming education will be accepted in a HE setting, what particular aspects of programming that students find to be difficult, what aspects of programming would be best to incorporate into a serious game and what format and type of game would be more suitable to implement.
- Utilise empirical study one and the literature to formulate a framework for content integration for programming education games and directly integrate the pedagogical content into the game.
- Develop a game with the programming content integrated into a gamified environment.
- Perform an intervention, post-test experiment (empirical study 2) to ascertain if the developed serious game can be utilised as a formative or summative assessment tool in a programming class in HE.

2.2. Research questions

This paper has the following research questions:

Research Question 1: *What are the main problems associated with traditional programming teaching approaches?*
Research Question 2: *How have computer games been used for the purposes of teaching computer programming in the past?*
Research Question 3: *What issues are identified in terms of pedagogical content integration for computer games teaching programming? What existing frameworks (if any) are there?*
Research Question 4: *To what extent will a GBL approach be accepted by learners for teaching computer programming?*
Research Question 5: *What programming pedagogical content would participants like to see in a game to teach programming at HE level?*
Research Question 6: *Can a computer game be effective as a formative assessment tool for computer programming in HE?*

2.3. Selection of research methodology

The research methodologies selected are archival research in the form of a systematic literature review, survey research and experimental post-test design with non-parametric statistical analysis techniques. The archival systematic review is more suitable to ascertain what problems associated with teaching computer programming exist in the literature, how computer games have been utilised teaching computer programming, issues associated with pedagogical content associated with serious games and programming education and whether there are any existing frameworks. A systematic literature review of relevant databases is the most efficient method to answer the first three research questions.

Research question 4,5, are attempting to ascertain a number of requirements for the proposed GBL environment/serious game to be produced in terms of whether that game will be accepted by the target learners, what particular concepts the learners would like to see integrated into that game and what type of game should be produced in terms of genre, platform, and graphical fidelity. To answer these questions, the most efficient method to obtain a large amount of data from the participants was a survey/questionnaire methodology to perform numerical analysis for potentially more generalizable results.

Research question 6 is attempting to ascertain the effectiveness of a serious game for teaching programming concepts and to answer the question of whether it can be used as a formative assessment mechanism/tool in Higher Education. It was decided that an experimental post-test design would be utilised with non-parametric statistical tests being utilised.

This is primarily a quantitative research study with a positivist paradigm being adopted due to the researchers Computer Science and research background. The research questions and the selected methodologies for answering each research question are listed below.

2.4. Research methodologies

Research question 1, 2 and 3 will be answered utilising archival research using the following search terms: *("computer games" OR "video games" OR "serious games" OR "simulation games" OR "games-based learning" OR "online games" OR "MMOG" OR "MMORPG" OR "Virtual" OR "Augmented") AND ("program*" OR coding OR "computer* science")*

The following relevant electronic databases were searched: ACM, EBSCO (including CINAHL Complete, eBook Collection, Education Source, Library, Information Science& Technology Abstracts, PsycARTICLES, PsycBooks, Psychology and Behavioural Sciences Collection, SocINDEX), Ethos, IEEE, IngentaConnect, ScienceDirect and Web of Science and SpringerLink and the grey literature. There has been a near exponential increase in the publication of GBL studies over the past five years, so it was necessary to limit the search to titles of papers and abstracts when cross referencing the search terms. As a result, the number of papers found in each search engine are displayed in Table 1.

The papers were then identified, screened, considered for eligibility and then included in the final analysis utilising a preliminary PRISMA methodology shown in Fig. 1. (see Fig. 2)

Research question 4 and 5 will utilise the questionnaire methodology to reach a wide and diverse range of participants in an efficient cost effective manner. The questionnaire was constructed utilising a number of open-ended, close-ended and matrix questions utilising appropriate academic guidelines (Blair, Czaja, & Blair, 2013) and modelled on a previous questionnaire produced to establish the motivations for playing computer games (Hainey, Connolly, Stansfield, & Boyle, 2011b). The pedagogical content for the questionnaire was constructed utilising a number of programming books (including Dawson, 2014; Malik, 2014), subject matter experts and utilising a review of the content of three HE programming modules at the university including: Introduction to Programming, Programming with Objects and Software Development for Games. The questionnaire was constructed in SurveyMonkey and emailed to Creative Technology students at the University. The survey was available for a two week period in April of 2019. The full questionnaire is available in *Appendix 1*.

Table 1
Systematic literature review relevant papers.

Search Engine	Total Papers	Relevant Papers
ACM	1233	29
EBSCO	1484	6
Ethos	93	1
IEEE	776	251
IngentaConnect	1176	19
ScienceDirect	2489	26
Web of Science	2739	23
SpringerLink	548	0
Totals	10538	364

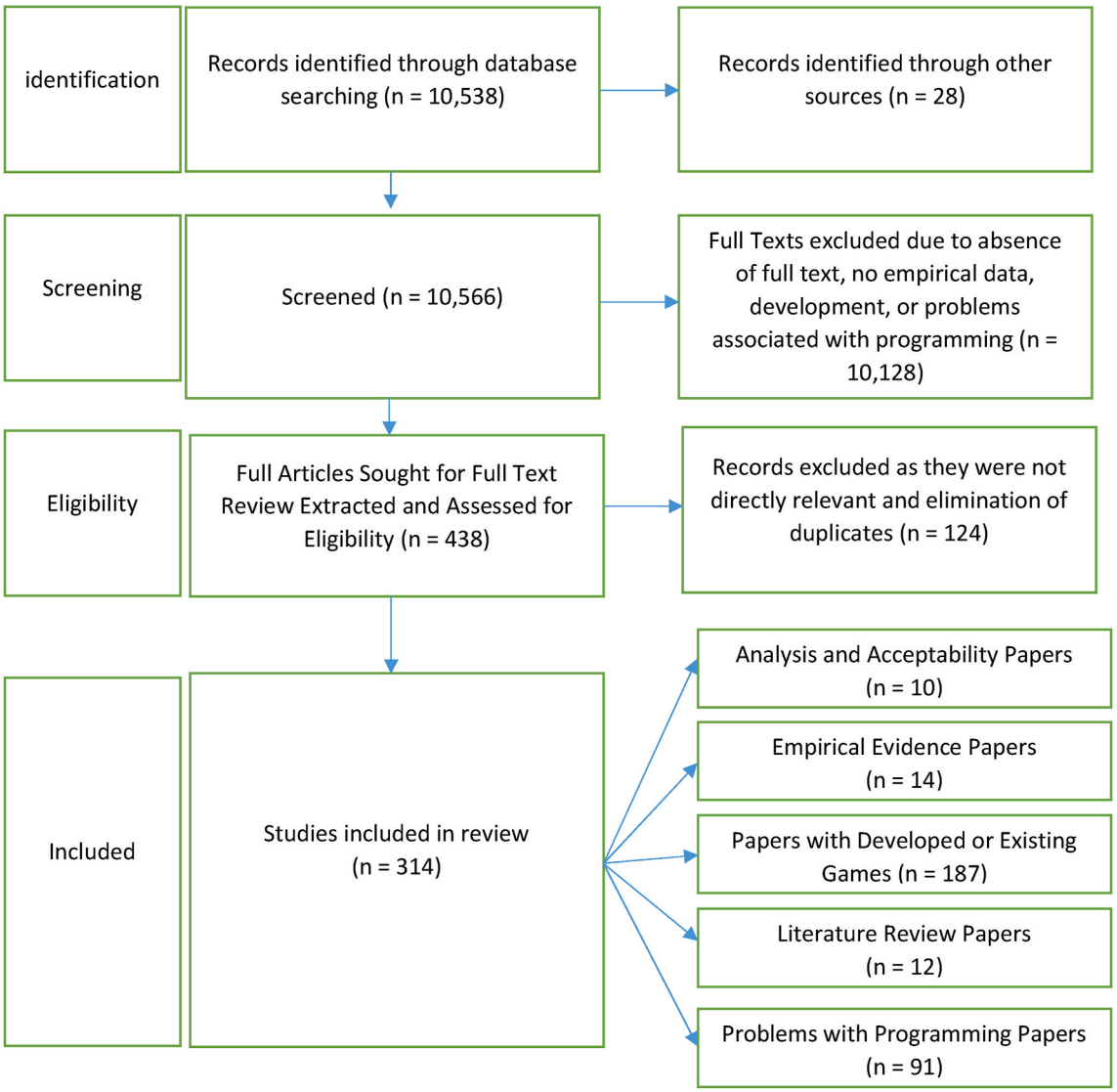

Fig. 1. PRISMA Methodology for relevant paper inclusion.

Research question 6 will utilise a post-test experimental group design and will employ a similar questionnaire to evaluate the game. The game was made available to a second year programming module at University in a supplementary, formative assessment tool capacity via email. After the game has been played the participants are invited to complete a post-test evaluation questionnaire on SurveyMonkey. This questionnaire was of a similar design as the previous questionnaire and is available in *Appendix 2*.

2.5. Developmental design

This paper will outline the entirety of a first developmental cycle and includes the following stages:

- Ascertaining the pedagogical difficulties with programming.
- Gauging acceptability and development requirements (Empirical Study 1)
- Formulating a general pedagogical content integration framework for a programming serious game.
- Implementation of the game
- Evaluating the game as a formative assessment tool in a course (Empirical Study 2)

3. Background research

3.1. Pedagogical difficulties associated with computer programming education

Computer programming is a highly complex skill with many pedagogical issues and a considerable amount of time is required to effectively learn that skill and maintain it. Rigorous computer programming has been introduced by the Department for Education (DfE) in recent years with programming being proposed as a skill that everyone should have with codeacademy.org and code.org indicating that it is a glamorous path to employment (Lee et al., 2014). Hwang et al. (2010) note that almost all students enrolled in a department related to the computer field (i.e. computer science or information management systems) are required to take a programming class at some point. Despite the fact that employment opportunities for Computer Scientists are rising, the number of students enrolling on and graduating from Computer Science related degrees is declining with the majority of attrition taking place in the first two years of a degree programme (Chaffin et al., 2009) and fewer students are actually choosing a career in Computer Science or Engineering (Marques et al., 2012). There is a disparity between numbers of graduates, graduates in employment and unfilled positions in related industries (Jordaan, 2018).

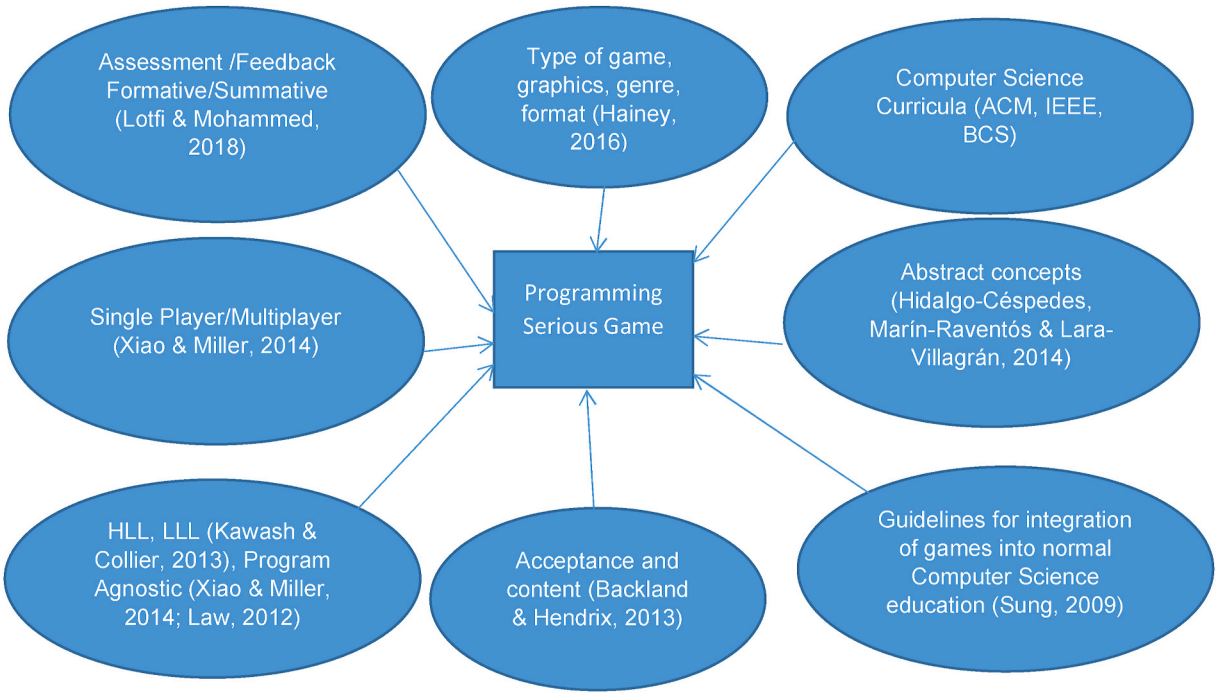

Fig. 2. Content integration considerations for a game to teach computer programming.

The literature review identified numerous issues associated with computer programming with the result being a "*programming incompetency*" as graduates do not meet industry expectations in terms of competence and capabilities (Kumar & Sharma, 2018). Sun, Phelps, Egert, and Bayliss (2009) highlight that teaching introductory programming is a multifaceted problem encompassing: issues of abstraction, algorithmic thinking, software construction, and debugging strategies. Kaučič and Asič (2011, p. 1) state that mastering programming related competencies demands "*acquisition of previously completely unknown knowledge, problem solving skills and strategies, design strategies and approaches wrapped by implementation abilities*". Computer programming courses are perceived as extremely difficult with high levels of attrition (Wei et al., 2009; Chaffin et al., 2009; Jantan & Aljunid, 2012; Lee et al., 2014). Learners can struggle to grasp the most basic of programming concepts and skills. For example Deitz and Buy (2016, p. 37) state that "*novice programmers often struggle with many concepts underlying computer programming*" such as variables and the misinterpretation of program states and variable states remains a frequent source of errors for novice programmers. Cagin (2013) summarised the literature and stated that there is a lack of rigorous empirical research associated with the difficulties experienced by introductory programmers. Some of these difficulties are:

- Confusion with branching and locations within programming logic where the explanation of different programming segments and the combined whole is considered educationally beneficial.
- Difficulties in the composition and coordination of programming components.
- Understanding error messages to appropriately debug programs.
- Thinking in programming syntax, as this requires thinking in an operational level of abstraction before the production of code in a specific language can take place.
- Difficulty in the visualisation of programming constructs for given problems.
- Learners require well-structured timely feedback in response to their actions.

In terms of specific topics within Computer Science related to programming education, some of the following topics are considered to be problematic for learners: algorithms due to abstract mathematical theories, modelling complicated concepts and possibly describing complicated changes in dynamic data structures with the intent of solving complex problems (Shabanah, Chen, Wechsler, Carr, & Wegman, 2010). Novice programmers also find it incredibly difficult to decipher code and to determine what it is doing, i.e., programming comprehension (Miljanovic & Bradbury, 2017). Yan (2009) notes that these problems are not really associated with prior computing knowledge but because the subject is considered to be difficult, there is a lack of problem solving skills and an appreciation of abstract concepts and logical thinking (Khenissi et al., 2013). The difficulties are experienced from both a learner and a teacher perspective as these problems are exacerbated with the addition of abstract concepts associated with Object Orientation which is the industry standard programming paradigm where programs are constructed of interacting objects. Rudimentary programming education is not just about abstract Object-Oriented concepts and practical programming alone but rather the interactions and interweaving of the two. Writing a computer program can require a great deal of implicit knowledge beyond the scope of simply being familiar with a specific language (Hwang et al., 2010). Learners generally begin introductory programming classes by producing text-based applications which they consider to be relatively unimpressive. Mitamura, Suzuki, and Oohori (2012) points out two key points in relation to learning programming, firstly that the learning experience differs significantly in relation to academic background and secondly, that learners sometimes do not experience enjoyment with traditional teaching approaches. Traditional teaching approaches in relation to programming have a reputation for lacking engagement (Berkling & Thomas, 2013) and do not provide ways to think *outside of the box* (Jayasinghe & Dharmaratne, 2013). They provide low motivation and there can be an issue of generational gaps between teachers and students (Khenissi et al., 2013). Kaučič and Asič (2011) highlighting that traditional teaching approaches for programming education tend to be seen as dull and uninteresting as a result of teachers prioritising theory rather than practical work. A number of approaches are highlighted to overcome some of the issues with traditional education such as pair programming, individual/collaborative work, read before write approach

and the use of games (Kaučič & Asič, 2011). De Aquino Leal and Ferreira (2013) believe that difficulties learning programming are possibly due to deficient training at elementary level or misconceptions in relation to the course they are entering as motivated users of computer technology select a course based on this fact. Tan, Ting, and Ling (2009) performed a survey asking 182 participants undertaking an undergraduate course on computer programming about their level of understanding of certain rudimentary and advanced concepts. The participants rated variables, decision logic structures, iteration logic structures, error handling, arrays and parameters as the easiest concepts to understand and rated all of the object oriented principles (polymorphism, inheritance encapsulation), pointers, abstract data types and recursion as the most difficult concepts to understand.

It was considered important to gain a greater appreciation of the advantages, disadvantages and suitability of the main teaching approaches utilised in software engineering and by implication programming education in this study. Table 2 attempts to summarise a review performed by Hainey (2010) and Macphail, Hainey, and Connolly (2012) citing the following sources: ADPRIMA, 2019, Andrianoff and Levine (2002), Bernstein and Klappholz (2001), Barrett (1997), Biggs (1999), Bonwell (1996), Cashin (1985), Cope and Horan (1996), Connolly, Stansfield, McLellan, Ramsay, and Sutherland (2004), Davis (2001), Polack-Wahl (1999), Simsarian (2003) and Shaw and Dermoudy (2005).

3.2. Synthesis and presentation of model framework

As well as considering the integration of learning content, there are a number of questions and issues to be addressed when considering the basic requirements of a serious game identified in some of the following categories:

3.2.1. Assessment

Assessment integration is one of the key challenges of serious games where it is necessary to consider what particular types of assessment mechanics are incorporated into a serious game. Assessment can be categorised into formative and summative assessment. Conole and Warburton (2005) state that formative assessment is *"when the evidence is actually used to adapt the teaching to meet student needs*". Summative assessment is defined by the QAA (2012) as "*used to indicate the extent of a learner's success in meeting the assessment criteria used to gauge the intended learning outcomes of a module or programme."* Lotfi and Mohammed (2018) address assessment in a mobile game for teaching Object Oriented Programming principles in the context of using the assessment data to ensure that the learning outcomes of the game are actually met and raise the question of whether this assessment should take place at the end of the game in a summative fashion or throughout the game in a formative fashion.

3.2.2. Type of game and player dynamic

One important distinction in relation to computer games is whether the game should be played in a single-player or a multi-player capacity and determining what mode is preferable to the players. Many serious games are generally single-player mode due to the implementation complexity, however research indicates that players behave differently when playing in a single-player mode than in a multi-player mode (Xiao & Miller, 2014; Siu, Guzdial, & Riedl, 2017).

3.2.3. Type of programming language

When designing a game for programming education it is important to ascertain what particular programming paradigm and language should be the focus of the game e.g. should the game focus on Pascal, C++, C# or Visual Basic? Programming languages have now gone through 5 generations where one main distinction is High Level Languages (HLL) and Low Level Languages (LLL) (Kawash & Collier, 2013). Alternatively as a result of programming education resulting in transferable skills it is

Table 2
Attributes of traditional teaching approaches.

Attributes	Lectures	Role-play	Paper-based case studies	Live-through case studies
Instructor has control of the learning, has to be an effective speaker and communicate the intrinsic interest of a subject through enthusiasm.	HIGH	MEDIUM	LOW	LOW
Participants are actively involved in the exercise provide experiential learning and deeper understanding to have a considerable lasting effect.	LOW	HIGH	MEDIUM	LOW
Can be time consuming and therefore not always practical.	MEDIUM	MEDIUM	MEDIUM	HIGH
Scales with larger numbers of students and/or quantity of material	HIGH	LOW	HIGH	HIGH
Can provide a model of how professionals address problems and questions.	HIGH	MEDIUM	LOW	HIGH
Can provide elements of variety, reality and specificity.	MEDIUM	HIGH	MEDIUM	MEDIUM
Likelihood of transfer to the real world is improved.	MEDIUM	HIGH	MEDIUM	MEDIUM
Appeals to learners who learn by listening.	HIGH	HIGH	LOW	LOW
Appeals to learners who learn by reading.	HIGH	LOW	HIGH	HIGH
Appeals to learners who learn from demonstration.	LOW	HIGH	LOW	LOW
Appeals to learners who learn from discussion.	MEDIUM	HIGH	LOW	LOW
Appeals to learners who learn from practice by doing	LOW	HIGH	HIGH	MEDIUM
Presents a risk for learners for example, tension or embarrassment.	LOW	HIGH	LOW	LOW
Allows the learners to identify the mistakes of previous developers, thus reducing the possibility of making the same mistake.	MEDIUM	LOW	LOW	HIGH
Provides a safe environment to increase practice experience when real life experiences are unavailable, for example highlighting client frustration.	MEDIUM	HIGH	MEDIUM	LOW
Immediate feedback is provided.	LOW	HIGH	LOW	LOW
Can give new perspectives on situations.	MEDIUM	HIGH	MEDIUM	LOW
Learners can apply newly developed analytical problem solving skills in their own time to find a large number of solutions for complex issues.	LOW	MEDIUM	HIGH	HIGH
Maintains the attention of the learner.	LOW	HIGH	MEDIUM	MEDIUM
Suited for teaching abstract, complex	LOW	MEDIUM	MEDIUM	MEDIUM

(*continued on next page*)

Table 2 (*continued*)

Attributes	Lectures	Role-play	Paper-based case studies	Live-through case studies
subjects, or higher order thinking skills such as values, motor skills, analysis and application etc.				
Takes into account that learners are at different levels of understanding and learn at different paces.	LOW	MEDIUM	LOW	LOW
Puts pressure on the learner to perform.	LOW	HIGH	LOW	LOW
May require a debriefing session.	MEDIUM	HIGH	MEDIUM	MEDIUM
Has to be well planned, monitored and orchestrated or it may lack focus.	MEDIUM	HIGH	MEDIUM	MEDIUM
Puts pressure on instructor (s) to perform.	HIGH	HIGH	MEDIUM	LOW
Monitoring by a knowledgeable person to provide good feedback is required.	MEDIUM	HIGH	HIGH	MEDIUM
It is easy for the learners to see the relevance to their own situation.	MEDIUM	HIGH	HIGH	MEDIUM
Inappropriate results may occur if insufficient information provided.	MEDIUM	HIGH	MEDIUM	LOW

possible to have a programming game that is essentially program language agnostic (Law, 2012; Xiao & Miller, 2014).

3.2.4. Genre, format and level of graphical fidelity

Computer games are an incredibly versatile medium with a variety of different genres (Arsenault, 2009) including: action, adventure, shooter, fighting and racing. These genres can be delivered in a number of different formats such as: PC, Console, Mobile Device and Virtual Reality (Hainey, 2016) in different levels of graphical fidelity i.e. 2D or 3D (Gerling, Birk, Mandryk, & Doucette, 2013). Genre, format and level of graphics should be considered in relation to player preferences.

3.2.5. Recognised curriculum guidelines

A game created for the purposes of programming in HE should consider governing curriculum guidelines and bodies such as: The Institute of Electrical and Electronic Engineers (IEEE – https://www.computer.org), the British Computer Society (BCS – https://www.bcs.org) *and the Association for Computing Machinery (ACM* – https://www.acm.org*)*.

3.2.6. Integration of soft skills and abstract concepts

The integration of soft skills (e.g. teamwork, interpersonal relationships) and abstract concepts can be highly complex particularly with a subject that has the concept of abstraction (hiding complexity and information) as one of its key pillars (Hidalgo-Céspedes, Marín-Raventós, & Lara-Villagrán, 2014).

3.2.7. Appropriate integration into a module or course

When utilising a game to teach in any capacity in a course it is necessary to consider how to integrate the game/application into that course at the appropriate time (Ketamo, Kiili, Arnab, & Dunwell, 2013; Sung, 2009).

3.2.8. Acceptance and preferred content

Preferences, perceptions and attitudes to computer games for educational purposes and particular subject areas are extremely important when considering the design, implementation and evaluation of a serious game or GBL application. Establishing that the group of recipient learners will accept a game for a particular subject is beneficial and is also very important to ascertain the receiving groups requirements in terms of preferred content (Connolly, Stansfield, & Hainey, 2009; Backland & Hendrix, 2013).

Pedagogical Content integration is a key problem that exists because every serious game/GBL application is generally a custom built learning tool and general content integration frameworks are problematic to develop. The literature review revealed very little recent research performed particularly addressing pedagogical content integration for serious games. Aldrich (2004, 2005) categorises content into mainly simulation elements which are linear (a defined path from beginning to end), cyclical (addressing small competitive tasks combined to impact the overall environment), system (exposing the user to intertwined, complex relationships governing system operation) and open-ended (presenting a subject with no set experience). Gomez-Martin, Gomez-Martin, and Gonzales-Calero (2009) define intrinsic and extrinsic game content. Intrinsic content is a game in which learning content is integrated within the framework of the game. In this type of game, learning and gaming are closely related and both happen simultaneously. Extrinsic content is a game in which the scenario is separate or less integrated within the learning content. In this type of game, learning and gaming are independent activities. These categories are not an either/or but a continuum of possible options that complement different content styles. This is very similar and in line with Winn's (2009) categorisations of intrinsic endogenous games where the game play is informed by the learning content and pedagogical theory and exogenous games where the learning content is added on top of successful game mechanics without significant modification. General frameworks considering content integration also exist. Sauvé (2010) note in their content segmentation framework that an appropriate balance between game and learning time is required to maintain motivation and that major content segments should be defined in relation to learning objectives and target population. Echeverria et al. (2011) propose a classroom games design framework with an educational dimension based on Bloom's revised taxonomy focusing on cognitive processes and types of knowledge. Tang and Hanneghan (2010) propose the following guidelines for embedding pedagogical content into game-play:

- should be in line with defined learning objectives and assessable
- in increasing order of difficulty and achievable
- in the form of an intellectual exercise with minimal abstraction
- applicable and readily transferable to a real world scenarios
- Achievement should be based on Weiner's 'Attribution Theory' rather than pre-destined winning
- feedback should assist in success and error recognition
- be embedded within the storytelling/narrative of the game whenever possible
- use multiple representations where appropriate and be concise but avoid oversimplification
- contain challenges allowing learners to apply acquired knowledge and thus increase retention
- not have more than 7 key concepts in order to aid information recall.

Aspects to consider when integrating pedagogical content into a game for programming is summarised in Fig. 2.

4. Gauging acceptability and requirements (empirical study one)

4.1. Participants and methodology

61 participants completed the *Serious Games for Teaching Programming Concepts* questionnaire. 4 participants (6.6%) were female, 55 participants (90.2%) were male, 1 (1.6%) participant specified that they were non-binary/third gender and 1 participant (1.6%) preferred not to say. Participants played games for 17.54 h per week (SD = 10.43) with a range of 0–31 h. This is approximately 10 more hours per week than a general survey of the motivations for playing computer games in HE (Hainey, Connolly, Stansfield, & Boyle, 2011a) however; this is consistent as participants were on Computer Games related degree programmes therefore it was expected that they would play computer games more frequently. Participants performed 11.46 h (SD = 8.90) of independent programming study per week with a range of 0–31 h. A Wilcoxon-matched pairs signed rank test between the amount of hours spent playing computer games and the amount of time spent performing independent study of computer programming indicated that participants play computer games for a significantly longer period in a week rather than engaging in independent study ($Z = -3.025$, $p < 0.002$). Participants were an average of 21.77 years of age (SD = 4.84) with a range of 17–48 years. Participants had been learning computer programming for 3.11 years on average (SD = 2.33) with a range of 0–10 years. 43 participants (70.5%) were on the Computer Games Development degree stream, 9 participants (14.8%) were on Computer Games Technology and 9 (14.8%) were on Computer Animation Arts. 41 participants (67.2%) described themselves as a novice, 19 participants (31.1%) were intermediate level and 1 participants (1.6%) described themselves as an expert programmer.

The questionnaire was constructed utilising a number of rudimentary programming books (including Dawson, 2014) and subject matter experts and utilising a review of the content of three HE programming modules at UWS (University of the West of Scotland) including: Introduction to Programming, Programming with Objects and Software Development for Games. The questionnaire was constructed in SurveyMonkey and emailed to students on the Computer Games Development/Technology and Computer Animation Arts Degree streams. The survey was available for a two week period. The survey was analysed utilising quantitative statistical analysis i.e. Mean, Standard Deviation, and non-parametric statistical inferential tests where appropriate such as Mann-Whitney *U* tests. The full questionnaire is in *Appendix 1*.

4.2. Acceptability of games for teaching computer programming in HE

Participants were asked if they believed that a game to teach computer programming concepts at HE was a good idea. 46 participants (75%) answered 'yes' to this question indicating that computer games could be a potential teaching approach.

4.3. Rudimentary programming concepts and perceived effectiveness of games

Participants were asked to rate the potential effectiveness of computer games to teach rudimentary programming concepts. A game was considered most effective to teach conditionals, logical operators, variable types, loops, objects and arithmetic operators. The results are displayed in Table 3.

4.4. Advanced programming concepts and perceived effectiveness of games

Participants were also asked to rate the potential effectiveness of computer games to teach advanced programming concepts. The results are shown in Table 4.

Table 3
Effectiveness of a game to teach rudimentary programming concepts.

Rudimentary Programming Concept	Rank	Mean	SD
Conditionals	1st	4.10	0.77
Logical Operators	2nd	4.06	0.76
Variable Types	2nd	4.06	0.83
Loops	2nd	4.06	0.80
Objects	3rd	4.04	0.73
Arithmetic Operators	4th	3.96	0.83
Switch Statements	5th	3.94	0.80
Declaring Variables	5th	3.94	0.81
Statements and Sequences	5th	3.94	0.79
Algorithms	6th	3.92	0.92
Methods and Functions	7th	3.90	0.81
Variable Scope	8th	3.89	0.84
Relational Operators	9th	3.85	0.78
Returning Types with Functions	9th	3.85	0.83
Compound Conditions	10th	3.80	0.84
Working with Constants	11th	3.79	0.87
Parameter Passing/Overloading Functions	12th	3.70	0.88
Programming Syntax	12th	3.70	0.92
Using Enumerators	13th	3.64	0.85
References	14th	3.62	0.87
Pointers	15th	3.60	0.99
Smart Pointers	16th	3.55	0.95

Table 4
Effectiveness of a game to teach advanced programming concepts.

Advanced Programming Concepts	Rank	Mean	SD
Inheritance	1st	3.83	0.79
Defining classes, Constructors and Destructors	2nd	3.81	0.82
Setting Member Access Levels, Data Members, Member Functions, Scope	3rd	3.77	0.88
Encapsulation	4th	3.74	0.79
Static Data Members/Static Member Functions	5th	3.72	0.85
Polymorphism	6th	3.70	0.81
Static Object Instantiation	7th	3.68	0.84
Mutator Accessor Functions	8th	3.66	0.84
Composition	9th	3.64	0.85
ADTs, Container Classes, Iterators, Interfaces	10th	3.63	0.91
Aggregation/Association	11th	3.62	0.85

4.5. Data structures and perceived effectiveness of games

The participants reported the following data structures to be the least difficult to learn arrays, stacks, standard vectors and multi-dimensional arrays. The participants were also asked to rate the potential effectiveness of a game for teaching the data structures. The results are displayed in Table 5.

The results indicated that a computer game would be most effective at teaching: arrays, queues, stacks, multi-dimensional arrays and

Table 5
Effectiveness of a game to teach data structures.

Data Structure	Rank	Mean	SD
Arrays	1st	3.94	0.77
Queues	1st	3.94	0.84
Stacks	2nd	3.90	0.88
Multi-Dimensional Arrays	3rd	3.88	0.86
Standard Vector	4th	3.77	0.78
Game Trees and Minimax Trees	5th	3.73	0.96
Bi-Directional Graphs	6th	3.65	0.91
Trees	6th	3.65	0.93
Hash Tables	7th	3.64	0.90
Graphs	8th	3.62	0.95
Uni-Directional Graphs	9th	3.60	0.89
Bit Vectors	9th	3.60	0.90
Priority Queues and Heaps	9th	3.60	0.92
Binary Trees	9th	3.60	0.90
Binary Search Trees	9th	3.60	0.95

Table 6
Effectiveness of a game to teach algorithms.

Algorithm	Rank	Mean	SD
Basic Path Finding	1st	3.83	0.79
Bubble Sort	2nd	3.72	0.85
Merge Sort	3rd	3.70	0.88
Heap Sort	4th	3.68	0.89
Quick Sort	4th	3.68	0.89
Binary Search	4th	3.68	0.91
Robust Path Finding	4th	3.68	0.84
Radix Sort	5th	3.62	0.90

standard vectors. The results also indicated that computer games would be least effective for binary search trees, binary trees, priority queues and heaps and bit vectors.

4.6. Algorithms and perceived effectiveness of games

The following three algorithms were reported as the most difficult to learn: robust path finding, radix sort and basic path finding while the least difficult algorithms were bubble sort, quick sort, heap sort and merge sort. The participants were asked to rate the potential effectiveness of computer games for teaching the searching and sorting algorithms. The results are displayed in Table 6.

4.7. Suitability of game genre to teach programming

Participants were asked to assess the suitability of game genres for potentially teaching programming concepts. The results are shown in Table 7.

80% specified a single-player game and 20% specified multi-player. Participants were asked whether they thought the game should be 2D or 3D. The majority of the participants stated that it did not matter if the game was in 2D or 3D (36 participants). Participants were also asked what particular level of graphical fidelity they believed the game should be. The majority of participants (71%) specified that they would prefer medium graphical fidelity. Participants were asked to rate the suitability of different platforms/fromats for a game to teach programming concepts. The most suitable format for a game in this context was the PC. Table 8 shows the preferred format of participants.

The participants were asked to rate the importance of the technical aspects of a potential game where control mechanism and interface, clear goal structure and help and scaffolding were considered to be the most important aspects. The results are show in Table 9.

Table 7
Game genre ranked by suitability.

Game Genre	Rank	Mean	SD
Educational Game	1st	4.42	0.78
Puzzle Game	2nd	4.36	0.94
Simulation Game	3rd	4.29	0.82
Serious Game	4th	4.16	0.98
Adventure Game	5th	4.07	0.91
Turn-Based Strategy Game (TBSG)	6th	4.07	0.86
Role Playing Game (RPG)	7th	3.91	1.10
Text-Based Adventure Game	8th	3.89	1.01
Action Adventure Game	9th	3.84	0.98
Platform Game	10th	3.73	1.07
Board Game	10th	3.73	0.99
Real Time Strategy Game (RTSG)	11th	3.71	1.12
Action Game	12th	3.47	1.06
Stealth Game	13th	3.33	1.09
Rhythm Game	14th	3.24	1.23
Survival Horror Game	15th	3.00	1.21
First Person Shooter Game	16th	2.98	1.29
Massively Multiplayer Online Game (MMOG)	17th	2.80	1.31
Fighting Game	18th	2.75	1.33
Sports Game	19th	2.69	1.14
Racing Game	19th	2.69	1.16

Table 8
Preferred game format.

Format	Rank	Mean	SD
Personal Computer	1st	4.82	0.58
Tablet PC	2nd	4.02	0.75
Mobile Phone	3rd	3.84	1.00
Augmented Reality	4th	3.31	1.16
Virtual Reality	5th	3.20	1.29
Games Console	5th	3.20	1.12

Table 9
Technical aspects of computer game to teach programming.

Technical aspect	Rank	Mean	SD
Control Mechanism and Interface	1st	4.58	0.58
Clear Goal Structure	2nd	4.53	0.63
Help and Scaffolding	3rd	4.22	0.67
Navigation Inside the Environment	4th	4.20	0.79
Narrative and Dialog	5th	3.89	0.83
Sound	6th	3.71	0.87
Collaboration	7th	3.49	0.87
Graphics	8th	3.09	1.10
Realism of Environment	9th	2.44	0.92
Realism of Characters in the Environment	9th	2.44	0.87
Character Customisation	9th	2.24	1.00

Table 10
Preferred level of education.

Level of Education	Rank	Mean	SD
Secondary Education	1st	4.44	0.62
Primary Education	2nd	4.20	0.99
Further Education	3rd	4.07	0.89
Informal Private Education	4th	3.96	0.90
Higher Education	5th	3.84	1.13
Industrial Training	6th	3.69	1.33

4.8. Suitability of games for level of education

Participants were asked how suitable a game for teaching programming would be with regards to suitability for different levels of education. The results are displayed in Table 10.

5. CODE: S.P.A.C.E: a game to teach programming

5.1. Game pedagogical content development

The survey performed in empirical study 1 was utilised to determine the requirements for the game in relation to pedagogical content that it would be suitable to teach, the preferred format, genre of game and overall acceptability. The majority of participants (75%) believed that a game would be suitable to teach programming in HE. The top ranking attributes were then used to produce the requirements for the game e.g. for the game genre in Table 7, the top attributes were the following:

Game Genre	Rank	Mean	SD
Educational Game	1st	4.42	0.78
Puzzle Game	2nd	4.36	0.94
Simulation Game	3rd	4.29	0.82
Serious Game	4th	4.16	0.98
Adventure Game	5th	4.07	0.91

It was therefore decided that the game should be a mixture of these genres and it ended up being an educational, puzzle, simulation, adventure, story-based serious game.

The game was developed in Unity 5 with the incorporation of intrinsic and endogenous narrative content integration being the main philosophy meaning that the learning outcomes and the game goals are inextricably interconnected. Utilising a narrative based story provided a

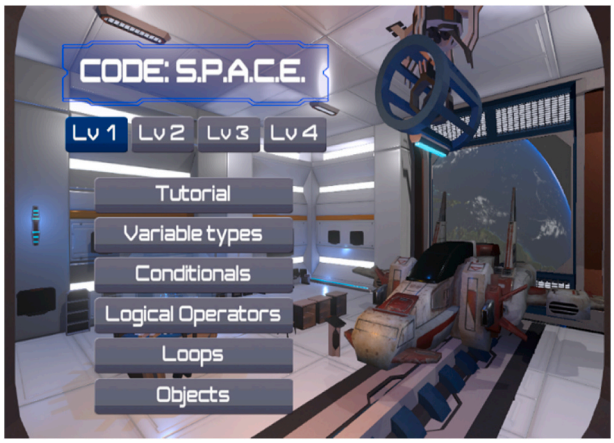

Fig. 3a. Menu screen.

game where the discourse provided immersion, engagement, motivation, and learning (Naul & Liu, 2020). The pedagogical content was integrated based on student preferences in relation to what the game would contain in the broad sense and the intricate learning content was taken from two undergraduate programming courses. One at year 1 and one at year 2 as these are considered to have the highest levels of attrition in Computer Science related courses. The game is a story-based space, puzzle adventure game requiring the player to navigate through an asteroid field where they are then confronted with a number of programming related questions. The development utilised some of the following guidelines from Tang and Hanneghan (2010) to embed pedagogical content into game-play:

- should be in line with defined learning objectives and assessable
  - •in increasing order of difficulty and achievable
  - •feedback should assist in success and error recognition
  - •be embedded within the storytelling/narrative of the game whenever possible
  - •contain challenges allowing learners to apply acquired knowledge and thus increase retention
  - •not have more than 7 key concepts in order to aid information recall.

The game is divided into 5 levels which allow the player to select the particular topic, there is then a screen that displays a tutorial and the player then shoots there way though waves of enemy spacecraft to have the opportunity to answer the questions to proceed. They are then presented with a question and they have to shoot the correct answer 10

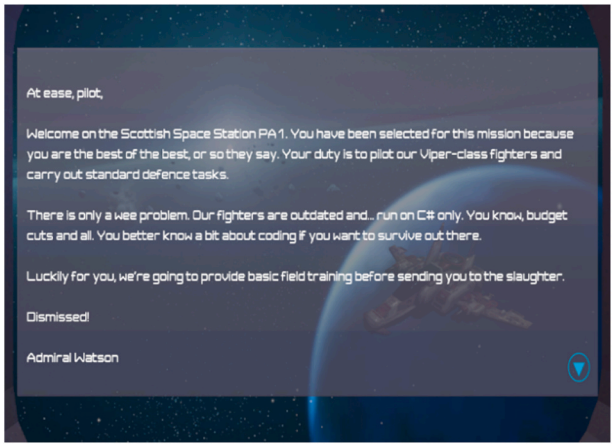

Fig. 3b. Narrative content.

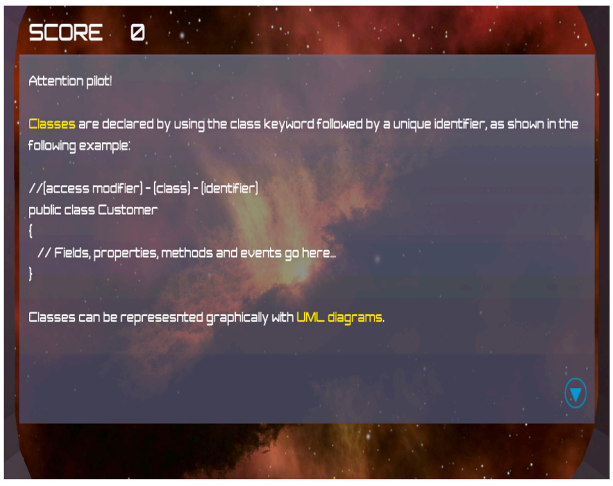

Fig. 4a. Class tutorial screen.

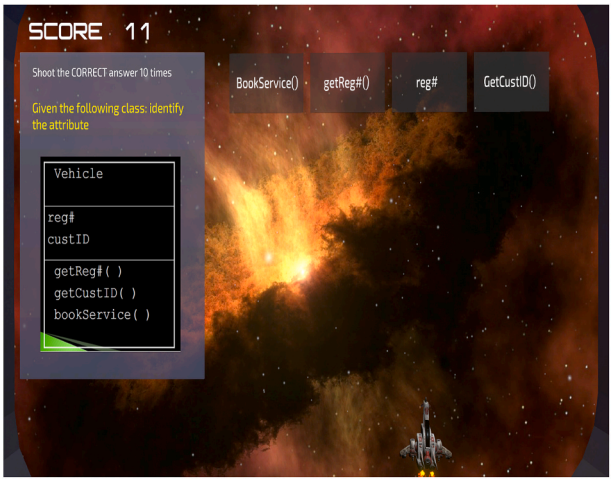

Fig. 4b. Class level game play.

times. An example of the sequential game screen shots are as follows: Fig. 3a and b illustrate the menu and a screen shot of the narrative within the game. Fig. 4a and b shows the class subject matter tutorial and the gameplay associated with the subject matter.

Level 1 was derived from Table 3 where the top rated preferred content in terms of perceived effectiveness for teaching rudimentary programming concepts were:

Rudimentary Programming Concept	Rank	Mean	SD
Conditionals	1st	4.10	0.77
Logical Operators	2nd	4.06	0.76
Variable Types	2nd	4.06	0.83
Loops	2nd	4.06	0.80

Ten questions were devised on each on the topics of variable types, conditionals, logical operators and loops taken from the two programming modules illustrated in Fig. 5a, b, 6a and 6b.

Level 2 was derived from Table 4 where the top rated preferred content for effectiveness for teaching advanced programming concepts were:

Advanced Programming Concepts	Rank	Mean	SD
Inheritance	1st	3.83	0.79
Defining classes, Constructors and Destructors	2nd	3.81	0.82
Setting Member Access Levels, Data Members, Member Functions, Scope	3rd	3.77	0.88
Encapsulation	4th	3.74	0.79

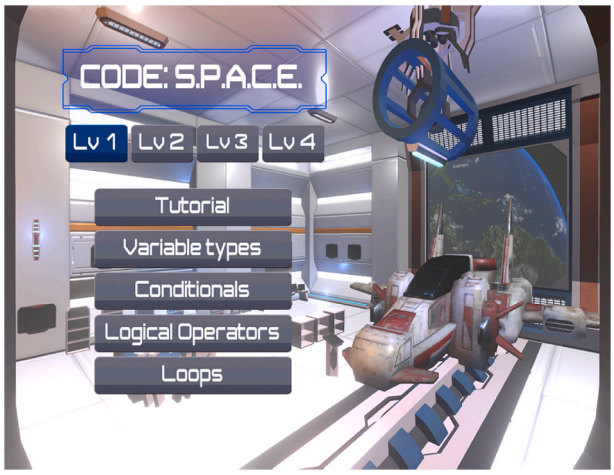

Fig. 5a. Menu screen.

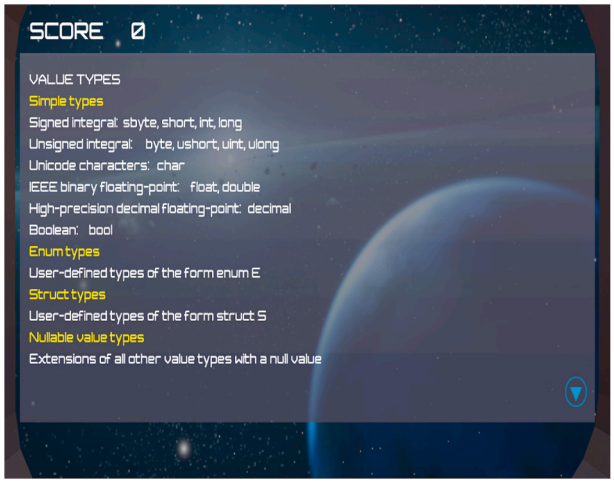

Fig. 5b. Narrative content.

Fig. 6a. Shooting screen.

Therefore, there are ten questions each on classes, constructors, access modifiers and inheritance. The types of questions are illustrated in Fig. 4a and b.

Level 3 was derived from Table 5 where the top rated preferred content and the games potential effectiveness for teaching data structures were:

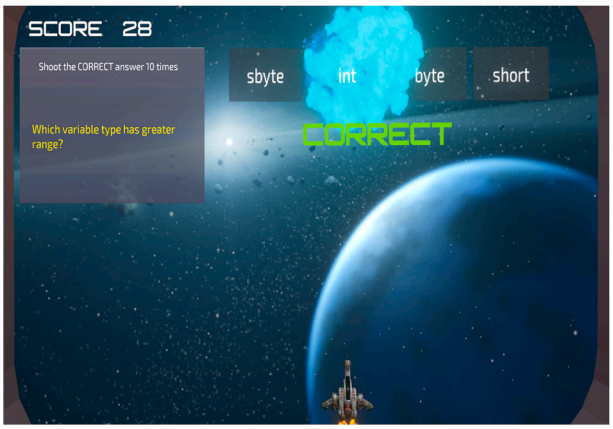

Fig. 6b. Question answering screen.

Data Structure	Rank	Mean	SD
Arrays	1st	3.94	0.77
Queues	1st	3.94	0.84
Stacks	2nd	3.90	0.88
Multi-Dimensional Arrays	3rd	3.88	0.86

There are 10 questions on: arrays, queues, stacks, vectors, and multi-dimensional with the previous format. The tutorial content is shown in Fig. 7a and one of the example questions is shown in Fig. 7b.

Level 4 was derived from Table 6 where the top rated preferred content and the games potential effectiveness for teaching data structures were:

Algorithm	Rank	Mean	SD
Basic Path Finding	1st	3.83	0.79
Bubble Sort	2nd	3.72	0.85
Merge Sort	3rd	3.70	0.88
Heap Sort	4th	3.68	0.89
Quick Sort	4th	3.68	0.89

Level 4 therefore teaches: bubble, heap, quick, merge sort and path finding with various questions on each. Fig. 8a and b illustrate the narrative and the intrinsic, endogenous pedagogical content integrated into the game.

Level 5 is a randomised combination of all of the previous levels.

In summary the game has three types of integrated, intrinsic, endogenous narrative in relation to pedagogical content integration.

- Instructional pedagogical content in the form of tutorials that are revealed in the narrative discourse of an admiral speaking to a pilot.
- Tested pedagogical content in the form of questions that are being asked in the middle of the space shooter game where the player has to shoot the correct answer 10 times to progress.
- Interface pedagogical content in the form of menu choices that are still within the ethos of the game. An illustration of this is in Fig. 9.

The summary results which were placed into a requirements specification in Appendix.

Appendix 3.

6. Evaluation of CODE: S.P.A.C.E Study 2 (Empirical Study Two)

6.1. Procedure and methodology

The methodology utilised for evaluating the game was an experimental post-test intervention design where the post-test was a survey

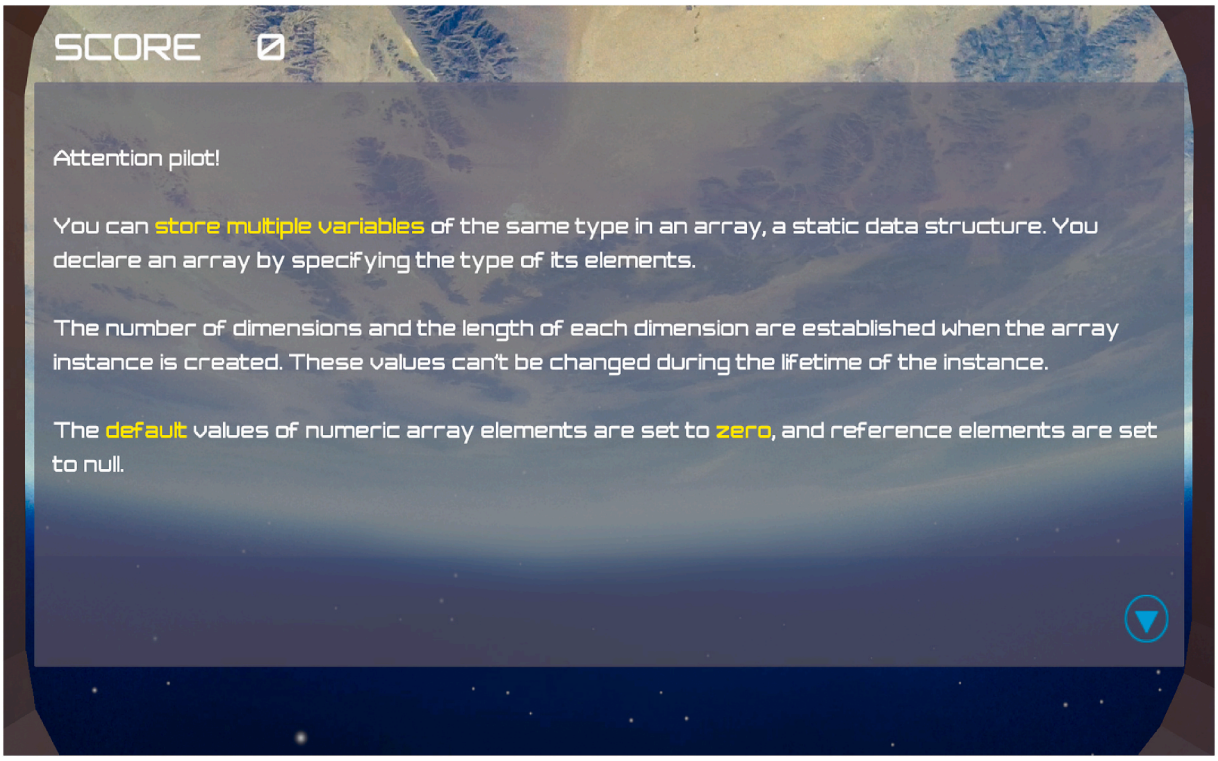

Fig. 7a. Tutorial narrative content.

Fig. 7b. Pedagogical endogenous intrinsic content.

which was based on the survey used for empirical study one. The survey is available at *Appendix 2*. As in empirical study one the data was analysed utilising quantitative non-parametric statistical tests such as Mean, Standard Deviation and Mann-Whitney *U tests*.

The instructions for carrying out the evaluation, CODE: S.P.A.C.E and a link to the evaluation questionnaire were distributed to students at the UWS undertaking a Computer Games Programme. After playing the game participants were invited to complete the *CODE: S.P.A.C.E evaluation questionnaire*.

6.2. Participants

A second year undergraduate programming course called Software Development for Games participated in the evaluation of CODE: S.P.A.C.E. 48 participants completed the *CODE: S.P.A.C.E evaluation questionnaire (Appendix 2)* after data cleansing. 38 participants (81.3%) were male, 6 participants (12.5%) were female and 3 participants (6.3%) preferred not to say. Participants played games for 16.95 h per week (SD = 9.68) with a range of 0–31 h which is consistent with study one. Participants performed 10.95 h (SD = 8.10) of independent

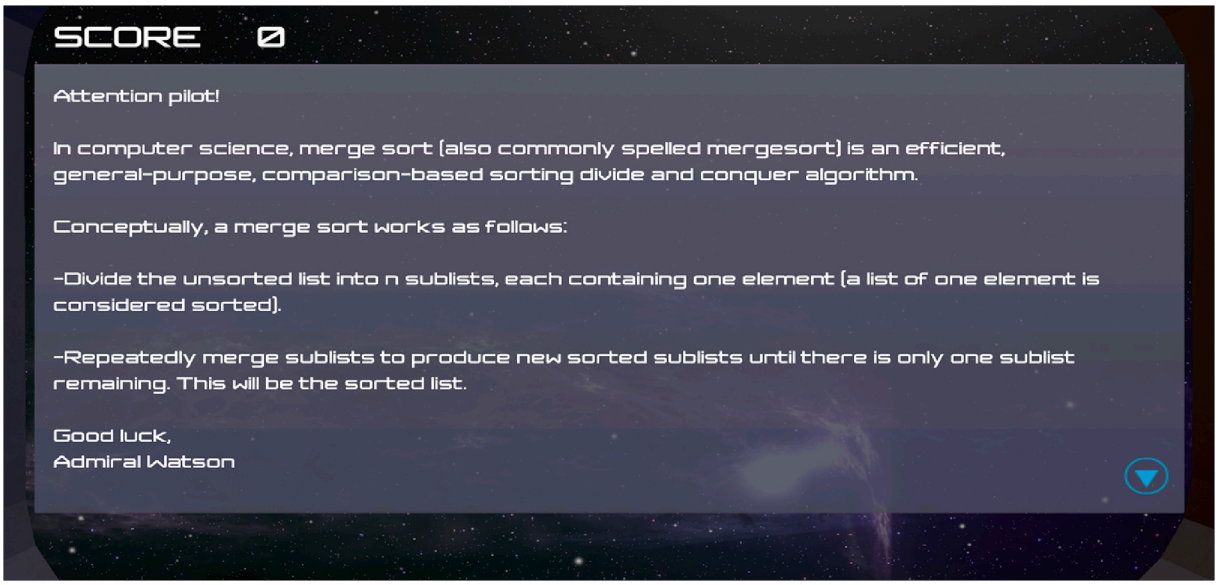

Fig. 8a. Tutorial narrative content.

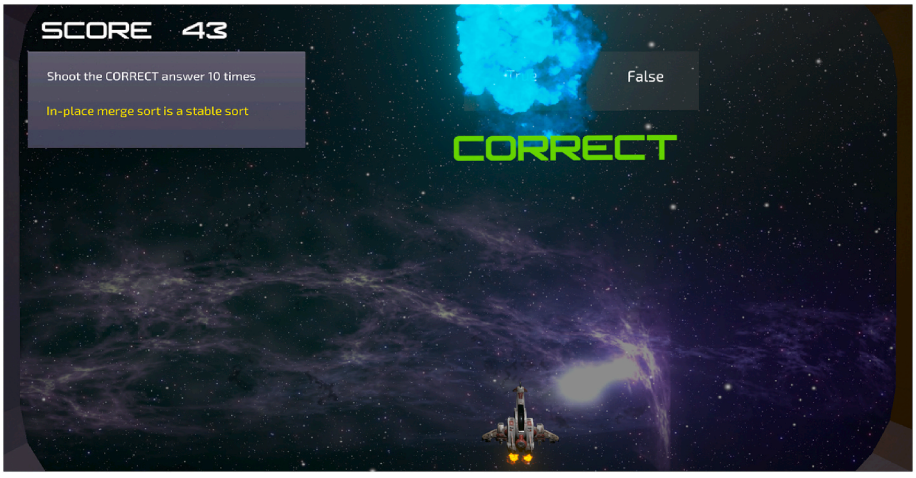

Fig. 8b. Intrinsic, endogenous content.

Fig. 9. Menu interface pedagogical content.

programming study per week with a range of 0–31 h. Again this is relatively consistent with study one. The average age of participants was 23.29 (SD = 4.78) with a range of 18–39. Participants had been learning programming for an average of 4.72 years (SD = 1.97) with a range of 1–10 years. Mann-Whitney *U* tests indicated no significant differences between males and females in relation to age, how long they played computer games per week, how many hours a week they invested in learning to program or the years spent learning to program. 37 participants (77%) were from the Computer Games Development Programme and 11 participants (23%) were from the Computer Games Technology Programme.

Participants were asked to rank the approaches for teaching programming in terms of how motivational they believed the approach to be. The results are displayed in Table 11.

The results indicate that GBL/Serious Games and Computer Games for Learning are potentially a highly motivational teaching approach as they are second and third in the overall ranking after practical lab sessions and individual course works. Participants were asked to rank the approaches for teaching programming in terms of how effective they believed the approach to be. The results are displayed in Table 12.

Computer games for learning were rated as 5th and GBL and Serious Games were rated as 6th in terms of overall effectiveness as teaching approaches. While these approaches were not rated as highly as individual course works, practical lab sessions and tutorials it seems that there is clear trend in the results suggesting that more interactive, practical approaches are rated more highly than passive, less interactive

approaches. 22 participants (45.8%) classified themselves as a novice programmer, 25 participants (52.1%) classified themselves as an intermediate level programmers and only 1 participants (2.1%) classified themselves as an expert level programmer. A Mann-Whitney *U* test indicated that intermediate programmers spent significantly more hours per week (13.36, SD = 8.39) learning to program that novice programmers (7.32, SD = 5.16) ($Z = -2.583$, $p < 0.010$). A Mann-Whitney *U* test also indicated that intermediate level programmers had been learning to program for a significantly longer number of years than (5.46, SD = 1.73) than novice programmers (3.72, SD = 1.75) ($Z = -3.135$, $p < 0.002$).

6.3. Effectiveness of CODE:S.P.A.C.E for teaching rudimentary programming concepts

Participants were asked to rate the effectiveness of the game to teach rudimentary programming concepts. The results are displayed in Table 13.

The results indicated that the game was rated as most effective at teaching the following concepts: variable types, declaring variables, programming syntax, conditionals, loops, switch statements and methods and functions. The game was rated lowest at teaching the following programming concepts: working with constants, smart pointers, and pointers. Mann Whitney *U* tests indicated that there were no significant differences between novice and intermediate level programmers in relation to the game's perceived effectiveness.

6.4. Effectiveness of CODE:S.P.A.C.E for teaching advanced programming concepts

Participants were asked to rate the effectiveness of the game for teaching advanced programming concept and the results are displayed in Table 14.

The game was considered to be most effective at teaching the following: defining classes, constructors and destructors, inheritance and encapsulation.

6.5. Effectiveness of CODE:S.P.A.C.E for teaching data structures

Participants were asked to rate the effectiveness of the game for teaching algorithms and the results are displayed in Table 15. The results indicated that the game was most effective at teaching: arrays, queues, multi-dimensional arrays, stacks and standard vectors.

Mann-Whitney *U* tests again indicated that there was no significant differences associated with perceived effectiveness in relation to novice or intermediate programming level skill.

6.6. Effectiveness of CODE:S.P.A.C.E for teaching algorithms

The participants were asked to rate the games effectiveness for teaching algorithms and the results are displayed in Table 16.

Table 11
Ranking of programming teaching approaches in relation to motivation.

Teaching Approach	Ranking	Mean	SD
Practical Lab Sessions	1st	4.02	0.98
Individual Course Works	1st	4.02	0.81
Games-based Learning/Serious Games	2nd	4.00	0.80
Computer Game for Learning	3rd	3.98	0.86
Classroom Tutorials	4th	3.73	0.98
Online Tutorial	5th	3.52	0.92
Lecturing Classroom	6th	3.31	1.01
Live Thru Case Studies	7th	3.27	0.89
Role Playing	8th	3.26	1.03
Group Based Course Works	9th	3.23	1.06
Paper-based Case Studies	10th	2.71	1.09

Table 12
Ranking of programming teaching approaches in relation to effectiveness.

Teaching Approach	Ranking	Mean	SD
Individual Course Works	1st	4.27	0.76
Practical Lab Sessions	2nd	4.08	0.87
Class Room Tutorials	3rd	3.96	0.82
Online Tutorials	4th	3.92	0.94
Computer Game For Learning	5th	3.79	0.97
Games-based Learning/Serious Games	6th	3.77	0.97
Group Based Course Works	7th	3.63	1.16
Lecturing Classroom	8th	3.55	1.16
Role Playing	9th	3.04	1.02
Live Thru Case Studies	10th	2.98	0.81
Paper Based Case Studies	11th	2.75	0.89

Table 13
Effectiveness of the game for teaching rudimentary programming concepts.

Rudimentary Programming Concept	Ranking	Mean	SD
Variable Types	1st	3.89	1.07
Declaring Variables	2nd	3.81	0.97
Programming Syntax	2nd	3.81	0.91
Conditionals	3rd	3.78	0.95
Loops	3rd	3.78	1.08
Switch Statements	3rd	3.78	0.96
Methods and Functions	4th	3.73	0.99
Objects	5th	3.70	0.97
Arithmetic Operators	5th	3.70	0.97
Logical Operators	5th	3.70	0.94
Statements and Sequences	6th	3.69	0.89
Returning Types with Operators	7th	3.62	0.95
Variable Scope	8th	3.58	1.05
Using Enumerators	9th	3.54	1.02
Compound Conditions	10th	3.51	0.87
Parameter Passing and Overloading Functions	10th	3.51	1.04
Using References	11th	3.46	1.12
Relational Operators	12th	3.44	0.88
Algorithms	13th	3.42	1.00
Working with Constants	14th	3.38	0.92
Smart Pointers	15th	3.35	1.21
Pointers	16th	3.32	1.16

Table 14
Effectiveness of the game to teach advanced programming concepts.

Advanced Programming Concepts	Rank	Mean	SD
Defining Classes Constructors and Destructors	1st	3.78	1.00
Inheritance	2nd	3.74	1.07
Encapsulation	3rd	3.70	0.97
Setting Member Access Levels	4th	3.65	1.01
Polymorphism	5th	3.62	1.06
Static Data Members and Static Data Functions	6th	3.59	1.01
Aggregation/Association	7th	3.51	1.12
Mutator Access Functions	8th	3.44	1.00
Abstract Data Types	9th	3.43	1.09
Static Object Instantiation	10th	3.42	1.13
Composition	11th	3.41	1.12

The results indicated that the game was most effective at teaching the bubble, quick and merge sort. Mann Whitney *U* tests again indicated that there was no significant difference between novice and intermediate level programmers in relation to the games effectiveness in teaching algorithms.

6.7. Effectiveness of CODE:S.P.A.C.E for teaching particular skills

The participants were asked to rate the effectiveness of the game for teaching particular skills that were important for learning programming. The results are displayed in Table 17 showing that recollection, problem solving and analysing and classifying are the top ranked skills.

Table 15
Effectiveness of the game to teach data structures.

Data Structure	Rank	Mean	SD
Arrays	1st	3.92	0.98
Queues	2nd	3.78	0.98
Multi-Dimensional Arrays	2nd	3.78	0.95
Stacks	3rd	3.72	1.00
Standard Vector	4th	3.69	0.95
Priority Queues And Heaps	5th	3.51	1.10
Binary Search Trees	5th	3.51	1.12
Trees	6th	3.50	1.08
Binary Trees	7th	3.49	1.12
Game Trees And Minimax Trees	8th	3.43	1.19
Graphs	9th	3.42	1.25
Uni-Directional Graphs	9th	3.32	1.23
Bi-Directional Graphs	10th	3.30	1.29
Hash Tables	10th	3.30	1.20
Bit Vectors	11th	3.28	1.28

6.8. Technical aspects of the game

The participants were asked to rate the importance of the technical aspects of the game. The results are presented in Table 18. Clear goal structure and control mechanism and interface as well as navigation inside the environment and help and scaffolding are considered important in the environment. This is consistent with the requirements that were formulated in empirical study one.

The clear goal structure, help and scaffolding refer to the highly detailed instructions, narrative, and detailed tutorial level to set the scene within the environment. Fig. 10 shows an example of the tutorial level and narrative to assist the player in understanding the goal structure and provides clear direction within the environment. The game has an exceptionally smooth interface for navigation between the gaming and puzzle elements i.e. the enemy ships and the questions. The game interface provides an intrinsic menu structure where the narrative always situates the player in the role of 'pilot' of a ship. Intrinsic pedagogical menu structures can be viewed in Fig. 9.

6.9. Formative assessment effectiveness, type and format suitability

Participants were asked how effective the game was as a preparation mechanism for their class test assessment. The results were generally positive. 37 participants answered this question were 5 participants (14%) rated the game as very effective, 19 participants (51%) rated the game as effective, 9 participants (24%) rated it as neutral, 3 participants (8%) rated it as ineffective and 1 participant (3%) rated the game as very ineffective. Participants were also asked how they believed that a computer game could be utilised within the context of a module at HE level in either a formative or summative capacity or both. 37 participants answered this question where 25 (68%) believed it could be used in both a formative and summative way, 6 (16%) believed that it could only be used in a formative capacity and 6 (16%) in a summative way. 20 participants (54%) believed that a game could in fact be an assessment for a module where 17 participants (46%) believed that it could not be used as an assessment for a module.

Table 16
Effectiveness of game to teach algorithms.

Algorithm	Rank	Mean	SD
Bubble Sort	1st	3.81	0.97
Quick Sort	2nd	3.78	1.03
Merge Sort	3rd	3.70	0.94
Binary Search	4th	3.68	0.97
Heap Sort	5th	3.62	1.01
Basic Path Finding	6th	3.59	1.17
Robust Path Finding	7th	3.51	1.12
Radix Sort	7th	3.51	1.22

Table 17
Effectiveness of the game to teach skills.

Skills	Rank	Mean	SD
Recollection	1st	3.78	1.22
Problem Solving	2nd	3.62	1.19
Analysing/Classifying	3rd	3.57	1.09
Reflection	4th	3.49	1.19
Critical Thinking	5th	3.46	1.17
Mathematics	6th	3.42	1.23
Management	7th	3.05	1.33
Creativity	8th	3.03	1.21
Leading/Motivating	9th	2.58	1.34
Collaboration/Teamwork	10th	2.43	1.34

Table 18
Ratings of technical aspects of the game.

Technical aspect	Rank	Mean	SD
Clear Goal Structure	1st	4.46	0.56
Control Mechanism and Interface	2nd	4.08	0.92
Navigation Inside the Environment	3rd	4.05	0.88
Help and Scaffolding	4th	3.81	1.17
Sound	5th	3.38	1.11
Narrative and Dialog	6th	3.30	1.33
Collaboration	7th	2.97	0.96
Graphics	8th	2.76	1.14
Realism of the Characters in the Environment	10th	2.57	1.26
Realism of the Environment	11th	2.51	1.17
Character Customisation	12th	2.20	1.23

The game was produced utilising the Unity 5 Games Engine and as a result could really be delivered on a number of platforms. Participants were asked to rate the suitability of the game for different platforms of delivery. The results are displayed in Table 19.

The results indicated that the game was most suitable for the PC, Tablet PC or Mobile Phone. The game was integrated as a formative assessment revision supplementary tool into a specific module at HE which has now run for a period of two years. In the first year of running the students in the class did not have access to this game, however in the second year of running the module they did. A Mann-Whitney *U* test was performed between the two independent year groups to ascertain if the game made a statistically significant difference to the overall grades in the module and the grades in the class test worth 40%. The results of the test indicated that there was no significant differences in overall grades ($Z = -1.073$, $p < 0.283$) or the class test ($Z = -0.743$, $p < 0.453$). This is possibly due to the small cohorts in each level and the fact that the experimental design could have been stronger in relation to evaluation. The best way to evaluate such a game would require a Randomised Controlled Trial (RCT) with 30 representative samples in each group to produce a statistically significant comparison.

7. Conclusions

This section will revisit the original research questions and provide a summarised answer for them.

Research Question 1: What are the main problems associated with traditional programming teaching approaches?

Many problems were found associated with computer programming education. Generally speaking programming is perceived as difficult, not particularly motivational and with high levels of attrition. Traditional didactic approaches including lectures, paper-based based studies, live-through case studies and role playing are not seen as effective. One of the findings of the questionnaire study was that practical lab exercises and course work are the most effective and motivational teaching approach to computer programming, however the

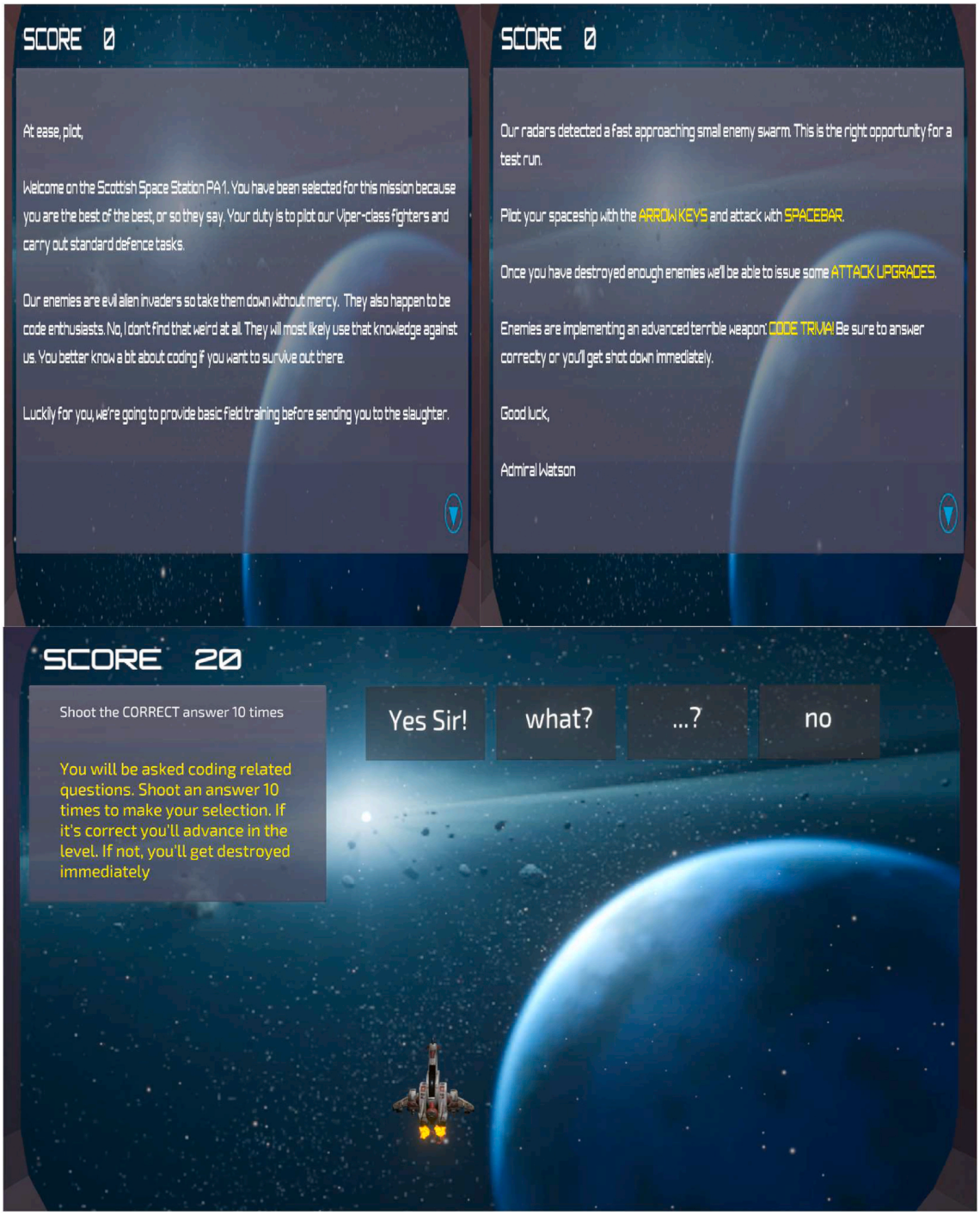

Fig. 10. Tutorial level and narrative within the game.

Table 19
Platform suitability rating.

Format	Rank	Mean	SD
Personal Computer	1st	4.81	0.40
Tablet PC	2nd	4.11	0.81
Mobile Phone	3rd	3.73	1.07
Games Console	4th	3.61	1.08
Augmented Reality	5th	2.72	1.11
Virtual Reality	6th	2.53	1.18

literature indicates that one problem associated with current approaches is lack of practical, industry-based experience representing a holistic view of the software development or programming process. New more interactive approaches such as games, GBL, serious games and even the gamification of courses have to be more frequently utilised to act as a supplementary learning approach for traditional approaches to address this situation.

Tan et al. (2009) performed a similar questionnaire study asking 182 undertaking an undergraduate course on computer programming about their level of understanding of certain rudimentary and advanced concepts. The study found that the following concepts were easiest to understand: variables, decision logic structures, iteration logic structures,

error handling, arrays and parameters. The hardest concepts to understand were: object oriented principles (polymorphism, inheritance encapsulation), pointers, abstract data types and recursion which is highly consistent with the findings of this particular study. This verifies that the results obtained in this study were generalizable and by backing up the findings of this study has proven that these results have wide reaching applications in Scottish HE overall. It is also apparent that the study performed by Tan et al. (2009) was performed almost a decade previously indicating that ten years on these issues have not been sufficiently addressed or resolved.

Research Question 2: How have computer games been used for the purposes of teaching computer programming in the past?

314 papers were included in the final literature review. 187 of these were papers with developed or existing games revealing that computer games have certainly been used for teaching computer programming in the past. These games are at various different levels of abstraction, utilising different platforms, implementation languages, levels of graphical fidelity and to teach a number of different topics including pointers, search algorithms, logic, syntax, object orientation and data structures, functions/methods and debugging. The IEEE search engine revealed the majority of these games which is not particularly surprising with an excess of 60 games being discovered. In general, the study has indicated that computer games are a highly versatile medium for teaching computer programming. A full analysis of these games and the other identified studies from Fig. 1 are beyond the scope of this study but can be a future research direction.

Research Question 3: What issues are identified in terms of pedagogical content integration for computer games teaching programming? What existing frameworks (if any) are there?

A number of issues were identified in terms of considering pedagogical content integration for a game to teach computer programming. The issues identified in the literature were some of the following: what particular genre, format and type of game, whether the game should be single or multiplayer, assessment integration and feedback, adherence to the computer science curriculum, whether the game is to address high level, low level programming languages or to be programming language agnostic, how are abstract concepts to be incorporated into the game and general acceptance of the game. While there are frameworks in the literature that address content integration such as the content segmentation framework (Sauvé, 2010), the classroom design framework (Echeverria et al., 2011) and the guidelines for embedding pedagogical content into game-play (Tang & Hanneghan, 2010), these frameworks are too general and not specific enough for considering the issues of incorporating specialised content into a game for teaching programming. This resulted in the development of a new preliminary framework for considering content integration for a game to teach computer programming shown in Figure 2. This framework has been crucial in the development of the prototype game as a result and is a preliminary framework which can undergo further future refinement.

Research Question 4: To *what extent will a games-based learning approach be accepted by learners for teaching computer programming?*

75% of participants believed that computer games could be used to teach computer programming concepts at HE level in Study One. Ibrahim, Yusoff, Omar, and Jaafar (2011) found a similar positive result where 60% of participants believed that they could learn better with educational games being an option in the curriculum and 80% of participants believing that games make the subject more interesting and assist them to think critically which coincidentally is the 3rd most important skill identified by participants to be a successful programmer in this study. In conclusion games are generally accepted by learners but it would appear that as with traditional approaches they are not entirely preferred by everyone. This study has expanded on the study performed by Ibrahim et al. (2011) by utilising the literature and the questionnaire to produce a framework for a programming serious game in terms of content integration. This framework has also been utilised to produce an implemented game to act as a formative assessment teaching and scaffolding tool on two programming course at HE level. The results of integration of this game into a module have been positive and this study can be expanded to again verify that the results have wider applications in Scottish HE and potentially internationally.

Research Question 5: What programming pedagogical content would participants like to see in a game to teach programming at HE level?

Participants indicated that the more simpler programming concepts were suitable to be taught by a game which is possibly due to the fact that more simplistic concepts are easier to visualise in a gaming scenario. Table 20 shows the most popular rudimentary and advanced programming concepts as well as data structures and algorithms to be incorporated into a computer game.

Research Question 6: Can a computer game be effective as a formative assessment tool for computer programming in HE?

The results indicate that GBL/Serious Games and Computer Games for Learning are potentially a highly motivational teaching approach with computer games for learning rated as 5th and GBL and Serious Games were rated as 6th overall. The results indicated that that more interactive, practical approaches are preferable.

In relation to rudimentary programming concepts, the game was rated more effective at teaching: variable types, declaring variables, programming syntax, conditionals, loops, switch statements and methods and functions and objects which was consistent with the preferred content materials from study one. The game was rated lowest at teaching the following programming concepts: working with constants, smart pointers and pointers.

In relation to advanced programming concepts the game was rated highly effective in being able to teach the following concepts: defining classes, constructors and destructors, inheritance and encapsulation which is again consistent with the preferred content identified by potential participants in study one. The game was rated highly in terms of being able to teach the following data structures effectively: arrays, queues, multi-dimensional arrays, stacks and queues which again is relatively consistent in relation to the content that was identified for incorporation. A similar finding was found in relation to algorithms with the exception of the basic path finding algorithm possibly indicating that the game produced does not intuitively lend itself to this particular concept.

No significant differences between novice and intermediate level programmers in relation to the game's effectiveness with regards to rudimentary programming concepts, advanced concepts, data structures or algorithms was detected. This is possibly due to the generic game mechanics allowing practically any pedagogical content to be integrated

Table 20
Preferred Content to be Integrated.

Rudimentary	Advanced	Data structures	Algorithms
• Conditionals •Logical operators •Variable types •Loops •Objects	•Inheritance •Defining classes, constructors and destructors •Setting member access levels, data members, member functions and scope	•Multi-dimensional arrays •Standard vectors •Stacks •Arrays	•Basic path finding •Bubble sort •Quick sort •Heap sort •Merge sort

into it. The game has a tutorial narrative and shooter quiz with 4 possible answers and one correct answer structure. This may mean that optimistically it can be effective at teaching novice and intermediate level programmers. Pessimistically this could be a further research study direction to identify novice and intermediate level content and perform separate evaluations with a control structure to ascertain the games perceived effectiveness at teaching different programming concepts/levels.

The results indicated that the game was particularly effective at developing the following important skills for learning to program: recollection, problem solving, analysing/classifying, reflection and critical thinking. The main difference from the first empirical study was that recollection was rated as 8th most important, however problem solving and critical thinking ranked highly in both study one and study two. In terms of the technical aspects of the game it is evidence in both study one and two that a clear goal structure, control mechanism and the interface, navigation inside the environment and help and scaffolding were all rated highly indicating consistency.

14% rated the game as very effective and 51% rated the game as effective for their class test preparation. 68% believed it could be used in both a formative and summative way, 54% believed that a game could in fact be an assessment for a module and 46% believed that it could not be used as an assessment for a module. A comparison of class test grades and overall module grades between students who had access to the game and students who did not indicated that there was no significant differences in overall grades or the class test. The overall results suggest that a computer game for teaching programming concepts can be used effectively as a formative and potentially summative assessment tool that is preferable to a majority of students in a preliminary evaluation. There are a number of implications to be considered for the game to be used in as a formative or summative assessment tool. Firstly, whether the game should be used in a formal or informal capacity i.e. whether the results of the game are actually assessed and contribute towards a grade in a formal capacity or whether it is used as a formative supplementary revision tool before in an informal capacity before formal assessment takes place. Secondly it is necessary to consider whether a game could in fact be suitable for every type of learner within a class and it is important to consider learner preferences and more importantly accessibility in relation to equality to provide equal opportunities to succeed.

7.1. Limitations

One primary limitation of games is that they will never replace didactic traditional teaching approaches and it is highly important to be realistic about this for the moment. Computer games have a number of negative connotations associated with them including that they are conducive to negative personality traits and behaviours (Rosas et al., 2003). Similarly, to every other teaching approach computer games are subject to individual preferences, perceptions, motivations and attitudes. If for example an educationalist is attempting to teach mathematical concepts with a computer game and the learner has a negative attitude to both mathematics and computer games – then this approach will significantly reduce in effectiveness.

Another limitation of computer games associated with attitudes is that games are primarily viewed as fun entertaining interactive pass times where a highly challenging aspect is separating the fun from the educational engagement. This limitation was implicitly implied from the literature review with the idea of intrinsic, extrinsic and exogenous and endogenous content integration.

These results were presented to a number of educationalists in the field of serious games and programming at a workshop on Serious Games in Cyber Security and it was discovered that there is a disparity between the learner and an educationalist perspective in relation to how a prototype game should be developed. Learners would prefer to have the most simplistic concepts incorporated into a prototype game where educationalists believe that it would be more advantageous to incorporate the more complex and difficult subjects to be educationally far more useful rather than easier. This indicates that the design is entirely subjective and should take into account programming educationalists in the future.

7.2. Implications and future research directions

There are a number of implications and future recommendations for the future including:

Greater appreciation and utilisation of the potential of the use of computer games, GBL, serious games, gamification, Games-Based Construction Learning in computer science, computer games development and technology courses at a number of education levels. This can be achieved in three ways.

- Educationalists themselves developing games by the use of their own games development skills, grants or subject matter experts. With the increasing simplicity of games development engines such as Unity, Unreal and Game Maker this will become more and more feasible without having to possess great games programming knowledge.
- Utilisation of students on these courses at a more advanced level reflecting on particular aspects of programming education that they found difficult and setting them assessments to develop game applications to assist students at a lower level.
- There are a vast number of games for teaching a number of different areas and a particular concentration of computer games to teach aspects of programming. A taxonomy and catalogue of serious games in general and for programming education should be created to make use of already existing games.

Levels of education, experience, knowledge and ability have to be more deeply considered in good Higher Educational Practice in relation to computer programming courses and programmes. This is especially an issue when encountering issues such as direct entry and articulation from FE (Further Education) institutions to HE institutions or simply coming from SE (Secondary Education) institutions as the learning experience significantly differs in relation to programming Mitamura et al., 2012 and is maybe deficient at earlier levels of education (De Aquino Leal & Ferreira, 2013). Traditional approaches can be seen as lacking motivation, engagement and hence become ineffective (Berkling & Thomas, 2013) with issues of generational gaps between teachers and students (Khenissi et al., 2013). These issues raise a number of recommendations for policy in computer programming courses in relation to formative and summative assessment. Educators have to enhance their responsibility in relation to formative assessment to suitably gauge level of experience, knowledge and skills. Computer games can assist with these issues in relation to being a formative assessment mechanism which is potentially more engaging and ideally reduces any learner embarrassment as they can be played on an individual level at home to highlight problems and bring learners up to the same base level. Lecturers in HE have to perform more research into formative assessment methods which are inclusive and involve learners in their development to a far greater degree. The Scottish Qualifications Authority (2018, p. 11) in their future of assessment report compiled by young adults in Scotland had the following aspiration:

"*Assessment should be fair, and inclusive a "level playing field" for all, but to achieve this assessment needs to be personalised, with a broad variety of assessment methods that young people can choose from over the course of a school year. We need to strive for equity, not equality.*"

With regards to giving young adults the choice of assessment methods in education – it may very well be the case that one of the novel teaching approaches that is selected may be computer games.

The following future research directions have been compiled from the synthesis of this study:

- This paper has presented a preliminary evaluation of the game assimilated into a HE module however a more detailed and stringent experimental study could be performed in other modules and runs of the same module to ascertain validity and utilise a Randomised Controlled Trial (RCT) to gain deeper comparative insights of games and traditional teaching approaches.
- Due to the size and complexity of this study it was not possible to fully detail the analysis of all of the studies found in the systematic literature review as it was beyond the scope of the study. Therefore, further detailed analysis of the systematic literature review findings will be performed for the following reasons:
  - o to fully detail the developed and existing games where 187 papers were found. It should be noted that there are duplicate games in these papers as there can be several papers published associated with one game for teaching programming.
  - o to fully detail the problems associated with programming where 91 papers were found.
- A taxonomy of existing serious games will be developed for the use of educationalists and serious game researchers particularly in the area of computer programming. This would take some time to produce but the idea would be that educationalists would be able to augment their material with a number of supplementary serious games to assist learners.
- Further research is required in the area of formative and summative assessment mechanisms to ascertain how learners in computer programming courses would like to be assessed and also an additional framework could be developed in relation to formative assessment to gauge levels of experience, knowledge, and skill. This is particularly relevant in relation to articulation and direct entry into university courses.
- No significant differences were found between the game and non-game groups in relation to class tests and overall grades and this warrants further rigorous investigation. A more rigorous RCT should be conducted with a suitable representative sample in each group to allow parametric statistical tests to test for statistical significance.
- A disparity was discovered between educationalists and learners and future research could collect the perceptions, attitudes, and motivation of programming educationalists to obtain a different perspective and increase understanding.
- While this study has been primarily quantitative, additional qualitative studies could be performed to obtain deeper analysis to adhere to pedagogical, evaluation and assessment standards.
- From a design and development perspective of the current prototype game there are four particular future research avenues that can be explored:
  - o Firstly, validation of the game in other levels of education and in other institutions to gather more conclusive empirical data to support the validity of the approach. So for example allowing the game to be used in Further Education and Secondary Education and possibly other Higher Education institutions.
  - o Secondly, the game could be translated into other programming languages as it is currently customised for teaching C#. Despite the fact that programming principles are transferrable it may be more effective to convert the game to the programming languages that the students are learning in their course such as C++, Java, Python etc.
  - o Thirdly, the game could be expanded to be generic in relation to allowing the educationalists to add their own pedagogical content. This may mean the development of an editor tool to allow teachers to add questions for their students similar to the Dialog Editor that was created for the Requirements Collection and Analysis Game (Hainey, 2010).
  - o Fourthly, the game could have challenge and difficulty customisation implemented into it to allow for adjustment to programming knowledge e.g. specifying a level of easy, medium or hard or even specifying level of programming ability as novice, intermediary or expert.
- Finally, the content integration framework presented in this study was a preliminary framework which could undergo refined future development.

CRediT authorship contribution statement

Thomas Hainey: Conceptualization, Formal analysis, Funding acquisition, Investigation, Methodology, Project administration, Resources, Software, Supervision, Validation, Visualization, Writing – original draft. **Gavin Baxter:** Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work is funded by the Carnegie Trust – Grant Number: RIG007432.

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.cexr.2024.100061.

References

ADPRIMA Instructional methods information website.(2019). Retrieved 6 August, 2019, from https://www.adprima.com/teachmeth.htm.

Aldrich, C. (2004). *Simulations and the future of learning: An innovative (and perhaps Revolutionary) approach to e-learning*. San Francisco: Pfeiffer.

Aldrich, C. (2005). Learning by doing: A comprehensive guide to simulations computer games, and pedagogy in e-learning and other educational experiences. *Pfeiffer*.

Alhazbi, S., & Ismail, L. S. (2010). Supportive online learning environment to improve student' satisfaction in object-oriented programming courses. In *In Proceedings of the 2nd international Congress on engineering education, december 8 – 9, Kuala Lumpar, Malaysia*.

Anderson, E. F., & McLoughlin, L. (2007). Critters in the classroom: A 3D computer game-like tool for teaching programming to computer animation students. In *In Proceedings of SIGGRAPH '07 San Diego, California, August 05 -09*.

Andrianoff, S., & Levine, D. (2002). Role playing in an object-oriented world. In *In Proceedings of the 33rd SIGCSE technical Symposium on computer science education, Kentucky, USA* (pp. 121–125). ACM Press.

Arsenault, D. (2009). Video game genre, evolution and innovation. *Eludamos Journal of Computer Game Culture, 3*(Issue. 2), 149–176.

Backland, P., & Hendrix, M. (2013). Educational games – are they worth the effort? A literature survey of effectiveness of serious games. In *Proceedings of 5th international Conference on Games and virtual Worlds for serious applications (VS - GAMES), 11-13 Sep 2013, Poole UK*.

Baker, A., Zhang, J., & Caldwell, E. R. (2012). Reinforcing array and loop concepts through a game-like module. In *In proceedings of the 2012 17th international conference on computer games (CGAMES), july 30 – Aug 1, Louisville, KY, USA*.

Berkling, K., & Thomas, C. (2013). Gamification of a Software Engineering Course and a detailed analysis of the factors that lead to its failure. In *Proceedings of the 2013 international Conference on interactive collaborative learning (ICL), September 25 – 27, Kazan, Russia*.

Bernstein, L., & Klappholz, D. (2001). Getting software engineering into our guts. CrossTalk. *The Journal of Defense Software Engineering*, 25–26.

Biggs, J. (1999). *Teaching for quality learning at university*. Buckingham: Open University Press.

Blair, J., Czaja, R. F., & Blair, E. A. (2013). *Designing surveys: A guide to decisions and procedures*. SAGE Publications, Inc.

Bonwell, C. C. (1996). Enhancing the lecture: Revitalizing a traditional format. In T. E. Sutherland, & C. C. Bonwell (Eds.), *In Using active learning in college classes: A range of options for faculty, new Directions for Teaching and learning No. 67* (pp. 31–44). Place: Wiley Periodicals Inc.

Boticki, I., Barisic, A., Martin, S., & Drljevic. (2012). Sortko: Learning sorting algorithms with mobile devices. In *In proceedings of the seventh IEEE international conference on wireless, mobile and ubiquitous technology in education. March 27-30, Takamatsu, Japan*.

Boyle, E. A., Hainey, T., Connolly, T. M., Gray, G., Earp, J., Ott, M., et al. (2016). An update to the systematic literature review of empirical evidence of the impacts and

outcomes of computer games and serious games. *Computers & Education, 94*, 178–192.
Browne, K., & Anand, C. (2013). Gamification and serious game approaches for introductory computer science tablet software. In *Gamification'13 proceedings of the first international conference on gameful design, research and applications, Toronto, Ontario, Canada* (pp. 50–57).
Cagin, K. (2013). Empirical evidence that proves a serious game is an educationally effective tool for learning computer programming constructs at the computational thinking level. In *PhD thesis, university of Greenwich*.
Cashin, W. E. (1985). *Improving lectures idea paper No. 14. Manhattan*. Kansas State University, Center for Faculty Evaluation and Development.
Chaffin, A., Doran, K., Hicks, D., & Barnes, T. (2009). Experimental evaluation of teaching recursion in a video game. In *In Sandbox '09 proceedings of the 2009 ACM SIGGRAPH symposium on video games* (pp. 79–86). New Orleans, Louisiana.
Chang, M.&K. (2010). Web-based multiplayer online role playing game (MORPG) for assessing students' Java programming knowledge and skills. In *In proceedings of the third IEEE international conference on digital game and intelligent toy enhanced learning (DIGITEL)*.
Chang, W.-C., Chou, Y.-M., & Chen, K.-C. (2010). Game-based digital learning system assists and motivates C programming language learners. In *In proceedings of the 6th international conference on networked computing and advanced information management* (pp. 10–16).
Connolly, T. M., Boyle, E. A., MacArthur, E., Hainey, T., & Boyle, J. M. (2012). A systematic literature review of empirical evidence on computer games and serious games. *Computers & Education, 59*(2), 661–686.
Connolly, T. M., Stansfield, M. H., & Hainey, T. (2007). An application of games-based learning within software engineering. *British Journal of Educational Technology, 38* (Number 3), 416–428.
Connolly, T., Stansfield, M., & Hainey, T. (2009). Towards the development of a games-based learning evaluation framework. In T. Connolly, M. Stansfield, & L. Boyle (Eds.), *Games-based learning advancements for multi-sensory human computer interfaces: Techniques and effective practices* (pp. 251–273). Hershey, PA, USA: IGI Global.
Connolly, T. M., Stansfield, M. H., McLellan, E., Ramsay, J., & Sutherland, J. (2004). Applying computer games concepts to teaching database analysis and design. In *In proceedings of the international conference on computer games, AI, design and education, reading, UK, november 2004*.
Conole, G., & Warburton, B. (2005). A review of computer-assisted assessment. *ALT-J, Research in Learning Technology, 13*(1), 17–31.
Cope, C., & Horan, P. (1996). The role played case: An experiential approach to teaching introductory information systems development. *Journal of Information Systems Education, 8*(2–3), 33–39.
Davis, B. G. (2001). *Tools for teaching*. San Francisco, CA: Jossey-Bass Publishers.
Dawson, M. (2014). Beginning C++ through game programming. In *4th edition, Boston, USA, CANGAGE learning*.
De Aquino Leal, A. V., & Ferreira, J. (2013). Teaching computer programming based on pattern with activities and collaborative games using concrete materials for high school students. *Proceedings of the IEEE Frontiers in Education Conference (FIE), 1*, 1604–1610.
Deitz, S., & Buy, U. (2016). From video games to debugging code. In *In proceedings of IEEE/ACM 5th international workshop on games and software engineering (GAS), Austin, TX, USA, may 16 2016*.
Doss, K., Juarez, V., Vincent, D., Doerschuk, P., & Liu, J. (2011). Work in progress – a survey of popular game creation platforms used for computing education. In *In proceedings of 2011 frontiers in education conference (FIE), Oct 12 – 15, Rapid city, SD, USA*.
Echeverria, A., Garcia-Campe, C., Nussbaum, M., Gil, F., Villalta, M., Amestica, M., et al. (2011). A framework for the design and integration of collaborative classroom games. *Computers and Education, 57*(1), 1127–1136.
Finkelstein, S. L., Nickel, A., Harrison, L., Suma, E. A., & Barnes, T. (2009). cMotion; A new game design to teach emotion recognition and programming logic to children using virtual humans. In *In proceedings of IEEE virtual reality conference, March 14 – 18, Lafayette, LA, USA*.
Gerling, K. M., Birk, M., Mandryk, R. L., & Doucette, A. (2013). The effects of graphical fidelity on player experience. In *In Proceedings of the 13th international Conference on making Sense of converging media* (p. 229).
Gomez-Martin, M. A., Gomez-Martin, P. P., & Gonzales-Calero, P. A (2009). Content Integration in Games-Based Learning Systems. In T. M. Connolly, & M. H. Stansfield (Eds.), *Games-based Learning Advancement for Multisensory Human Computer Interfaces: Techniques and Effective Practices*. Hershey: Idea-Group Publishing.
Hainey, T. (2010). Using games-based learning to teach requirements collection and analysis at tertiary education level. *Doctoral Dissertation*. Retrieved April 2019 from http://cis.uws.ac.uk/thomas.hainey/Final%20PhD%20Thesis%20Tom%20Hainey.pdf.
Hainey, T. (2016). Moving digital games for learning forward. *On the Horizon, 24*(1), 132–136.
Hainey, T., Connolly, T. M., Boyle, E. A., & Wilson, A.&R. (2016). A systematic literature review of games-based learning empirical evidence in primary education. *Computers & Education, 102*, 202–223.
Hainey, T., Connolly, T. M., Stansfield, M. H., & Boyle, E. A. (2011a). Evaluation of a game to teach requirements collection and analysis in software engineering at tertiary education level. *Computers & Education, 56*(1), 21–35.
Hainey, T., Connolly, T. M., Stansfield, M. H., & Boyle, E. A. (2011b). The differences in motivations of online game players and offline game players: A combined analysis of three studies at higher education level. *Computers & Education, 57*(4), 2197–2211.
Hays, R. T. (2005). *The effectiveness of instructional games: A literature review and discussion*. DTIC Document.
Hidalgo-Céspedes, J., Marín-Raventós, G., & Lara-Villagrán, V. (2014). Playing with metaphors: A methodology to design video games for learning abstract programming concepts. *ITICSE, 14*. Jun 21-25 2014, Uppsala, Sweden.
Hwang, C.-S., Su, Y.-C., & Tseng, K.-C. (2010). Effects of computer game-based instruction on students' programming achievement in Taiwan. In *In proceedings of the international conference on computational aspects of social networks* (pp. 233–236).
Ibrahim, R., Yusoff, R. C. M., Omar, H. M., & Jaafar, A. (2011). Students perceptions of using educational games to learn introductory programming. *Computer and Information Science, 4*(Issue No 1).
Jantan, S. R., & Aljunid, S. A. (2012). An experimental evaluation of scaffolded educational games design for programming. In *In proceedings of the 2012 IEEE conference on open systems* (pp. 21–24). Malaysia: Kuala Lumpur.
Jayasinghe, U., & Dharmaratne, A. (2013). Game based learning vs. Gamification from higher education students' perspectives. In *In proceedings of the 2013 IEEE international conference on teaching, assessment and learning for engineering (TALE), August 26 – 29, Bali, Indonesia*.
Jiau, H. C., Chen, J. C., & Ssu, K.-F. (2009). Enhancing self-motivation in learning programming using game-based simulation and metrics. *IEEE Transactions on Education, 52*(4).
Jordaan, D. B. (2018). Board games in the computer science class to improve students' knowledge of the Python programming language. In *In proceedings of the international conference on intelligent and innovative computing applications (ICONIC), december 6 – 7*. Maritus: Plain Magnien.
Karapinar, Z., Zavarak, S., Senturk, A., Kara, R., & Erdogmus, P. (2012). A game to test pointers: Path finding. In *In proceedings of the 2012 international conference on information technology based higher education and training (ITHET)* (pp. 21–23).
Kaučič, B., & Asič, T. (2011). Improving introductory programming with scratch?. In *In Proceedings of the 34th international convention MIPRO, may 23 – 27, Opatija, Croatia*.
Kawash, J., & Collier, R. (2013). Using video game development to engage undergraduate students of assembly language programming. In *In SIGITE'13 Proceedings of the 14th annual ACM SIGITE conference on Information technology education* (pp. 71–76).
Ketamo, H., Kiili, K., Arnab, S., & Dunwell, I. (2013). Integrating games into the classroom: Towards new teachership. In S. de Freitas, M. Ott, M. M. Popescu, & I. Stanescu (Eds.), *New pedagogical approaches in game enhanced learning: Curriculum integration* (pp. 534–537). IGI Global. https://doi.org/10.4018/978-1-4666-3950-8.ch007.
Khenissi, M. A., Essalmi, F., & Jemni, M. (2013). Presentation of a learning game for programming languages education. In *In Proceedings of the 13th International conference on advanced learning technologies*. July 15 – 18.
Kumar, B., & Sharma, K. (2018). A gamified approach to achieve excellence in programming. In *In Proceedings of the 4th international Conference on computing sciences (ICCS), August,* (pp. 30–31).
Law, R. (2012). Teaching programming using computer games: A program language agnostic approach. In *In Proceedings of the 6th European Conference of games-based learning (ECGBL), Cork, Ireland* (pp. 4–5).
Lawrence, R. (2004). Teaching data structures using competitive games. *IEEE Transactions on Education, 47*(4), 459–466.
Lee, M. J., Bahmani, F., Kwan, I., LaFerte, J., Charters, P., Horvath, A., et al. (2014). Principles of a debugging-first puzzle game for computing education. In *In proceedings of the IEEE symposium on visual languages and human-centric computing (VL/HCC), july 28 August 1, Melbourne, VIC, Australia*.
Lotfi, E., & Mohammed, B. (2018). Teaching object oriented programming concepts through a mobile serious game. In *In SCA'19 proceedings of the 3rd international conference on smart city applications, Tetouan, Morocco, October* (pp. 10–11).
Macphail, A., Hainey, T., & Connolly, T. M. (2012). Applying mLearning in software engineering education: A survey of mobile usage. In *In proceedings of IADIS international conference mobile learning* (pp. 107–114).
Malik, D. (2014). *C++ programming: Program design including data structures, 7 edition*. CENGAGE Learning.
Malliarakis, C., Satratzemi, M., & Xinogalos, S. (2014). Integrating learning analytics in an educational MMORPG for computer programming. In *In Proceedings of the 14th international Conference on advanced learning technologies, july 7 – 10*. Athens, Greece.
Marques, B. R. C., Levitt, S. P., & Nixon, K. J. (2012). Software visualisation through video games. In *In Proceedings of SAICSIT'12, October 01-03*. Pretoria, South Africa.
Masso, N., & Grace, L. (2011). Shapemaker: A game-based introduction to programming. In *In Proceedings of the 16th international Conference on computer games (CGAMES)*. July 27-30, Louisville, KY, USA.
Miljanovic, M. A., & Bradbury, J. S. (2017). RoboBUG: A serious game for learning debugging techniques. In *In ICER '17 proceeding of the 2017 ACM conference on international computing education research* (pp. 93–100).
Mitamura, T., Suzuki, Y., & Oohori, T. (2012). Serious games for learning programming languages. In *In Proceedings of the 2012 IEEE international Conference on systems, Man and cybernetics (SMC), October 14 – 17, Seoul, South Korea*.
Morsi, R., Richards, C., & Rizvi, M. E. (2010). Work in progress – BINX: A 3D XNA educational game for engineering education. In *In Proceedings of the 40th Frontiers in education conference*. October 27 -30, Washington DC.
Naul, E., & Liu, M. (2020). Why story matters: A review of narrative in serious games. *Journal of Educational Computing Research, 58*(3), 687–707. https://doi.org/10.1177/0735633119859904
Polack-Wahl, J. (1999). Incorporating the client's role in a software engineering course. In *Proceedings of the 30th SIGSCE technical symposium on computer science education* (pp. 73–77). Los Angeles, USA: ACM Press.
QAA. (2012). Understanding assessment: Its role in safeguarding academic standards and quality in higher education: A guide for early career staff. Retrieved 15 October 2016

from http://www.qaa.ac.uk/en/Publications/Documents/understanding-assessment.pdf .
Rosas, R., Nussbaum, M., Cumsille, P., Marianov, V., Correa, M., Flores, P., et al. (2003). Beyond nintendo: Design and assessment of educational video games for first and second grade students. *Computers & Education, 40*, 71–94.
Sauvé, L. (2010). Effective educational games. In D. Kaufman, & L. Sauvé (Eds.), *Educational gameplay and simulation environments: Case studies and lessons learned* (pp. 27–50). Portland, OR: Book News, Inc.
Scottish Funding Council. (2016). *Articulation*. Retrieved 20 August 2019 from http://www.sfc.ac.uk/web/FILES/CMP_AccessandInclusionCommittee16February_17022016/AIC16_07_Articulation_Policy_Update.pdf.
Scottish Qualifications Authority (SQA). (2018). Young Scot report. Retrieved January 2024 from https://www.sqa.org.uk/sqa/files_ccc/221018_SQAAssessmentFutures_YoungScotReport.pdf.
Shabana, S., & Chen, J. X. (2009). Simplifying algorithm learning using serious games. In *In the proceedings of the 14th Western Canadian conference on computing education, Burnaby, British Columbia, Canada. May 1 – 2, 2009* (pp. 34–41).
Shabanah, S. S., Chen, J. X., Wechsler, H., Carr, D., & Wegman, E. (2010). Designing computer games to teach algorithms. In *In Proceedings of the seventh international Conference on information technology: New Generations on information technology: New generations, April 12 – 14, Las Vegas, NV, USA*.
Shaw, K., & Dermoudy, J. (2005). Engendering an empathy for software engineering. In *, Vol. 42. Proceedings of the 7th australasian computing education conference (ACE2005), Newcastle, Australia* (pp. 135–144).
Sheth, S., Bell, J., & Kaiser, G. (2013). A competitive-collaborative approach for introducing software engineering in a CS2 class. In *In proceedings of the 26th international conference on software engineering education and training (CSEE&T), may 19 – 21, San Francisco, CA, USA*.
Simsarian, K. (2003). Take it to the next stage: The roles of role playing in the design process. In *In proceedings of CHI 2003, April 5-10, 2003, Ft. Lauderdale, Florida, USA. ACM 1-58113-637-4/03/0004*.
Siu, K., Guzdial, M., & Riedl, M. O. (2017). Evaluating singleplayer and multiplayer in human computation games. In *In Proceedings of the 12th international Conference on the Foundations of digital games*. Article No. 34.
Sun, Q., Phelps, A. M., Egert, C. A., & Bayliss, J. D. (2009). *Games in the classroom at the rochester Institute of technology*. IEEE Computer Society.
Sung, K. (2009). Computer games and traditional CS courses. *Communications of the ACM, 52*(12), 74–78.
Tan, P.-H., Ting, C.-Y., & Ling, S.-W. (2009). Learning difficulties in programming courses: Undergraduates' perspective and perception. In *Proceedings of the 9th international conference on computer technology and development* (Vol. 1, pp. 42–46). Washington, DC, USA: IEEE.
Tang, S., & Hanneghan, M. (2010). Designing educational games: A pedagogical approach in design and implementation of educational games: Theoretical and practical perspectives. In P. Zemliansky, & D. Wilcox (Eds.), *Hershey, PA: IGI global, ISBN13: 9781615207824* (pp. 108–125).
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)* (Vol. 2014, pp. 1–7). IEEE.
Wassila, D., & Tahar, B. (2012). Using Serious game to simplify algorithm learning. In *In proceedings of the international conference on education and e-learning innovations. July 1 – 3, Sousse, Tunisia*.
Wei, C.-S., Chen, Y., & Doong, J.-G. (2009). A 3D virtual world teaching and learning platform for computer science courses in second Life. In *In proceedings of the international conference on computational intelligence and software engineering*.
Winn, B. M. (2009). The design, play, and experience framework. In R. E. Ferdig (Ed.), *Handbook of research on effective electronic gaming in education* (Vol. III, pp. 1010–1024). Hershey PA: Information Science Reference.
Wong, Y. S., Yatim, M. H.b. M., & Tan, W. H. (2015). Learning object-oriented programming with computer games: A game-based approach. In *Proceedings of the 9th European Conference on games-based learning (ECGBL), Steinkjer, Norway, October 8-9*.
Xiao, D., & Miller, R. C. (2014). A multiplayer online game for teaching software engineering practices. In *In Proceedings of the first ACM conference on Learning @ scale conference*. Atlanta, Georgia, USA March 4 – 5.
Yan, L. (2009). Teaching object-oriented programming with games. In *Proceedings of the sixth international conference on information technology: New generations* (pp. 969–974). IEEE Computer Society.
Yap, J. (2011). Virtual world labyrinth: An interactive maze that teaches computing. In *In proceedings of the defence science research conference and expo (DSR)*. August 3 – 5, Singapore.