GameCode: Choose your Own Problem Solving Path

Simon Liénardy
Université de Liège, Montefiore Institute – Belgium
simon.lienardy@uliege.be

Benoit Donnet
Université de Liège, Montefiore Institute – Belgium
benoit.donnet@uliege.be

ABSTRACT

This abstract focuses on a CS2 course in which gamified homework exercises are provided to students instead of in-class exercise sessions. The course, provided to first-year Computer Science students, introduces a rigorous methodology to write programs using Loop Invariants [2], recursion, and basic data structures such as Files, Lists, Queues, and Stacks. In early 2020, the COVID-19 pandemic caused a lock-down in our country. The universities decided to fully switch to remote teaching. As the exercises sessions previously consisted of solving problems on a blackboard, we had to design in a hurry course materials that would cope with remote teaching. Instead of giving students yet another podcast in their course schedule, we gave them homework exercises, we called *GameCode*, that they could do at their own convenience. These exercises are inspired by GameBooks in which the reader can choose the path she takes to complete the story. With GameCode, students can choose their own solving path for each exercise. This can be related to *gamification* [1, 3–5].

CCS CONCEPTS

• **Theory of computation → Algorithm design techniques**;
• **Social and professional topics → Computing education**.

KEYWORDS

Loop Invariant, Problem Solving, CS2, Gamification, GameBook

1 ADAPTING GAMEBOOKS TO GAMECODE

Fig. 1 illustrates the principle of any GameCode exercise. A plain arrow represents a jump to a particular item in that same part of the document while a dashed one illustrates a jump between two parts of the documents. A GameCode exercise is made of three main parts: a general theoretical reminder on the subject covered by the exercise, possibly divided into several "Items" (in red), the description of the subject of the exercise (in purple), and the exercise resolution (in blue). This last part is divided in several "Steps" (labeled from 1 to m in fig. 1) corresponding to a division of our programming methodology and containing at least these parts: (*i*) Introduces formal notations that will be helpful in the following, (*ii*) Provides formal specifications of the problem, (*iii*) Finds a formal Loop Invariant (or a recursive formulation), and (*iv*) Builds the code upon the Loop Invariant (resp. the recursive formulation).

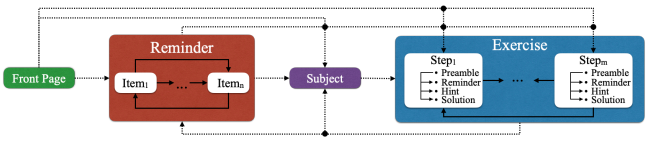

Figure 1: General architecture of any GameCode exercise.

Each Step is typically divided into four sub-parts: a "Preamble" allowing the student to directly jump to their preferred sub-part, a theoretical "Reminder", a "Hint", and the "Solution" to this step. Each sub-part contains pointers to each other.

Moreover, any GameCode exercise meets the following requirements: (*i*) stand-alone book (a GameCode exercise is self-sufficient and contains the minimal information to complete the exercise), (*ii*) just-in-time theory (the theoretical reminders are placed where they are needed and are as short as possible), (*iii*) "no spoilers hints" (hints given to the students never reveal a solution, nor a part of it), and, (*iv*) no single solution (several solutions are always possible and a GameCode exercise always references the course forum to discuss them).

2 PRELIMINARY EVALUATION

A preliminary evaluation on the number of GameCode downloads shows that few students took part in the exercises. This can be explained by the COVID-19 lock-down (lower motivation) but also by the fact that, at this stage of the year, lots of students have abandoned their courses. A survey was organized and we got 14 answers, confirming the number of courses abandonment. A slight majority declared they would have preferred podcast but if we remove those who did not do the exercise, the numbers of agreeing (43%) and disagreeing (50%) students are close.

REFERENCES

[1] Darina Dicheva, Keith Irwin, and Christo Dichev. 2019. OneUp: Engaging Students in a Gamified Data Structures Course. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. 386–392.
[2] Edsger. W. Dijkstra. 1976. *A Discipline of Programming*. Prentice-Hall, Inc.
[3] Brian Harrington and Ayaan Chaudhry. 2017. TrAcademic: improving participation and engagement in CS1/CS2 with gamified practicals. In *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education*. 347–352.
[4] Maria-Blanca Ibanez, Angela Di-Serio, and Carlos Delgado-Kloos. 2014. Gamification for engaging computer science students in learning activities: A case study. *IEEE Transactions on learning technologies* 7, 3 (2014), 291–301.
[5] Gina Sprint and Diane Cook. 2015. Enhancing the CS1 student experience with gamification. In *2015 IEEE Integrated STEM Education Conference*. IEEE, 94–99.

ICER '20, August 10–12, 2020, Virtual Event, New Zealand

ACM ISBN 978-1-4503-7092-9/20/08.
https://doi.org/10.1145/3372782.3408122