Toward a learning game on Computational Thinking Driven by Competencies

Malak Kanaan, Sébastien Maillos, Mathieu Muratet

Toward a learning game on Computational Thinking Driven by Competencies

Malak Kanaan[1], Sébastien Maillos[1], Mathieu Muratet[1, 2], Sébastien Jolivet[3], Bertrand Marne[4] and Karim Sehaba[5]
[1] Sorbonne université, CNRS, LIP6, Paris, France
[2] INSHEA, Suresnes, France
[3] IUFE, Université de Genève, Genève, Switzerland
[4] ICAR UMR 5191, Université Lumière Lyon 2, Lyon, France
[5] LIRIS - Université Lumière Lyon 2, Lyon, France
malak.kanaan@lip6.fr
sebastien.maillos@lip6.fr
mathieu.muratet@lip6.fr
sebastien.jolivet@unige.ch
bertrand.marne@ens-lyon.fr
karim.sehaba@liris.cnrs.fr

Abstract: There are many learning games related to the theme of programming and computational thinking (CT) that exist nowadays. However, the main problem currently in France is that teachers lack training to teach K-12 learners to modern computer concepts. Teachers understand the competencies of referential but are not comfortable with and don't know how to develop teaching sessions for these competencies. Our proposition is: a learning game (LG) driven by competencies will assist teachers to develop teaching sessions with learners. Our main objective is to help teachers to appropriate learning games on CT. To do this, we conducted an analysis based on the PIAF (*Pensée informatique et Algorithmique dans l'enseignement Fondamental* - Computational Thinking in primary school) reference framework which aims at developing CT in elementary school. It lists a set of competencies related to the development of algorithmic thinking in fundamental education. We chose to analyse three existing learning games on CT with this framework. We selected these three learning games ("Blockly Maze", "Compute-it" and "Kodu") from 48 learning games identified on CT. These analyses show that many competencies aren't present in the games. Then we study how to link the PIAF framework in a LG. We work on the learning game named SPY in which the player has to program an agent to escape a maze. Our contribution for this paper is double: (1) an analysis of existing learning games and (2) propositions how to express PIAF competencies in gameplay features.

Keywords: Serious Games, Computational Thinking, Didactic Engineering, Fundamental Education, Learning Game.

1. Introduction and positioning

In 2016 it was established that "40% of Europeans have an insufficient level of digital competencies and that among those who have none (i.e., 22%) 42% are unemployed." (Van den Brande, 2016). This finding shows that learning digital competencies from an early age is crucial because mastering digital tools is becoming essential for all citizens.

In our research context, dealing with education in France, the question of computer science education dates back to the 1970s and is characterized by a balance between two conceptions of what should be taught. On the one hand, there is the idea of teaching IT and computer science as a tool. On the other hand, the idea of computer science as an academic subject, with its own concepts and methods to be taught (Baron and Drot-Delange, 2016).

In France, computer science has made a comeback in elementary school curricula, where it is referred to as "CT". According to Jeannette Wing, CT involves five cognitive abilities: (1) algorithmic thinking, (2) abstraction, (3) evaluation, (4) decomposition, and (5) generalization (Wing, 2006). The French researcher Gilles Dowek (Dowek, 2011) has structured computer science into four concepts so that the content taught provides an accurate picture of the discipline itself: (1) digital information, (2) algorithms, (3) languages, and (4) computing machines. However, teaching CT and computer science in France, at both elementary and high school levels, requires a major involvement from teachers (Kradolfer et al, 2014) in order to tackle this new discipline, due to a lack of training (both pre-service and in-service). Even if teachers don't master CT competencies, they are able to understand the competencies presented inside referential but are not comfortable with and don't know how to

develop teaching sessions for these competencies or to choose the appropriate learning tools/games for these competencies.

Our proposition is: a learning game driven by competencies will assist teachers to develop teaching sessions with learners.

The PIAF[1] competency framework (Parmentier and al, 2020), established in 2021, provides a classification of competencies related to the development of CT in basic education, as well as various pedagogical scenarios to help teachers transmit these competencies to their students. A competency framework is a single underlying construct framework that provides a rational, consistent and practical basis for the purpose of understanding people's behaviours at work and the likelihood of being able to succeed in certain roles and in certain environments (Bartram, 2006). There are already a lot of competency frameworks related to the CT and technology (CSTA K12, Computing, Digital Competencies, …). Parmentier et al (2020) synthesize the content of each framework and propose a synthesis centred on CT and targeted at basic education.

Today there are several dozen LGs dealing with learning programming in various forms (Saddoug et al, 2022). But these games don't explain how competencies are worked on and offer very little freedom for teachers to personalize their activities (Atlan et al, 2019).

In this article, we study a selection of LG meant to teach CT based on the PIAF reference framework. Then we study the possibility of modifying a LG so that it covers the competencies of the PIAF framework.

Next section presents the methodology of the research. In section 3 we analyse three learning games: "Blockly Maze", "Compute-It" and "Kodu". This analysis is based on the PIAF reference framework. We identify the competencies present and those missing in these three games. Section 4 introduces the game SPY and our contribution: we detail how we associate PIAF competencies and game features to design a learning game driven by competencies. We conclude this paper in section 5.

2. Methodology

We followed a didactic engineering research methodology (Barquero and Bosch, 2015). We focus in this paper on the two first phases of didactic engineering: the first concerns preliminary analysis (the analysis of learning games mentioned below through the PIAF competencies) and the second concerns a priori analysis and the design (proposition of a gameplay feature for each PIAF competency). Our goal is to provide a tool for the SPY game (SPY is presented in section 4) that enables teachers to select a set of competencies on CT and automatically build a level on SPY focused on these competencies.

Regarding the first phase, we identified 48 learning games from the review of learning games for CT (Saddoug et al, 2022). This review also combines a list of games identified from reviews related to CT (Lindberg et al, 2019; Miljanovic and Bradbury, 2018; Vahldick et al, 2014). We then focused our analysis on three of them: "Blockly Maze", "Compute-it" and "Kodu", we argue our choices of these three games in the next section.

As we state in the introduction, we choose to base our work on the PIAF framework. It mainly focuses on CT and offers six competencies to be developed with students. Each of these competencies is broken down into a set of three to seven sub-competencies for a total of 26 sub-competencies[2]:

1. C1 - Defining abstractions / generalizing is to develop the ability to name (groups of) objects or actions for later reference.
2. C2 - Compose/decompose a sequence of action aims to develop the ability to manipulate actions to solve tasks.
3. C3 - Controlling a sequence of actions aims to develop the ability to control when a sequence of actions can be performed.

[1] PIAF : *Pensée Informatique et Algorithmique dans l'enseignement Fondamental* (Computational Thinking in primary school)

[2] The complete list of competencies and their descriptions are available on the official PIAF website. https://piaf.loria.fr/en/outcomes/competencies-framework/, accessed May 12, 2022

4. C4 - Evaluating objects or sequences of actions is designed to develop the ability to think about the construction of sequences of actions.
5. C5 - Translate information into different representations should develop the ability to adapt to various equivalent representations
6. C6 - Constructing a sequence of actions iteratively aims to develop a work methodology based on an iterative approach specific to CT.

In next sections, we will refer to a competency as follows: CX.Y which means sub-competency number Y of category number X, for instance C2.4 corresponds to the fourth sub-competency of C2.

To conduct the analysis for the three games mentioned above, we proceeded as follows: Understanding the PIAF competencies, playing the entire games, identifying PIAF competencies worked on, performing a comprehensive analysis, filling in the table of competencies with 0s and 1s (1 if the competency is covered by the game, 0 if not), and lastly analysing results.

Concerning the second phase of the didactic engineering research methodology, we study how to design new game features in the SPY project to cover competencies defined by the PIAF framework. We aimed to establish features that could be enabled/disabled depending on the competencies we wish to work on. We started first by defining which competencies were already present in SPY. If this wasn't the case, or if a competency wasn't represented properly, we have designed new features to integrate it into the game.

3. PIAF based analysis of games dealing with CT teaching

There are already several reviews of learning games for CT in the literature. We conduct our analysis on the work of Saddoug et al (2022) which reviewed the adaptability of a set of learning games meant for teaching CT. This review identifies 48 games. Among these 48 games, we have focused on 3, with complementary features, in order to analyse them with the PIAF framework:

1. "Blockly Maze" (BM) because it is well tested and free software. Indeed, inspired by Scratch (Resnick et al, 2009), BM has been reused in many derivative works (for instance Hour of Code[3], Algorea[4] and more than 400 derivatives on the forge GitHub) and it is well used by teachers. The fact that the game is widely tested has particularly interested us to analyse it based on the PIAF framework.
2. "Compute-it" because it focuses on code reading to teach CT. We considered this game interesting in our study due to its uncommon gameplay structure (executing code rather than writing code).
3. "Kodu" because it was judged as the most adaptable game by Saddoug et al (2022), which highly interested us in our goal of proposing a helping tool for teachers.

3.1 Blockly Maze analysis

In Blockly Maze[5] students are introduced to simple computer programming concepts with a graphical editing tool that uses blocks instead of typed characters. To analyse BM based on the PIAF framework, we identified the competencies covered in each level. For instance, the C5.1 entitled "Represents objects or sequences of actions using a formal representation" consists of "being able to represent an object or an action using a precisely defined mode of representation" (Parmentier et al, 2020). In BM, at level 1 (and this is the case in all levels), several blocks with a precise meaning (linked to an action) have been used to describe a sequence of actions, for example, using the block "turn left" to say that you have to turn to the left.

Table 1: **Analysis of 3 games based on PIAF**

			Blockly Maze	Compute-it	Kodu
PIAF Competencies	C1	1.1	0	0	1
		1.2	1	1	0
		1.3	1	0	1
		1.4	1	1	1
		1.5	1	0	0
		1.6	0	0	1

[3] Hour of Code: https://hourofcode.com/fr/en, accessed May 18, 2022

[4] Algorea: https://algorea.org/main.html#/, accessed May 18, 2022

[5] Blockly Maze: https://blockly.games/maze?lang=en accessed May 19, 2022

			Blockly Maze	Compute-it	Kodu
		1.7	1	0	1
	C2	2.1	1	0	1
		2.2	1	0	1
		2.3	1	0	1
		2.4	1	0	1
		2.5	1	0	1
		2.6	0	0	1
	C3	3.1	1	0	0
		3.2	1	0	0
		3.3	1	0	0
		3.4	1	0	0
	C4	4.1	0	0	1
		4.2	0	0	1
		4.3	1	0	1
	C5	5.1	1	1	1
		5.2	1	0	1
	C6	6.1	0	1	0
		6.2	0	0	0
		6.3	0	0	0
		6.4	0	0	0
Sum	6	26	17	4	16

Furthermore, while trying to test the C6.3 entitled "Correct a sequence of actions to achieve a given goal" in BM, we didn't identify a situation that was proposing a wrong sequence of actions to the player and asking him/her to correct it. So, no levels in BM are designed to work specifically on C6.3 competency. This is the same for C6.1, C6.2 and C6.3.

The results from the analysis showed that there are 17/26 competencies of PIAF covered in the 10 levels of BM (see Table 1).

3.2 Compute-it analysis

Compute-it[6] focuses on code reading. After finishing Compute-it, students will have a good practice on how a computer reads and executes some code. It is a useful competency when discovering code written by other people and to debug his own code. In this game the player has to read code that contains positive and negative conditions, for and while loops, functions and recursivity.

This game is interesting because our analysis points to only 4 PIAF competencies (see Table 1). We noticed C1.2 (i.e., Differentiate atomic actions and non-atomic actions) for levels that contain basic actions and functions. We noticed C1.4 (i.e., Describe the outcome of a sequence of actions) because this competency includes the ability to execute a sequence of action that is the centre of Compute-it. We noticed C5.1 (i.e., Represent objects or sequences of actions in various formal representations) because players have to understand the formal language of Compute-it. We noticed C6.1 (i.e., Test a sequence of actions with respect to a given goal) because executing a program step-by-step is a key competency to test a sequence of actions.

We didn't check C2.X competencies because Compute-it doesn't ask the player to edit a sequence of actions. C3.X are more ambiguous because Compute-it includes control structures but always in a reading perspective and not in a writing perspective as described in the PIAF framework. We can see here that it was difficult to describe Compute-it finely with the PIAF framework.

3.3 Kodu analysis

Kodu[7] is a game aimed at teaching CT as well as creativity, storytelling and problem solving to students, by allowing them to create their own game and edit existing ones. Kodu is based on event-driven programming,

[6] Compute-it: https://compute-it.toxicode.fr/ accessed May 19, 2022

[7] Kodu: https://www.kodugamelab.com/ accessed May 19, 2022

the player defines the actions to associate to a set of events. The events are structured on pages and the player can control the page to execute.

The event-driven programming is an interesting feature of Kodu because it requires a different manner to resolve problems than procedural thinking. For instance, all the C3 competencies are void in Kodu because not applicable of the event-driven programming included in the game. Nevertheless, even if the programming paradigms are different between Kodu and PIAF a lot of competencies have been identified (16/26 see Table 1).

3.4 Conclusions from the analyses

In the three games that we have studied, we found that they don't cover all of the PIAF competencies, each of them has its orientation and its pedagogical choices. We also identified that some games address competencies outside the scope of PIAF like event-driven programming or recursivity. This incomplete analysis shows that even if the three games are about CT the means used to tackle these competencies are diverse.

In the 48 game references identified (Saddoug et al, 2022) none of them explains the links between the targeted competencies and the game mechanics. Within the rest of this paper, we detail how we link the PIAF competencies in the LG we are developing: SPY.

4. How to link competencies and game features?

SPY is a learning game focused on CT. It is an open-source project[8] developed by Sorbonne University. The game principle is to select from a list of actions, those that will allow a robot to get out of a maze. These actions are represented as blocks and the player has to build a sequence of actions before the robot executes them (see Figure 1).

To get out of the maze, the player has to program the robot to avoid obstacles. These obstacles have been built for educational purposes:

1. Sentinels end the level if they spot the robot (if the robot passes through their detection area). Players can select every sentinel to see the sequence of actions assigned to it. Players must know how to read programs, to understand it and to anticipate the sentinels' movements in order to create a proper sequence to reach the objective with the robot.
2. Doors can be activated with terminals. This feature was designed to engage a step-by-step resolution process. Players have to understand that to reach the exit, it is necessary to solve the level in sub-steps (Objective 1: Activate the terminal to open the door. Objective 2: Reach the exit). This also allows players to manipulate the states of an object (open or closed).
3. Program several robots with a unique program. Players need to find a generic solution that enables two robots to reach their exit in different mazes.

In sum, according to Wing (2006), players have to observe and model the simulation (abstraction), decompose their strategy in sub-steps (decomposition), determine the best solution (evaluation), plan the actions to perform (algorithmic thinking) and reuse and adapt previous solutions on new problems (generalization).

Currently a sequence of actions could be created with action blocks: advance, rotate left/right, U-turn, activate and wait; and with control blocks: if condition and for loop. The “if” block is currently limited to one criteria selectable in a drop-down menu (is a wall in front of the robot? Is a sentinel in front of the robot? …).

SPY loads levels from XML files, in this way, it is possible to customize the pedagogical scenario by changing levels order. The levels being in XML format, it is possible to create new levels or to modify those already existing. Currently SPY doesn't include a level editor to help to create new levels or update existing ones, so editing XML files is only accessible for users who know XML format. We used this prototype to implement new features and modify existing ones to improve them.

[8] SPY: https://github.com/Mocahteam/SPY, accessed May 19, 2022

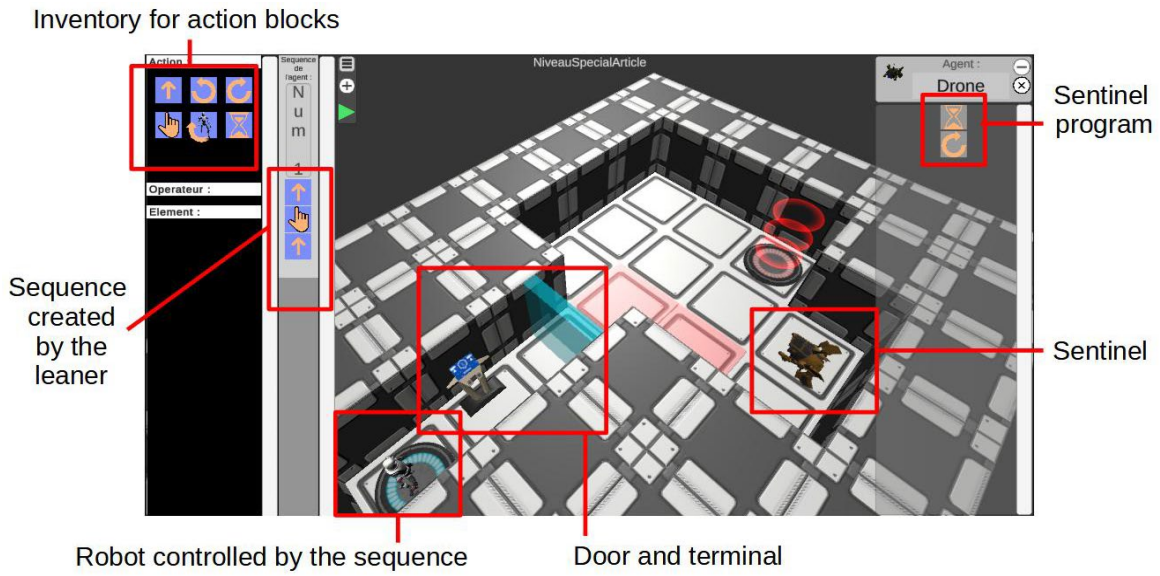

Figure 1: SPY example level

We have designed features for each of the competencies offered by PIAF. We propose 22 features for the 26 PIAF competencies. We remind the goal of this research: providing a tool for the SPY game that enables teachers to select a set of competencies on CT and automatically build a level on SPY focused on these competencies. The list of the 22 features is: F1: Enable library; F2: Enable only required actions; F3: Drag and Drop elements; F4: Initialize robot with an incomplete starting sequence; F5: Enable step by step resolution; F6: For loop; F7: If Else; F8: Play button enabled; F9: Name robot and container to associate them; F10: Select the appropriate actions to solve a level; F11: Ask question according sequence executed by the robot; F12: Select the appropriate sequence to solve a level; F13: Paint the robot and colour detector; F14: Select sequence(s) from history; F15: Provide specific sequence(s); F16: While loop; F17: Operator (And, Or, Not); F18: Select the appropriate exit; F19: Enable several exits + distance criteria; F20: Select an icon to save custom sequence actions; F21: Console mode; F22: Select wrong blocks.

Table 2: Relations between the features and the PIAF competencies

		Features																						
		F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	F17	F18	F19	F20	F21	F22
PIAF Competencies	C1.1									O													
	C1.2	O																					
	C1.3										O												
	C1.4											O											
	C1.5												O										
	C1.6													O									
	C1.7														O								
	C2.1		O	O																			
	C2.2			O	O																		
	C2.3			O																			
	C2.4			O																			
	C2.5			O												O_1							
	C2.6					O																	
	C3.1						O																
	C3.2																O						
	C3.3							O															
	C3.4																	O					
	C4.1																		O	O			
	C4.2												O							O			
	C4.3			O												O_2				O			
	C5.1																				O		
	C5.2																					O	

		Features																					
		F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	F17	F18	F19	F20	F21	F22
	C6.1								O														
	C6.2															O_3							0
	C6.3			O												O_3							
	C6.4			O											O								

O_1: Incomplete sequences O_2: Non optimal sequence O_3: Sequence with bugs

Table 2 shows a list of the established features and the PIAF competencies they correspond to. The white and grey features were already present in the first version of the game, but the grey features need to be modified to better match the competencies. The black features correspond to the proposed new features.

We can't detail all the new features we designed but we focus in some of them:

1. F9 asks the player to give a name to the robot and the container of actions in order to associate them. If a container and a robot have the same name, the actions inside this container will control the associate robot. Thus, we link this feature to the C1.1 competency (i.e., Name objects and (sequences of) actions) which is defined as "Being able to give names to objects, actions and sequences of actions".
2. F13 provides an action to paint the robot with a specific colour. The player will have to use this action to enable the robot to pass through magic doors that let enter only objects with a specific colour. We link this feature to C1.6 competency (i.e., Using objects whose value can change) that refers to the notion of variables. In our case the robot can store a property (the colour) that can be read by the door.
3. F20 allows the player to save a sequence of actions he has created. When he saves it, he has to associate an icon with it (from a selection offered by the game). The icon is then added to the list of available blocks and the player can use it to create new sequences. When the icon is read during the execution of the sequence, the robot performs the sequence of actions associated with the icon. We associate this feature to the C5.1 competency (i.e., Represent objects or sequences of actions in various formal representations).
4. F22 shows the player an erroneous sequence of actions to solve a level. Then the player has to select the erroneous blocks in the sequence. We associate this feature with C6.2 competency (i.e., Check a sequence of actions with respect to a given goal).

We can see in Table 2 that only two competencies used exactly the same features. The competencies C2.3 "Create a sequence of actions to reach a simple goal" and C2.4 "Create a sequence of actions to reach a complex goal" both represent the creation of sequences. The only difference is that C2.4 is about more complex problems. We explain this similarity in next paragraphs. As discussed in Section 3, we didn't identify any research that details in existing games how the relationship between competencies and game features was designed. Table 2 shows the features we associate to PIAF competencies in the SPY game. But we identify that these associations reveal additional constraints. Indeed, we have constraints between PIAF competencies (for instance the framework states that C2.3 is an extension of C2.1) and constraints between features (for instance "Enable library" is required to "Drag and drop element"). Table 3 summarizes these constraints and we will detail some of them.

Table 3 has to be read line by line. The symbols "||" mean: if a row competency is selected, at least one of the column competencies must be selected. The symbols "&&" mean: when a row competency is activated, all column competencies must be selected. An empty cell means: the row competency is compatible with column competency, the both competencies can be enabled at the same time. The symbols "x" means: when the row competency is selected the column competency is in conflict, the both competencies cannot be selected at the same time.

We can see that for some competencies (C1.3, C1.4, C1.5, C4.1, C4.2 and C6.2), the lines are entirely filled with "x". These competencies are not based on writing a program but require to identify an error, to select a specific answer, to analyse the result of a program, etc. To deal with these competencies we propose specific game features. For instance, C1.3 "Identify the input parameters of a sequence of actions" will be processed by presenting a situation to the player and asking him/her to select required action blocks to resolve this situation but s/he will not have to build a sequence with these blocks (this concerns the competencies C2.X). Then C1.3 will be a specific activity to select a set of blocks without using them and can't be combined with other competencies with SPY. We followed the same process for C1.4, C1.5, C4.1, C4.2 and C6.2.

As we introduce before, "x" symbol refers to incompatible competencies combinations with SPY. This incompatibility is symmetrical. This can be explained by constraints between competencies (e.g., C2.3 and C2.4). If the teacher wants to work on the creation of a sequence of actions to resolve a complex problem (i.e., C2.4), then competency C2.3 (i.e., create a sequence of actions to resolve a simple problem) is not compatible. But this can also be explained by the features. For instance, C2.5 asks to reach an objective by combining already known sequences. This competency requires to use and combine only sequences of actions and exclude features that enable to use action blocks. At the same time C2.2 (i.e., complete a sequence of actions to reach a simple objective) asks to add missing actions in a given sequence without using already known sequences. These two features (i.e., use only sequences and use only action blocks) are mutually exclusive.

The symbols "&&" and "||" mostly work symmetrically. For instance, C6.1 requires at least one competency between C1.7, C2.1, C2.2, C2.3, C2.4, C2.5, C4.3, C5.2, C6.3 and C6.4 that are marked with "||". This is because C6.1 (i.e. check if a sequence of actions reaches an objective) is integrated into SPY by activating the run button that enables to play and check a sequence of actions. This competency cannot be worked on alone. It is necessary to combine it with a competency that requires code execution, C2.2 for instance (i.e. complete a sequence of actions to reach a simple objective). In consequence working on C2.2 requires being able to check if a sequence of actions reaches an objective and then requires C6.1 that is coded in Table 3 by “&&”.

We identify some cases where “&&” and “||” are not symmetrical. For instance, C3.4 (that enables the use of AND, OR, NOT operators) requires at least C3.2 (i.e., “while” loop) or C3.3 (i.e., “if” block), but “&&” symmetry is not true. Indeed, it is possible to work on C3.2 or C3.3 without working on C3.4.

Table 3: Relationships between competencies depending on underlying game features in SPY

	C 1.1	C 1.2	C 1.3	C 1.4	C 1.5	C 1.6	C 1.7	C 2.1	C 2.2	C 2.3	C 2.4	C 2.5	C 2.6	C 3.1	C 3.2	C 3.3	C 3.4	C 4.1	C 4.2	C 4.3	C 5.1	C 5.2	C 6.1	C 6.2	C 6.3	C 6.4
C1.1			X	X	X													X	X				&&	X		
C1.2			X	X	X			||	||	||	||	||		||	||	||	||	X	X	||	||			X	||	||
C1.3	X	X		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
C1.4	X	X	X		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
C1.5	X	X	X	X		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
C1.6			X	X	X					X	&&							X	X					X		
C1.7			X	X	X													X	X					X		
C2.1		&&	X	X	X					X	X	X						X	X				&&	X		
C2.2		&&	X	X	X							X						X	X				&&	X		
C2.3		&&	X	X	X	X		X			X	X		X	X	X	X	X	X				&&	X		
C2.4		&&	X	X	X	||		X		X		X		||	||	||	||	X	X				&&	X		
C2.5		&&	X	X	X			X	X	X	X							X	X				&&	X		
C2.6			X	X	X													X	X				&&	X		
C3.1		&&	X	X	X					X	&&							X	X					X		
C3.2		&&	X	X	X					X	&&							X	X					X		
C3.3		&&	X	X	X					X	&&							X	X					X		
C3.4		&&	X	X	X					X	&&				||	||		X	X					X		
C4.1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		X	X	X	X	X	X	X	X
C4.2	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		X	X	X	X	X	X	X
C4.3		&&	X	X	X													X	X				&&	X		
C5.1		&&	X	X	X													X	X					X		
C5.2			X	X	X													X	X				&&	X		
C6.1			X	X	X		||	||	||	||	||	||						X	X	||		||		X	||	||
C6.2	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		X	X
C6.3		&&	X	X	X													X	X				&&	X		X
C6.4		&&	X	X	X													X	X				&&	X	X	

5. Conclusion and future perspective

In this paper we analysed three learning games to identify the implementation of PIAF competencies in each game. We found that not all competencies were present. But we were able to see that the ways in which these competencies were addressed were diverse. None of them, however, explain the links between the competencies targeted and the associated game mechanics.

Our contribution is to propose a set of features for a specific learning game, each feature is linked to one or several PIAF Framework competencies. In this way, we have identified different types of constraints related to the combination of competencies and features. These constraints have an influence on the pedagogical scenario that teachers can create. Some of the competencies they choose prevent them from selecting others at the same time, and some of the competencies they choose force them to also select other competencies at the same time.

We followed a didactic engineering research methodology. We focused in this paper on the two first stages of didactic engineering: the first is a preliminary analysis (the analysis of learning game mentioned above in view of PIAF competencies) and the second is the design and a priori analysis (proposition of a gameplay feature for each PIAF competency). Future works will focus on phases 3 (observation and data collection) and 4 (a posteriori analysis) and will engage the iterative process of didactic engineering research methodology.

Concerning the two first stages (preliminary analysis and design) we identified two main challenges. The first is a classic game design subject and consists of finding the right game concepts associated with each competency. The second is to create game mechanics that are the most independent from each other, but can be used together.

References

Atlan, C., Archambault, J. P., Banus, O., Bardeau, F., Blandeau, A., Cois, A., Courbin-Coulaud, M., Giraudon, G., Lefèvre, S.-C., Letard, V., Masse, B., Masseglia, F., Ninassi, B., de Quatrebarbes, S., Romero, M., Roy, D. and Viéville, T. (2019), "Apprentissage de la pensée informatique : de la formation des enseignant· e· s à la formation de tou· te· s les citoyen· ne· s", *Revue de l'EPI (Enseignement Public et Informatique)*.

Baron, G-L., Drot-Delange, B. (2016) "L'informatique comme objet d'enseignement à l'école primaire française ? Mise en perspective historique", *Revue française de pédagogie Recherches en éducation*, pp 51–62.

Barquero, B. and Bosch, M. (2015), "Didactic engineering as a research methodology: From fundamental situations to study and research paths", *In Task design in mathematics education*, Springer, Cham, pp 249-272.

Bartram, D. (2006), *The SHL universal competency framework*, SHL White paper, pp 1-8.

Dowek, G. (2011) "Les quatre concepts de l'informatique, Sciences et technologies de l'information et de la communication en milieu éducatif : Analyse de pratiques et enjeux didactiques", *New Technologies Editions*, Athens, pp 21-29.

Kradolfer, S., Dubois, S., Riedo, F., Mondada, F. and Fassa, F. (2014), "A sociological contribution to understanding the use of robots in schools: the thymio robot", *International Conference on Social Robotics*, Springer, Cham, pp 217-228.

Lindberg, R. S., Laine, T. H. and Haaranen, L. (2019), "Gamifying programming education in K-12: A review of programming curricula in seven countries and programming games", *British Journal of Educational Technology*, Vol 50, No. 4, pp 1979-1995.

Miljanovic, M.A., Bradbury, J.S. (2018), "A Review of Serious Games for Programming", *In Joint International Conference on Serious Games*, pp 204-216.

Parmentier, Y., Reuter, R., Higuet, S., Kataja, L., Kreis, Y., Duflot-Kremer, M., Laduron, C., Meyers, C., Busana, G., Weinberger, A. and Denis, B. (2020), "PIAF: developing computational and algorithmic thinking in fundamental education", *EdMedia+ Innovate Learning*, Association for the Advancement of Computing in Education (AACE), pp 315-322.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B. and Kafai, Y. (2009), "Scratch: programming for all", *Communications of the ACM*, Vol 52, No. 11, pp 60-67.

Saddoug, H., Rahimian, A., Marne, B., Muratet, M., Sehaba, K. and Jolivet, S. (2022), "Review of the Adaptability of a Set of Learning Games Meant for Teaching Computational Thinking or Programming in France", *Proceedings of the 14th International Conference on Computer Supported Education*, Vol 1: GonCPL, pp 562-569.

Vahldick, A., Mendes, A. J. and Marcelino, M. J. (2014), "A review of games designed to improve introductory computer programming competencies", *2014 IEEE Frontiers in Education Conference (FIE) Proceedings*, pp 1-7, doi: 10.1109/FIE.2014.7044114.

Van den Brande, L. (2016) *The European Digital Competence Framework for citizens*. 10.13140/RG.2.1.4687.1281.

Wing, J. M. (2006) "Computational thinking", *Communications of the ACM*, Vol. 49, No. 3, pp 33-35.