A FRAMEWORK FOR EVALUATING AND ORCHESTRATING GAME-BASED LEARNING THAT FOSTERS COMPUTATIONAL THINKING

Jalal Nouri1& Peter Mozelius2

1Department of Computer and Systems Sciences, Stockholm University (Sweden)
jalal@dsv.su.se

2Department of Computer and Systems Sciences, Mid Sweden University (Sweden)
peter.mozelius@miun.se

Keywords: game-based learning, computational thinking, programming, orchestration, evaluation

Abstract

For some years now many teachers around the world have explored programming with their pupils in K-9 education. Research has shown that educational games of different kinds are often utilized by teachers as a mean for teaching programming and developing computational thinking. However, teaching and learning programming and computational thinking through educational games is associated with a number of challenges.

One of those challenges are related to that teachers are presented with an ever increasing amount of educational games, and not supported with tools that, one the one hand, can help them evaluate the didactical affordances and potentials of specific games so they can select curriculum appropriate games, and on the other hand, that can help them design and orchestrate game-based learning activities. It is against such a background this paper presents a framework for the evaluation and orchestration of game-based learning activities that fosters computational thinking.

The framework consists of two dimensions, namely game mechanics and learning mechanics. These two dimensions consists of a number of aspects that teachers and researchers can take into account in order to evaluate and design activities, reaping the benefits of the didactical affordances of the games and the available scaffolding resources built inside games and available outside of them.

1 INTRODUCTION

The interest in programming education has been significantly strengthened recently as a consequence of an increasingly digital world that demands development of digital skills. As a result, more and more countries have introduced programming in their K-9 curricula, and more and more teachers are introducing programming in schools supported by visual block programming languages such as Scratch and various educational games developed for fostering programming skills.

This paper focus specifically on the fact that educational games and game-based learning environments such as Lightbot, Code.org and CodeCombat are becoming quite popular for developing computational thinking in K-9 education [1-2]. In a sense, the widespread efforts to develop computational thinking is driving consolidation and reinforcement of game-based learning in schools, which is a potentially very interesting development that bridges the gap between children’s habits and interests – i.e. their everyday activities outside of school – with formal learning in schools.

Furthermore, research on game-based learning has showed that educational games may promote problem-solving abilities, engagement and motivation [3-6] and result in the development of knowledge in various subjects [7-8].
However, there are a number of challenges related to game-based learning for developing computational thinking. One of those challenges are related to that teachers are presented with an ever increasing amount of educational games for computational thinking which – without guidance and support – makes it difficult for them to select for them didactically appropriate games [1]. That is, they lack guidance in evaluating the didactical affordances and potentials of specific games. Without such guidance there is a risk that fun games drive the development in schools instead of games that are both fun and learning oriented [9-11]. Another critical challenge relates to how educational games are utilized in schools, and more specifically, how game-based learning activities with focus on the development of computational thinking skills are orchestrated in order to maximize learning experiences; also an aspect that teacher might need guidance with. Providing children with educational games will in most of the cases not automatically result in learning without a reflective planning, regulation, coordination of learning [12]. Orchestration is, in this paper, defined as ‘the process of productively coordinating supportive interventions across multiple learning activities occurring at multiple social levels’ involving multiple contexts, and multiple tools and media [12-13].

It is against such a background this paper presents a framework for the evaluation and orchestration of game-based learning activities in programming education. We would like to argue that the framework, on the one hand, can support teachers (and researchers) to evaluate the didactical affordances and potentials of specific games so they can select curriculum appropriate games, and on the other hand, can help teachers to design and orchestrate game-based learning activities in programming education.

The framework consists of two dimensions, namely game mechanics and learning mechanics. These two dimensions consists of a number of aspects that teachers and researchers can take into account in order to evaluate and design activities, reaping the benefits of the didactical affordances of the games and the available scaffolding resources built inside games and available outside of them.

2 The proposed framework for orchestrating game-based learning activities for fostering computational thinking

The proposed framework that support the evaluation and orchestration of game-based learning activities consists of two dimensions: game mechanics and learning mechanics. Each of these two dimensions consist of specific categories. In the following, the dimensions and the corresponding categories are presented and exemplified through an analysis of the Lightbot game.

2.1 Game mechanics

This particular dimension describes the game narrative and basic game elements and mechanics that constitute the core of the game play and the game experience. It is crucial that the teacher who is the designer of a game-based learning activity, is aware of the game-mechanics and the game elements [14] that produce the play and learning experience.

An evaluation of Lightbot’s game mechanics and elements reveals that the game is built upon the scenario that the player controls a robot that are to walk on a grid with tiles towards a goal tile. On the way the player might be presented with obstacles on some tiles and be expected to light up certain tiles. The player is presented with a number of visual commands that represents either a spatial movement, the action to light a tile, or the calling of a procedure. The player is then expected to create a sequence of commands that instructs the robot to go towards the goal tile (see example in Picture 1 below).
2.2 Learning mechanics

This dimension consists of the categories: 1.) the learning objectives that can be pursued using the game; 2.) didactical elements and principles that support the pursuit of learning objectives; 3.) roles of learners; 4.) scaffolding structures built in the game; 5.) Scaffolding structures outside of the game; 6.) progression; and 7.) assessment.

2.2.1 Learning objectives

This particular category looks at the learning objectives that a particular game supports and that the teacher needs to orchestrate activities towards.

The Lightbot game covers several fundamental programming concepts. By design, two types of concepts are addressed, namely, as described by the designers: programming practices and control flow concepts. The programming practices supported by Lightbot are:

- Planning: the player need to plan how to instruct the robot to solve the level.
- Sequencing of instructions: the player must choose a sequence that executes the plan.
- Testing & Debugging: the player run the program and debug when a level is not solved correctly.

The control flow concepts on the other hand are:

- Commands: the player are instructing the robot through the use of commands.
- Procedures and loops: In order to solve certain levels, the players must use procedures. Thus, this learning element helps the player to extract patterns and re-use a set of commands multiple times.
- Mutual (recursive) loops: for some advance levels the player must call a procedure within a procedure itself thus fostering the development of the concept of recursive loops.

Other non-intended learning objectives might of course be pursued using the Lightbot game, a task for empirical research to identify.

2.2.2 Didactical elements and principles

This category describes the didactical elements and principles associated with the game that contribute or are thought to contribute to the achievement of learning objectives. With an awareness of these elements and principles, the teacher can more consciously design and orchestrate a sequence of activities. Therefore, we argue that it is important that teachers start out by identifying didactical elements and principles of educational games that are of interest to them.
For instance, in the case of Lightbot we can identify the following didactical elements and principles that support learning of programming:

- Learning through concrete objects: the player learns programming by using concrete objects and symbols (for example navigation symbols) that they understand thus building on their prior understandings [15].
- Reflecting and planning iteratively: the player need to reflect and plan a sequence of commands that solve the levels and refine based on testing and debugging.
- Applying: the player learn through applying, i.e. experimental learning and learning by doing [16].
- Instant feedback: the player is receiving instant feedback on actions which support self-regulated learning [17].
- Analysing and evaluating: As the game promotes testing and debugging, the player is supported to perform analysis and to engage in metacognitive activities [18].
- Motivation: the game increases users’ incentives to engage in learning of the programming concepts.

2.2.3 Roles of learners

This category focus on the roles of the learners while playing the educational game. More specifically it for instance looks at what type of support a particular game provides for collaborative learning and how the game play can be orchestrated taking individual and collaborative learning into account. If the game allows for collaborative play, how can then individual roles be defined within the group?

In the case of Lightbot, multiplayer mode is not a built-in option. However, there is possibilities to play with Lightbot collaboratively if orchestrated by the teacher. The orchestration in this context can entail that teachers give instructions about how the game should be played collaboratively (providing rules that determine individual roles) and through real-time monitoring and interventions that support pupils’ collaboration. Furthermore, one can also ask what the constraints of the game are for individual versus collaborative learning.

2.2.4 Scaffolding structures in the game

This particular category describes the scaffolding structures and mechanisms that are built into the game. Thus the category answers to how the game is supporting learners to achieve the learning objectives by providing for instance guidance and information.

Looking at Lightbot from this lenses can for instance reveal that learners are scaffolded in the following ways:

- Tutorials showing how the game works, explaining the commands and how to use them.
- Feedback on actions
- Clues that scaffold learners when they are stuck

In terms of the scaffolding limitations, what can be mentioned is, for instance, lack of extensive information about how a player should understand the concept/command of procedures.

2.2.5 Scaffolding structures outside of the game: roles of teachers

Related to abovementioned, this evaluation category focus specifically on the teacher role when pupils engage with the game. As such it asks, considering how the games affordances and constraints in terms of for instance provision of scaffolding and support of collaborative work, what should be the teacher’s role so that built-in scaffolds are complemented or enhanced? How can the teacher facilitate learning? What type of activities (for instance pre- and post activities) can the teacher plan in order to complement the learning constraints of the particular game?

In the case of Lightbot, the teacher might need to introduce the game, its rules and mechanics, as well as explaining learning goals and central concepts such as for instance commands, loops, testing and debugging, etc. The teacher could also arrange activities in which the teacher together with the class collective solves levels, or arrange collective discussions after game sessions in which central concepts are discussed and explained further.
2.2.6 Progression

This particular category addresses how we can orchestrate learning beyond the use of the specific educational game under evaluation, by considering how we can build on the learning that takes place when engaging with the game. That is, how can orchestrate learning that takes departure in the educational game at focus and builds on that using follow up lessons and activities, both unplugged and related to the use of digital means (which for example can be other educational games). By looking at progression possibilities we are, on the one hand, given an opportunity to evaluate the educational potential of games and, on the other hand, given an opportunity to plan/design for long-term learning trajectories.

Taking Lightbot as an illustrative example of such an evaluation, we could for instance conclude that Lightbot experiences can be built upon in the following way in order to ensure progression of knowledge and skills:

- Unplugged activities that introduce the formal concept of loops and procedures. (abstraction)
- Introducing visual block-programming such as Scratch in order to more freely apply the concepts introduced through Lightbot and deepen knowledge and skills.

2.2.7 Assessment

Assessment is another important aspect that needs consideration when a teacher is planning the orchestration of learning. Starting out, we must ask if and how the game incorporate automatic assessment or if it support assessment in any other way (and how it should be complemented and enhanced)? If there is no built-in assessment in the game, the teacher need reflect how the pupils learning and progress can be assessed.

The teacher should also ask what should be assessed considering the learning objectives supported by the game and the overall design of the learning activities.

In the case of Lightbot, no automatic assessment is built-in the game that can make teachers or the pupils themselves aware of what has been learnt. Therefore, the teacher need to design own assessment strategies and methods. A number of strategies and methods could be applied. For instance, the teacher could monitor how a pupil solves levels in the game in order to understand the skills developed. The teacher could also have a discussion with the pupils in an attempt to understand how the pupils understand specific concepts in the game (loops for example).

3 Conclusions

In the following paper we have presented a framework for the design of game-based learning activities involving educational games for programming curriculum. On the one hand, our intention with this framework is to highlight important aspects that needs to be taken into account when a teacher, researcher or a stakeholder is evaluating a specific educational game and the use of it. On the other hand, we view this framework as a conceptual tool that can support teachers to orchestrate game-based learning activities that reap the benefits of available scaffolding resources (built into the game and available outside of the game), and optimize supportive learning experiences based on reflective planning, regulation, and coordination. However, as a practical design and evaluation tool, the framework should not be understood as a recipe for successful orchestration, but rather as a support for designers (teachers or researchers) to reflect systematically and holistically on relevant aspects when evaluating and orchestrating game-based programming learning activities. The framework certainly needs to be further tested and its usability and generalizability further evaluated.

References


