A Framework for Games-Based Construction Learning: A Text-Based Programming Languages Approach

André Luiz França Batista[1], Thomas Connolly[2] and José Andre Peres Angotti[1]
[1]Federal University of Santa Catarina, Florianópolis, Brazil
[2]University of the West of Scotland, Paisley, Scotland
andreluiz@iftm.edu.br
thomas.connolly@uws.ac.uk
angotti@ced.ufsc.br

Abstract: Computer programming is a challenge for students and a major reason why people avoid Computer Science courses. Investigating alternative teaching methods is essential to encourage students to learn and understand the concepts of programming. The use of games in learning and training is advocated and supported by many researchers due to its motivational and attractive features. This study focuses on an approach that supports the use of learning methodologies based on constructionist activities. Therefore, a pedagogical framework is proposed to guide lecturers who teach programming on how to integrate games-based learning to present coding concepts in the context of familiar real-world applications like computer games development. The framework is supported by motivational and attractive game features in conjunction with the authentic and meaningful aspects of constructionist activities for Games-Based Construction Learning (GBCL). This paper summarises and presents a framework model based on a literature review and a panel of experts, with a view to performing a two-stage process to validate this framework. The paper discusses the design and validation of the framework and proposes actions regarding its implementation.

Keywords: Computer science education, games-based construction learning, pedagogy

1. Introduction

Programming teaching has a fundamental job in the formation of professionals in Information Communications Technology (ICT). It is a curricular unit of primary importance due to its contribution in the construction of logical and abstract thought, and as the basis of a solid conceptual foundation for other disciplines in the course (Dorn, Tew and Guzdial, 2007; Bayliss, 2009; Corral et al., 2014). Learning programming is one of the first and main challenges faced by students of ICT courses. High failure and dropout rates are a reflection of the obstacles faced by students in these initial programming disciplines (Ludi, 2011; Anderson et al., 2014).

The methods for teaching programming disciplines – Algorithms and Data Structures, Programming Logic – are diversified. In the traditional method, the teacher transmits his or her knowledge to students. Subsequently, s/he proposes solutions to problems so that the student absorbs such information without critical analysis and therefore without building their own solutions (Robins, Rountree and Rountree, 2003; Vihavainem, Airaksinen and Watson, 2014). Some researchers suggest the use of Game-Based Learning (GBL) as the main methodology in teaching computational concepts (Connolly, Boyle and Hainey, 2007; Todorova and Moffat, 2013; Soflano, Connolly and Hainey, 2015). In the particular case of programming education, there is evidence of good success rates in the use of games as an alternative method to traditional methodologies (Leutenegger and Edgington, 2007; Bayliss, 2009).

Some educators have explored the advantages of GBL in programming teaching using a constructivist approach (Sung, 2009; Bermingham et al., 2013; Majgaard, 2013; Wilson, Hainey and Connolly, 2013). The adoption of a constructivist learning methodology is supported by some researchers to engage and motivate the students in the learning process, and provided to build knowledge through practical activities (Papert and Harel, 1991; Kafai, 2006). Students building games in the classroom is a novel learning method but this idea has been in use for almost three decades (Kafai and Burke, 2015).

Within this constructivist context, some authors make use of specific educational digital environments, like Alice, Scratch or Greenfoot (Utting et al., 2010; Fincher et al., 2010), leaving aside conventional text-based programming languages. Such environments, although very useful for beginners, are not suitable for building industrial applications with the levels of complexity required by the market (Mselle and Kondo, 2014; Sung and Snyder, 2014). Therefore, alternative methods focused on the use of these conventional programming languages need to be explored by educators.

In summary, the following research question can be identified:

- In order to motivate and engage students in introductory programming courses, what framework would be suitable to implement Games-Based Construction Learning (GBCL) using text-based programming languages?

This paper discusses the literature surrounding GBCL in introductory programming courses in the next section and then presents an initial framework for the implementation of GBCL into these courses.

2. Related work

2.1 Games-based construction learning

According to Papert and Harel (1991), the theory of constructionism regards learning as a process in which the learners construct their knowledge by interacting with the subject of study. The constructionist perspective puts game-making in the hands of children to encourage their knowledge through developing their own games. In this approach, the aim of the game-making tool is to support such an activity by providing an appropriate learning environment. The constructionists focus their efforts on providing students with greater opportunities to construct their own games and to construct new relationships with knowledge in the process, rather than embedding lessons directly in games (Kafai, 2006).

Kafai and Burke (2015) also suggests that few people have chosen to take this different path of making instead of playing games for learning. Due to their simplicity and ease of use, some educators have chosen particular environments for programming such as Alice, Scratch, Greenfoot and other visual-programming tools (Utting et al., 2010; Fincher et al., 2010). However, these environments are not suitable for building industrial applications. With this in mind, other educators have deviated from these tools and sought to use a constructionist approach using text-based programming languages in introductory programming courses (Mselle and Kondo, 2014; Sung and Snyder, 2014; Weintrop and Wilensky, 2015). It is relevant to note that this present study is not aimed at comparing the different approaches to teaching programming. In fact, we do not question the effectiveness of any visual-programming methods as educational tools but suggest that a text-based programming paradigm is indeed compatible with the Computer Science Curriculum, able to reach young students with attractive content.

2.2 Text-based programming languages approach

Considering ICT courses, many educators have identified and taken advantage of the younger generation's interests and familiarity with digital games and integrated related content into their introductory programming courses. The ways of using games in introductory programming courses range from playing, analysing and developing games. In this paper we focus on a constructivist approach, i.e. how the game-making approach can engage and motivate young students in Computer Science courses using a text-based programming languages approach.

Many authors have implemented the GBCL approach using different text-based programming languages. The languages that have been used are C# (Sung et al., 2011; Sung and Snyder, 2014; Isomöttönen, Lakanen and Lappalainen, 2012; Wang, 2011; Corral et al., 2014; Giordano and Maiorana, 2013), Java (Drake and Sung, 2011; Lewis and Massingill, 2006; Schuster, 2010; Ludi, 2011; Soares Fonseca and Martin, 2015), Objective-C (Dickson, 2010), C/C++ (Paralic and Pietrikova, 2014; Leutenegger and Edgington, 2007). In these papers, the authors describe their experiences of the implementation and the students' feedback, which is useful as a support to build a framework with reference guidelines to help other educators to implement GBCL in their own learning environment. It is important to state that these papers intend to help students to understand abstract programming concepts rather than concepts specific to building games.

2.3 Literature review

We carried out a literature review to identify previous game-based construction research using ACM, Science Direct and IEEE (Institute of Electrical and Electronics Engineers) electronic databases. The following search terms were used:

(game OR gaming) (based OR themed) (development OR construction OR design OR making) (teaching OR learning OR course OR education)

In order to focus on empirical work undertaken with games construction using text-based programming languages within Computer Science courses, we selected 12 papers (see Table 1).

Table 1: GBCL with text-based programming languages

Author	Programming language	Study/Approach/Briefing
Corral et al. (2014)	C#	Presents a contribution to the teaching of Object Oriented Programming (OOP) languages through a game-oriented approach based on interaction with tangible user interfaces (TUIs).
Dickson (2010)	Objective-C with OpenGL	Details of the design and implementation of a video game design course that not only enables students to create video games but also to learn standard Computer Science skills.
Drake and Sung (2011)	Java	Analyses the inclusion of board, card and dice games as programming assignments in introductory programming courses.
Leutenegger and Edgington (2007)	C++	Describes the use of the "Game First" approach to teaching introductory programming concepts via two-dimensional game development.
Lewis and Massingill (2006)	Java	Describes a project that has been used to teach Computer Science in which the students develop a two-dimensional game using a framework written by one of the authors.
Ludi (2011)	Java	Describes the experiences of developing an educational game as a team project in a 10-week introductory software engineering course for second-year undergraduates.
Paralic and Pietrikova (2014)	C	Examines and discusses recent developments in teaching and learning programming, based on game-based learning and learning by game creation.
Schuster (2010)	Java	Describes the experiences of teaching CS1 using the ACM Java library to write arcade game programs.
Soares, Fonseca and Martin (2015)	Java	Discusses the implementation of a problem-based learning environment and the delivery of the entire introductory programming course in the context of game design.
Sung and Snyder (2014)	C# with Microsoft XNA framework.	Demonstrates the general effectiveness of the text-based programming paradigm to teach Computer Science principles.
Sung et al. (2011)	C# with Microsoft XNA framework.	Design, implementation and assessment of game-themed programming assignment modules.
Wang (2011)	C# with Microsoft XNA Framework	Describes an extensive evaluation of introducing a game project to a software architecture course to find out if game development projects can successfully be used to teach software architecture.

By reviewing the studies in regard to the approach taken by educators, the works found in this literature review can be considered as a game-themed programming course; in other words, programming courses with game-making activities. All these classes have made games a part of their programming learning experience. In each case, games were built according to programming topics previously covered in the classroom.

Although using GBCL in the classroom is still an emerging field, most of the cited papers report a significant increase in engagement and motivation of the students in these courses (Sung et al., 2011; Leutenegger and Edgington, 2007). Based on these results, it is possible to understand that the integration of games into introductory programming courses is a promising strategy to arouse the interest of young students. Considering the challenges and difficulties of these initial courses, it is important to note that this strategy must be adopted carefully by interested teachers and departments. For this reason, a proper framework must be followed in order to implement this approach successfully.

3. Methods

3.1 Literature research

A literature review was performed to identify any previous GBCL implementation frameworks. The electronic databases used were ACM, Science Direct and IEEE. The following search terms were used:

(strategies OR guidelines OR framework OR methodology OR approach) (based OR themed) game (development OR construction OR design OR making) (teaching OR learning OR course OR education)

This literature review returned papers that discuss the implementation of GBCL in primary education (Bermingham et al., 2013; Wilson, Hainey and Connolly, 2013) and higher education (Drake and Sung, 2011; Sung et al., 2011). Table 2 provides an overview of implementation frameworks/models for GBCL in education.

Table 2: GBCL frameworks

Author	Framework/Model	Purpose
Bermingham et al. (2013)	Model for game making in collaboration for learning.	Presents recommendations for collaborative game-making in the classroom.
Drake and Sung (2011)	Guidelines to teaching Introductory Programming with popular board games.	Presents some issues to consider when choosing a game for a programming assignment.
Sung et al. (2011)	Game-themed programming assignment modules.	Presents self-contained modules that enable faculty to begin exploring and developing their own expertise and materials to teach programming with games.
Wilson, Hainey and Connolly, (2013)	Framework for games-based construction learning.	Aims to give teachers a starting point for using GBCL in the classroom.

Bermingham et al. (2013) present in their paper a project named Making Games in Collaboration for Learning (MAGICAL). This project aims to explore the use of collaborative game-making as a pedagogic model. The main purpose is to verify whether this kind of approach can support collaboration, problem-solving, creativity and digital literacy skills. According to the authors, "collaborative game making provides a model in which learners can work together to create something that is meaningful for them, giving them input into both the process and product, and facilitating the development of a range of 21st Century Skills, such as digital literacy" (Bermingham et al., 2013).

Drake and Sung (2011) discuss the inclusion of board, card and dice games as programming assignments in introductory programming courses. The authors report that the implementation of such games generally requires less background knowledge from the lecturer. Furthermore, they raise some issues to consider when picking a game for a programming assignment, listing 32 specific games that are suitable for teaching the main topics in introductory programming. The paper analyses the implementation of some of these games and shows their successful use as programming projects for the students.

Sung et al. (2011) propose modules called Game-Themed Assignments (GTA) that are designed specifically for faculty members who lack background expertise in games. According to the authors, these GTA modules are independent and each is a self-contained game-like programming assignment that challenges students on concepts pertaining to a specific curriculum topic area. The paper also discusses design, implementation and assessment of these GTA modules in order to help faculty members use this approach, enabling them to begin exploring and developing expertise and materials to teach with games.

Wilson, Hainey and Connolly (2013) propose a framework for the introduction of GBCL into primary education by utilising the visual tool, Scratch. The authors present a generalised framework based on empirical work carried out in three primary schools in Glasgow, Scotland, in 2011/2012. According to the authors, some teachers feel they face many obstacles when trying to implement GBCL in school so the framework's purpose is help these teachers by giving them a starting point for using GBCL in the classroom.

The analysis of these frameworks shows us that the key points to be considered in the implementation of our framework should be:

- Preliminary Audit
- Planning and Designing
- Teaching
- Evaluation

3.2 Experts evaluation

From the above literature review findings, we designed an initial four-stage model and framework for integrating a GBCL in introductory programming courses with text-based language approach. To evaluate that prototype an expert evaluation was used. In the expert evaluation four specialists in GBL and Computer Science Education (CSE) had been asked to review the prototype and give feedback. A short background information about the specialists are provided in the Table 3.

Table 3: Experts background information.

Expert no.	Teaching position	Years of experience in teaching	Publications related to GBL or CSE in the last 5 years
Expert #1	Computing (Game Studies)	9	11
Expert #2	Computing (Educational Technology)	13	12
Expert #3	Computing (Programming Languages)	11	14
Expert #4	Computing (Programming Languages)	10	6

The goal of the evaluation was to obtain feedback about the prototype and thereby detect potential weaknesses before the framework was used by a lecturer into an introductory programming course. After review the initial model the expert received and filled a survey with five open questions: one for each stage of the prototype and one for the whole prototype. Furthermore, they were asked whether they had suggestions for improvement.

The feedback of the experts overall was positive and the following suggestions were made.

- First stage: The lecturer should consider the coding policies and styles. A guide for best-practices and most common rules could be set in this first stage.
- Second stage: (a) The materials should be designed to make the students' conditions as equal as possible; (b) When selecting the game, it should be the same (or as similar as possible) as the existing and well-known games; (c) Plan the assessment in advance, i.e., during the planning stage.
- Third stage: The teacher should present the games as a process (or problem), not as a game for having fun. This way the academic integrity of the course will be preserved.
- Fourth stage: The feedbacks should be double sided, i.e., from teachers to students and from students to teachers.

These suggestions made by the experts were used to make improvements in the initial four-stage framework. The design of the refined framework – revised by a panel of experts – is described in the next section.

4. The refined framework design

Based on the literature review and the observational findings from the experts' evaluation, we present the refined model and framework for integrating a GBCL in introductory programming courses with teaching strategies. An overview of the framework's implementation is presented in Figure 1 and Table 3. This model presents the various phases for best practice when teachers wish to implement GBCL into the programming courses curriculum and also shows a high-level summary of the framework in Table 3 which provides a more detailed view of implementation of GBCL within the introductory programming courses curriculum.

The framework has four stages: Preliminary Audit; Planning and Designing; Teaching; Evaluation. Each stage has a set of specific steps with guidelines to help teachers through the GBCL implementation. The implementation is conducted by the teachers and at each stage they must demonstrate strong leadership in order to captivate the students and progress with the implementation.

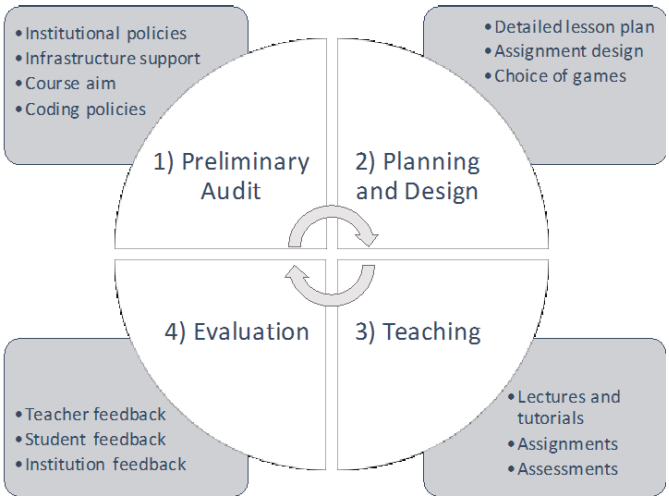

Figure 1: Refined GBCL framework implementation diagram

The first stage is the Preliminary Audit stage. In this first part, the teacher focuses effort to check three components: (a) institutional policies; (b) infrastructure support; (c) course aim; (d) coding policies/styles. In order to avoid the violation of any institutional rule, the departmental committee should be aware of and agree to significant changes to the course. To ensure appropriate infrastructure support, the teacher must check whether the resource tools (programming platforms, integrated development environment (IDE), programming libraries) are freely available and whether the related infrastructure requirements are simple and straightforward. Course objectives must be thoroughly reviewed to guide the teacher in the proper design of lectures and assignments in the second stage. Coding policies/styles should be considered because companies usually have their own coding policies and this is waiting for students after they leave school.

The second stage is the Designing stage. Once there is a clear picture of the circumstances in terms of institutional policies, infrastructure support and course aim, it is time to consider the pedagogical aspects of implementation. At this stage, the teacher plans and designs the lectures and assignments. Detailed lecture plans should consider the number of PCs/laptops available, the number of students in the class and the objectives the teacher wishes to achieve for the learning outcomes. Tutorials must be designed considering adequate context and pedagogy in order to preserve the academic integrity of the course. The assignment design considerations rely on neutrality and levels of challenge. It is relevant to observe material neutrality in order to avoid alienation of under-represented groups. The materials built should be gender and expertise neutral. Violent topics, unnecessary competition and superfluous graphics should be discouraged. Students' skills vary widely within and between classes. The levels of challenge designed for the game programming projects should be tweaked by changing how much of the work the students should perform. The level of challenge can be determined from making little modifications in an existing game to developing an entire game from scratch.

The assignment design also includes the selection of the games to be used. Teachers may prefer to choose an existing game rather than invent a new one. Picking games that the students are familiar with requires less time to explain the rules. Highly complex games should be avoided and the rules should be simple and easy to implement. Length of gameplay should be considered. Picking games that are quick to play provides faster feedback and also requires less time to implement.

The third stage is the Teaching stage. Once teachers have completed the second stage, the lessons can then be implemented. It is relevant to use game-making examples for each topic covered. Students should be encouraged to participate actively in order to guide the construction of their knowledge. To avoid disruptions in the learning process, the assignments should be self-contained without dependencies from other assignments. After lessons and tutorials have been implemented, a work evaluation must be performed. This is important to allow teachers to understand how students are progressing and whether they are achieving their learning goals. The assessment can be performed by the students themselves, by peer review or by self-assessment. Assessments should be developed before any lessons are implemented and should incorporate the learning objectives of the class teacher.

The fourth stage is the Evaluation stage. In this phase, the teacher's and students' feedback are collected and analysed. The entire implementation process is reviewed in order to improve future implementation. This final stage of the model aims to review the implementation. This review examines how teachers felt about the lessons and how the students completed them. At this stage of evaluation, we suggest that teachers collect data using surveys. Based on the analysis of collected and teaching experiences, the teachers can improve the next implementation process framework. For an overall understanding of the framework implementation, the experience data collection is important from all three previous stages, not only the Teaching stage.

Table 3: Refined GBCL framework implementation

1) Preliminary Audit	
Component	**Points to consider**
Institutional policies	Policies should be checked to avoid any violations regarding significant changes to the courses.
Infrastructure support	Resource tools must be free, simple and available.
Course aim	Course objectives should be analysed in order to avoid spoiling academic integrity.
Coding policies/styles	A guide for best-practices and most common rules could be set here.
2) Planning and Design	
Component	**Points to consider**
Lectures	Detailed lesson plans are needed, considering context, pedagogy and material neutrality.
Assignments	It is helpful when designing to consider levels of challenge and neutrality (gender and experience).
Choice of games	Choose a familiar game, which is quick to play and to implement.
3) Teaching	
Component	**Points to consider**
Lectures	Use game-making examples for each topic covered.
Assignments	Should be self-contained with no dependencies from other assignments.
Assessments	Employ peer-review and self-assessment. Verify whether the students understand the programming concepts taught and how to apply it in game development.
4) Evaluation	
Component	**Points to consider**
Teacher feedback	Review lessons and make changes as needed to teach again. The confidence of teachers to teach the skills will grow and continuously improve.
Student feedback	Students analysing lessons can help teachers make changes for future lessons.
Institution feedback	Institutional staff should be consulted on the impacts that the framework's implementation causes in the institution beyond the classroom.

5. Conclusions and futures directions

This framework is the first part of a project funded by the Brazilian government. The final results of the research have not been fully completed and interpreted. From the experience of accomplishing this paper, we still have the framework's implementation limitation. In an effort to continue the progression of this research, a two-phase validation process is in progress. The first phase had been performed by involving a panel of GBL experts invited to examine our GBCL framework model components and answer a survey (see Section 3.2). The contributions from these GBL experts had been used to refine our framework (see Section 4).

IN THE SECOND PHASE, based on the findings from the first phase, the refined framework will be implemented by a lecturer into an introductory programming course. IT IS IMPORTANT TO HAVE AN INDEPENDENT EVALUATOR ASSESSING THE INTEGRITY OF THE MODEL. This will help the researcher to present the issues and difficulties of the implementation.

By reviewing the literature, it is clear that GBCL is only just beginning to become more widely used within ICT courses. However, educators feel there are many barriers they face when trying to implement GBCL in their classes. The framework proposed intends to be a useful guide to support teachers in facing these barriers.
This paper shows that GBCLs have the potential power to motivate and engage students in an introductory programming course by using traditional text-based language. As stated before, this is an ongoing research and we hope that by completing it, this study will provide a useful guide to educators, teachers and researchers in the area of GBL. As a result of this study, we wish to help educators better to prepare today's ICT students for the computational challenges of tomorrow.

Acknowledgements

This work was supported in part by the Brazilian Federal Agency for Support and Evaluation of Graduate Education (CAPES) grant BEX 6369/15-4. The first author wishes to thank the Federal Institute of Triângulo Mineiro (IFTM) for the authorisation for sabbatical leave to carry out his doctoral studies, under the terms of processes numbers 23202.000368/2013-20 and 23202.000188/2015-18. All opinions, findings, conclusions and recommendations in this work are those of the authors and do not necessarily reflect the views of the CAPES or the IFTM.

References

Anderson, R. E., Ernst, M. D., Ordóñez, R., Pham, P. and Wolfman, S. A. (2014). Introductory programming meets the real world: using real problems and data in CS1. In Proceedings of the 45th ACM Technical Symposium on Computer Science Education (pp. 465-466). ACM.

Bayliss, J. D. (2009). Using games in introductory courses: tips from the trenches. In ACM SIGCSE Bulletin (Vol. 41, No. 1, pp. 337-341). ACM.

Bermingham, S., Charlier, N., Dagnino, F., Duggan, J., Earp, J., Kiili, K., Luts, E., Van Der Stock, L. and Whitton, N. (2013). Approaches to collaborative game-making for fostering 21st century skills. In European Conference on Games Based Learning (pp. 45-52). Academic Conferences International Limited.

Connolly, T. M., Boyle, E. and Hainey, T. (2007). A survey of students' motivations for playing computer games: A comparative analysis. In Proceedings. 1st European Conference on Games-based Learning (ECGBL), Scotland.

Corral, J. M. R., Balcells, A. C., Estévez, A. M., Moreno, G. J. and Ramos, M. J. F. (2014). A game-based approach to the teaching of object-oriented programming languages. Computers & Education, 73, pp. 83-92.

Dickson, P. E. (2010). Experiences building a college video game design course. Journal of Computing Sciences in Colleges, 25(6), pp. 104-110.

Dorn, B., Tew, A. E. and Guzdial, M. (2007). Introductory computing construct use in an end-user programming community. VLHCC, 2007, Visual Languages and Human-Centric Computing, IEEE Symposium on, Visual Languages and Human-Centric Computing, IEEE Symposium on 2007, pp. 27-32.

Drake, P. and Sung, K. (2011). Teaching introductory programming with popular board games. In Proceedings of the 42nd ACM Technical Symposium on Computer Science Education (pp. 619-624). ACM.

Fincher, S., Cooper, S., Kölling, M. and Maloney, J. (2010). Comparing Alice, Greenfoot & Scratch. In Proceedings of the 41st ACM Technical Symposium on Computer Science Education (pp. 192-193). ACM.

Giordano, D. and Maiorana, F. (2013). Object oriented design through game development in XNA. In Interdisciplinary Engineering Design Education Conference (IEDEC), 2013 3rd (pp. 51-55). IEEE.

Isomöttönen, V., Lakanen, A. J. and Lappalainen, V. (2011). K-12 game programming course concept using textual programming. In Proceedings of the 42nd ACM Technical Symposium on Computer Science Education (pp. 459-464). ACM.

Kafai, Y. B. (2006). Playing and making games for learning instructionist and constructionist perspectives for game studies. Games and Culture, 1(1), pp. 36-40.

Kafai, Y. B., and Burke, Q. (2015). Constructionist gaming: Understanding the benefits of making games for learning. Educational Psychologist, 50(4), pp. 313-334.

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

Lewis, M. C. and Massingill, B. (2006). Graphical game development in CS2: a flexible infrastructure for a semester long project. ACM SIGCSE Bulletin, 38(1), pp. 505-509.

Ludi, S. (2011). The benefits and challenges of using educational game projects in an undergraduate software engineering course. In Proceedings of the 1st International Workshop on Games and Software Engineering (pp. 13-16). ACM.

Majgaard, G. (2013). Creating games in the classroom - from native gamers to reflective designers. In European Conference on Games Based Learning (pp. 352-358). Academic Conferences International Limited.

Mselle, L. J. and Kondo, T. S. (2014). Against the "Hello World". International Journal of Computer Applications, 95(26), pp. 17-22.

Papert, S. and Harel, I. (1991). Situating constructionism. Constructionism. Norwood.

Paralic, M. and Pietrikova, E. (2014). Learning by game creation in introductory programming course: 5-Year-long study. In Emerging eLearning Technologies and Applications (ICETA), pp. 391-396, 2014 IEEE 12th International.

Robins, A., Rountree, J. and Rountree, N. (2003). Learning and teaching programming: A review and discussion. Computer Science Education, 13(2), pp. 137-172.

Schuster, D. L. (2010). CS1, arcade games and the free java book. In Proceedings of the 41st ACM Technical Symposium on Computer Science Education (pp. 549-553). ACM.

Soares, A., Fonseca, F. and Martin, N. L. (2015). Teaching introductory programming with game design and problem-based learning. Issues in Information Systems, 16(3), pp. 128-137.

Soflano, M., Connolly, T. M. and Hainey, T. (2015). An application of adaptive games-based learning based on learning style to teach SQL. Computers & Education, 86, pp. 192-211.

Sung, K. (2009). Computer games and traditional CS courses. Communications of the ACM, 52(12), pp. 74-78.

Sung, K. and Snyder, L. (2014). A case of computer science principles with traditional text-based programming languages. Journal of Computing Sciences in Colleges, 30(1), pp. 161-172.

Sung, K., Hillyard, C., Angotti, R. L., Panitz, M. W., Goldstein, D. S. and Nordlinger, J. (2011). Game-themed programming assignment modules: a pathway for gradual integration of gaming context into existing introductory programming courses. Education, IEEE Transactions on, 54(3), pp. 416-427.

Todorova, E. and Moffat, D. (2013). Evaluating the embedding of games based learning in a computing subject at university. In European Conference on Games Based Learning (pp. 776-784). Academic Conferences International Limited.

Utting, I., Cooper, S., Kölling, M., Maloney, J. and Resnick, M. (2010). Alice, Greenfoot, and Scratch - a discussion. ACM Transactions on Computing Education (TOCE), 10(4), pp. 17:1-11.

Vihavainen, A., Airaksinen, J. and Watson, C. (2014). A systematic review of approaches for teaching introductory programming and their influence on success. In Proceedings of the Tenth Annual Conference on International Computing Education Research (pp. 19-26). ACM.

Wang, A. I. (2011). Extensive evaluation of using a game project in a software architecture course. ACM Transactions on Computing Education (TOCE), 11(1), pp. 5:1-28.

Weintrop, D. and Wilensky, U. (2015). To block or not to block, that is the question: students' perceptions of blocks-based programming. In Proceedings of the 14th International Conference on Interaction Design and Children (pp. 199-208). ACM.

Wilson, A., Hainey, T. and Connolly, T. M. (2013). Using Scratch with primary school children: an evaluation of games constructed to gauge understanding of programming concepts. International Journal of Game-Based Learning (IJGBL), 3(1), pp. 93-109.