Design, Complexity, and Coding: A Framework to Evaluate Games

Bernadette Spieler[1] and Ferenc Kemeny[2]
[1]University of Hildesheim, Institute for Mathematics and Applied Informatics, Germany
[2]Institute for Psychology, University of Graz, Austria
bernadette.spieler@uni-hildesheim.de
ferenc.kemeny@uni-graz.at
DOI: 10.34190/GBL.20.156

Abstract: Game development-based learning and creative coding activities are seen as an opportunity for young people to acquire programming skills and overall, Computational Thinking (CT) skills in an entertaining way. The goal of such activities is to create playable gaming artefacts. While learning to code is becoming more and more important, young game designers miss a clear map to know how game and design elements can be organized and in which situations, specific elements or structures are appropriate. The aim of this paper is to present a valid assessment for structuring games created with visual coding languages. In different workshop settings (out-off school workshops, school workshops) 84 participants between 10 and 15 years old have individually created gaming applications with our Catrobat environment. Games created during these workshops (2017-2019) were analysed and structured in game elements, design elements, and game complexity. This method is based on various existing frameworks, for example, the Mechanics, Dynamics, and Aesthetics (MDA) framework, which provide a consistent structure to define game elements, goals, and rules, thereby delivering a common framework and vocabulary for games. The transfer of game concepts in a consistent structure should support the overall clarity of the game and should help to focus students on the relevant elements of games. We argue that students need guidelines to be able to implement more complex games even with visual programming languages. The result, the Playing, Engagement, Creativity, and Creating (PECC) Assessment Template should support the process of evaluating gaming apps in visual coding languages, can be used by researchers as a valid method for structuring games, helps all students to adhere to good game design principles, and teachers can use the analysis to evaluate games created with visual programming languages.

Keywords: game design, mobile gaming, mobile learning, game design patterns, game analysis

1. Introduction

The digital society places an ever growing demand of Computational Thinking (CT) and programming education (CECE, 2016), which are often referred to as the literacy of the 21st century on par with reading and mathematics (Wing, 2006). This movement goes well beyond the idea that "we need more software developers". Tracking changes in the CT development of children is difficult, though. One approach is to evaluate their end products – which are often games (Kafai and Burke, 2013). There are numerous ways to analyse games, e.g., with the help of players' reports, walkthroughs, discussions, observing others play, interviews of players, game documentation, playtesting reports, or interview game developers (Arseth, 2003). However, no analysis considers specifically game and design considerations regarding the elements used (e.g., structure of a game, game mechanics, dynamics, or aesthetic) in addition to the complexity of the created games. The authors of this paper argue that it is fairly difficult in different workshops settings (with often limited amounts of time) to create self-contained and playable games that are considered "good" by others. Teachers and trainers can support students during the game design process by sticking to a framework that supports the game design process, for example, the Mechanics, Dynamics, and Aesthetics (MDA) framework (Hunicke, Leblanc, and Zubek, 2004) or other techniques, like the use of storyboards (Co, 2006), and "ceremonies" (Penn, 2005). In this paper we present a valid assessment: 1) for teachers to give them a way to analyse games programmed by students, and 2) for students to stick to these concepts. Therefore, we made use of our Playing, Engagement, Creativity, and Creating (PECC) framework (Spieler and Girvan, 2020). We mainly aim to test the following hypothesis: "The creation of simple games in a visual programming environment can be guided and assessed by providing clear structures via storyboards and assessment templates".

The structure of the paper is the following: First, relevant concepts and game design strategies are presented from a theoretical point of view. Based on this theoretical framework, the authors describe in Section 3 the Methodology and the used materials (storyboards, assessment template), followed by Findings in Section 5. Section 6 presents the discussion in relation to assessment template. The last section concludes this paper and describes the author's future work.

2. Theoretical Framework

Games are likely to increase students' motivation for programming and allow them to develop their programming knowledge (Kafai and Vasudevan, 2015). More than that, such Playing, Engagement, Creativity, and Creating activities can influence the so-called "Generation Z" and beyond to be more interested in Computer Science careers (Spieler et al., 2020). However, not all students are used to playing games, and not all are familiar with common game elements and structures. Fortunately, several tools for programming beginners make game creation easier. These tools include visual programming environments, like Scratch (https://scratch.mit.edu), Alice (https://alice.org), or Catrobat (https://catrobat.org).

2.1 Computational Thinking (CT) Skills

Based on Papert (1985), Wing (2006) re-introduced the term "Computational Thinking" (CT) skills. Wing's idea that children who are introduced to CS learn more than just programming opened a new way of thinking, e.g., it showed the benefits of learning to think like a programmer (Wing, 2008). Wing argues that learning how to code reinforces CT skills but it is not just about programming. Moreover, students should think first about possible solutions to a given problem (use of problem solving skills) and second, implement their ideas by using a computing device (use of programming skills). In order to successfully implement their own solutions, students have to apply different programming concepts, such as loops and conditions, as well as practices, such as abstraction and debugging (Kafai and Burke, 2013).

2.2 Visual Block-Based Programming & Analytical Tools

In the last decade, a number of block-based visual programming tools, e.g., the Scratch environment (Resnik et al., 2009), or the Pocket Code App (Slany et al., 2018). These allow the creation of games, stories, animations, and many types of other apps directly on phones or tablets, thereby teaching fundamental programming skills. The open source software "Dr. Scratch" is an analytical tool evaluating Scratch projects in a variety of CT areas (Moreno-León, Robles and Román-González, 2015). It fosters the idea that students need to apply different programming concepts, such as loops and conditions, as well as CT practices, like abstraction and debugging in order to implement their own programming solutions (Lye and Koh, 2014). Despite being a helpful academic tool, "Dr.Scratch" has two remarkable weaknesses. First, design and gaming elements are not considered, meaning that this analysis says nothing about the quality of the game and if it is playable or not. Second, it focuses only on the complexity of the projects, instead of designing games. Consequently, students receive more points, by using more parallel scripts or broadcast messages. It would be beneficial to focus on "diversifiers" as they are used in Game Jams, define the scope of the game and provide small additional sub-goals to aim for (Eberhardt, 2016). With such sub-goals, students can be motivated to use important CT concepts but not naming them explicitly.

2.3 Structuring a Game-based Learning (GBL) Approach

The key aspect of a Game-based Learning (GBL) approach is an inherent constructionist approach (Kazimoglu et. al, 2010). Regarding coding, the constructionist theory states that it is much more effective when students program personalised games instead of just learning about computing (Kafai, 2006). This "learning through the exercise of designing" as popular GBL approach (Berland and Lee, 2012) enhances design thinking, creativity, and arithmetic skills. However, according to Sung et al. (2010) game creation is significantly challenging without considerable knowledge in computer graphics and a background in playing or designing games. Advanced game development units require considerable preparation time and material content.

Most characteristics are very similar in all games. For instance, rules, goals, variables, and uncertain outcomes (Seaborn and Fels, 2015). Games are formed from a variety of components and it is the players' perception, which determines whether the experience is fun and entertaining or not. Considering this, students have to take the role of a game designer and have to learn to "think as a game designer" (Arseth, 2003). At an early stage they have to think of the gaming world in general (the genre of a game, the theme, level design), and the game play (the story, main and side characters, their actions, strategies and motives).

The structure of a game (rules, goals) can be defined with the help of the Mechanics, Dynamics, and Aesthetics (MDA) frameworks. The MDA framework describes the separation of game components (Hunicke, 2004). Mechanics refer to game rules and logic (e.g., points, level, challenges/missions, virtual goods, leader boards), Dynamics to the results of mechanics during gameplay (rewards, status, achievements, competition, visual/sound feedback), and Aesthetics is the "look & feel" of the game and can be reinforced by the game's art, sound, and other clarifying elements (e.g., narrative, sensation, discovery). The MDA provides a useful lens for thinking about the various elements that comprise gameplay.

Game genres help to define what game design elements (MDA) are necessary to effectively create the chosen genre and to help the theme 'fit' into the genre classification. Examples for genres are: action, adventure, puzzle, simulation, or strategy (Lee et al., 2014). A theme, on the other hand, can be used to describe different aspects of a game. It can refer to a specific colour, story, or narrative (Brathwaite and Schreiber, 2009). While the theme can change during the game (e.g., with different levels), the genre usually remains the same (Rollings and Adams, 2003). The level of control or interactivity of the characters does not only depend on the genre but also on the technology. Characters in PC games are commonly controlled via the keyboard or the mouse, whereas in mobile games, finger positions or sensors are used. With some exceptions, most games have a goal or common objectives which represent one core concept of game design (Shi and Shih, 2015). The designer must think about what kind of experience to provide to the player. The goal of the game is some kind of victory and sometimes referred to as missions or quests (Brathwaite and Schreiber, 2009). Goals typically provide rewards, e.g., to level-up or the option to buy more advanced equipment. To transfer the concept of games in a consistent structure, several strategies are possible. For the European NOLB project, the team used the term "ceremony" and "Shape of a Game". In an article for EDGE Magazine, Gary Penn describes "Ceremonies are incentives to play and sustain player interest. Ceremonies frame and punctuate play, principally the start and end of play." (Penn, 2005, p. 92). According to Co (2006) a storyboard can help to define the gameplay, the design of the game, the story, and to provide an overall picture of the whole gaming world.

3. Methodology & Activity Design

3.1 Participants

For the current study, data of 84 children (64 girls and 20 boys, aged 11-16) was analysed. Students participated in one of five different workshops, Table 1.

Table 1: Participants of the different coding workshops

	Setting	**# children**
Workshop 1	mixed workshop (summer)	7 girls, 13 boys
Workshop 2	girls' coding week (summer)	28 girls
Workshop 3	girls' coding week (summer)	13 girls
Workshop 4	mixed in-school course	6 girls, 5 boys
Workshop 5	girls-only school course	10 girls
		64 girls, 18 boys

Each of the three summer workshops ran for four days. Participants were introduced to the Pocket Code app. During the first three days, ten different topics were covered through tutorials and hands-on programming experience. These topics included the most important concepts, like loops, logical operators, conditions, etc. Each topic ended with a challenge. The second half of day 3 was devoted to children designing their own final game. Before they started with their own game, all participants had to fill out two storyboards (see Section 3.2). In addition, students were introduced to important game and design principles. This information is summarized in video: https://www.youtube.com/watch?v=eHnAMKUtq14

The school-based extracurricular workshop covered the same topics as the summer workshops, except that instead of one intensive coding week they had coding lectures over a time period of four weeks, two hours each week. They had to solve a tutorial during two school lessons (45 minutes each) which covered the important concepts. In the second week all students started their own game. Also in this setting students received both storyboards and information on game design.

3.2 Materials: Storyboard and PECC Assessment Template

The PECC Assessment Template is part of the PECC Framework (Spieler and Girvan, 2020). Originally developed as the Game-Making Teaching Framework (GMTF) for middle and high school children during the H2020 project "No One Left Behind" (NOLB, funded by the European Commission [645215]), and it has developed based on emerging data (see Spieler and Slany, 2018a; Spieler and Slany, 2018b). The NOLB GMTF provided a frame for the NOLB project to understand and apply game mechanics and dynamics as well as to integrate the app and other tools and services into the academic curricula. Later, this model has been expanded to have a unique focus on gender consciousness by focusing on playing, engagement, creativity and creation in CS classrooms. For the assessment of the games the framework has been extended with the PECC Assessment Template. In addition to concepts described in Section 2.3, formal elements designated by Lankoski and Björk (2015) (e.g., the structure of a game, game play, or media/sound assets) are central to this assessment template. In addition, the assessment considers design elements, game elements, and game complexity, see Section 2.3. The PECC Assessment Template is available with the QR-Code/Link provided in Table 2.

Table 2: QR Code and Link to the PECC Assessment Template

QR Code:	**Link:**
	https://catrob.at/pecc

The storyboards displayed within this section support students with a template to stick to gaming concepts and are part of the PECC Assessment Template. Students received a graphical and a textual empty scheme. Table 3 shows an example of a textual storyboard as it was used in all of our coding workshops and Figure 1 illustrates an example of a graphical storyboard. The later should help students to stick to the "Shape of the Game" and at least to think of a title, introduction and end screen of their games.

Table 3: Example for a textual storyboard (available for download)

Name of the game:	Pokèmon care
Main character:	a Pokèmon (you can give it a name)
Genre/Theme:	simulation, minigames / fantasy
Goal:	evaluation, take care until it dies
Mechanics: (levels, increase in difficulty)	level 3: first evaluation, level 4: it gets babys, level 6: it dies
Dynamics: (points, life, countdown, virtual goods, choose a player)	points, virtual goods
Aesthetics: (visual/sound feedback, sensation, narrative)	sounds, narrative, sensation
Level of control: simple: tap, glide medium: sensors, buttons expert: clones, physics	inclination, touch sensor, buttons, animation, clones
Sub-goals:	• Use of two different sensors • Integrate a collision detection • Use bricks from five different categories • Objects should communicate with each other

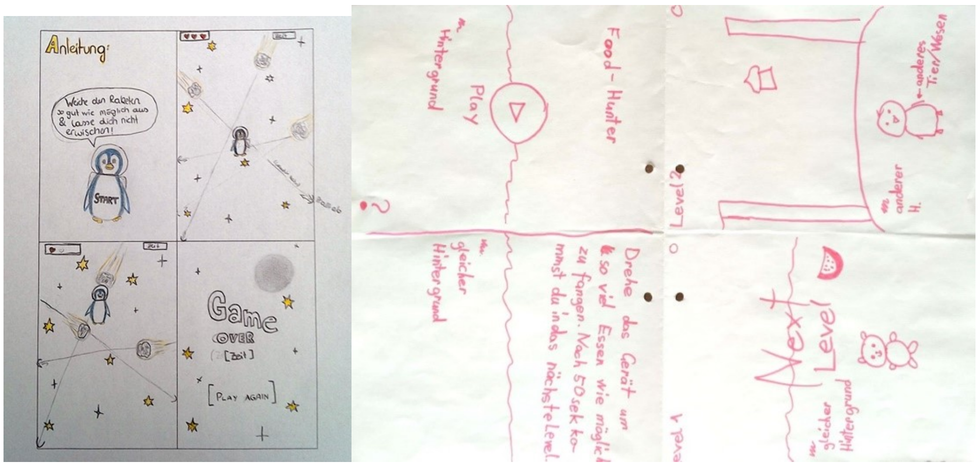
Anleitung:
Wache den Raketen so gut wie möglich aus & lasse dich nicht erwischen!
START
Game OVER
PLAY AGAIN
Food-Hunter
Play
Hintergrund
Level 2
Drehe das Gerät um so viel Essen wie möglich zu fangen. Nach 50 sek kommst du in das nächste Level.
gleicher Hintergrund
anderes Tier/Wesen
anderer H.
Next Level

Figure 1: An example of two graphical storyboards with different scenes for title, introduction, and game over/end screen.

The game and design elements were analysed manually during the students' presentation and compared with the pre-defined elements within their storyboards.

4. Findings

In this section we present a summary of the artifacts created by students. Links made to the PECC Assessment Template are highlighted in **bold**. At the end the whole analysis is illustrated in the form of a table. For every applied concept students received points.

In the first step, we analysed the **Design Elements** used within the games. Design elements refer to 1) the **"Shape of the Game"** and 2) **Visual/Sound Design**.

"Shape of the Game": With the help of the graphical storyboard, children draw a rough sketch of the title, introduction, and end screen of their game. In Pocket Code, users can create separate scenes for different levels. Every scene has their own background object and figures. The "Start scene"-brick is an easy option to switch between the scenes. This feature leverages the steps to integrate all necessary screens as separate scenes, see Figure 2.

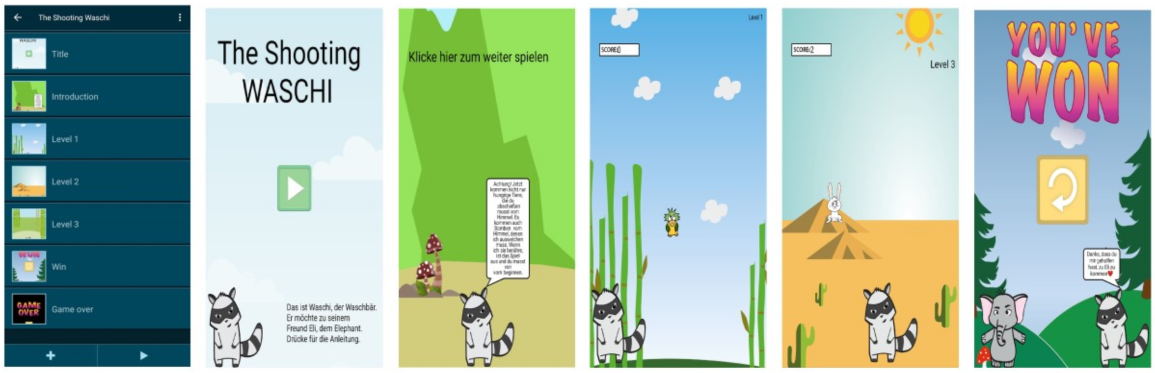
The Shooting Waschi
Title
Introduction
Level 1
Level 2
Level 3
Win
Game over
The Shooting WASCHI
Das ist Waschi, der Waschbär. Er möchte zu seinem Freund Eli, dem Elephant. Drücke für die Anleitung.
Klicke hier zum weiter spielen
Level 1
Level 3
YOU'VE WON

Figure 2: a) Different scenes within Pocket Code b) title screen, c) introduction screen, d+e) level 1 + 3, f) end screen

In total, 73.17% used all screens of the "Shape of the Game". Only three of the students did not include any of the screens necessary, and 5 programs did not include an end or win screen, 6 of the games did not use an introduction screen, and in 4 of the games more than two scenes were missing. Students received 3 points if they included all of the three scenes.

Visual and Sound Design: Students could choose from different sources. For example, they could choose to draw their graphics on their own (i.e., integrate their own artwork), draw pictures in the paint tool of Pocket Code, to edit existing graphics, or to search for graphics on the internet or in the Catrobat media library. The options for sounds include to record own sound or to use the Catrobat sound library. This decision is part of the graphical storyboard and students were encouraged to use different sources for their assets to spark creativity. Figure 3 shows different sources used in different games.

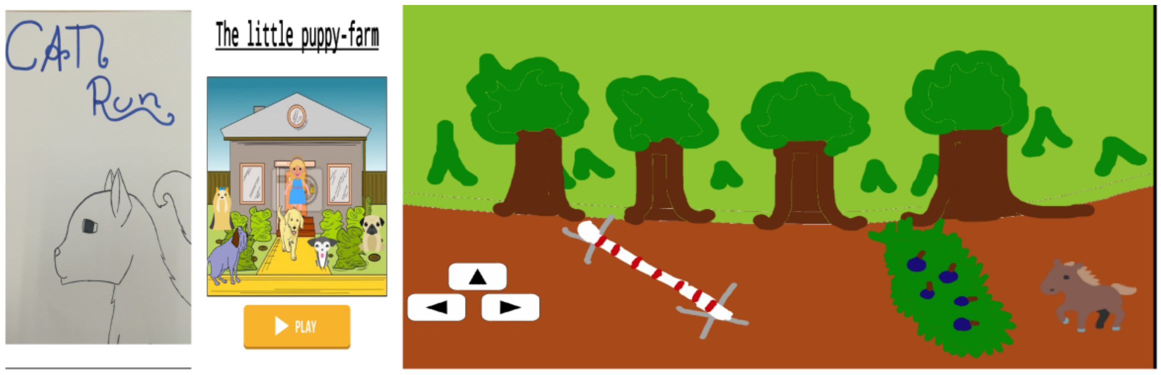

Figure 3: a) handmade artwork, b) a mix of internet graphics and the Catrobat media library, c) paint tool

On average, 2.41 different sources for assets were used through all games. Most programs used graphics provided by the Catrobat media library (85.88% of the programs), taken from the internet (55.29% of the programs), or drawn via the paint tool of Pocket Code (57.65% of the programs). Only 16.47% of the games used their own assets (handmade artwork). A percentage of 27.06% of the programs included sounds (100% of them were taken from the sound library). Students received three points if they used at least one self-made graphic (either handmade artwork, edited/created via Pocket), one prepared graphic (media lib, internet) and a sound.

In a second step, we analysed the **Game Elements** used within the games. Game elements defined by the 1) **Level of Control** (interaction) and 2) used **MDAs**.

Level of Control: This is part of the textual storyboard. Different concepts exist to move objects. Children can use **simple concept**s like animations (objects glide or move when programs start) and "When tapped" bricks (similar to the "When clicked"-brick in Scratch). Defined **medium concepts** are the use of sensors, for example the inclinations sensor of the phone or buttons (i.e. use of messages). The use of clones and physics bricks (e.g., velocity, gravity) count as **expert concepts**. Students were told to use different concepts. For simple concepts they got 1 point, for medium concepts 2, and for advanced they received 3 points.

Students used on average 1.78 different concepts for moving objects. Most of them used the simple concept of tap events (46.99% of the projects), or the medium concept inclination sensor (38.55% of the projects). The third most used concept was animations by moving and gliding objects (30.12% of the projects) and the use of buttons (28.92% of the projects). A percentage of 31.33% of the projects used advanced concepts like clones or physics bricks to move their objects.

MDA: Within the storyboards students should already decide which kind of MDAs they will use for their games. For **Mechanics** they should integrate level and increase in difficulty and for **Dynamics**, life, countdown, virtual goods, or choose a player. **Aesthetics** elements are sound and visual feedback as well as narrative and sensation. Narrative means that students integrate a storyline through the game and sensation if something unexpected is integrated. Examples that are mentioned were vibration effect if something happens, animations within the intro etc. Students were told to integrate at least one of each category, thus they could get three points in total for the MDA part.

On average the students used 2.84 different MDAs. A percentage of 30.12% used elements of all three types (Mechanics, Dynamics, and Aesthetics elements). Most games used points (68.67% of the games) and levels (60.24% of the games). Dynamic elements were used the most often (in 75.90% of the games).

Finally, the **game complexity** was analysed. Game complexity considered the **code statistic** and the **brick statistic**. Both are displayed at the Catrobat Sharing page in the project overview. The code statistic provides the number of scenes, scripts, objects, looks, sounds, and variables. The brick statistics show the total amount of bricks per category (bricks are of different categories: event, control, motion, look, sound, pen, data) and the amount of different bricks in each category. For the evaluation, the average value of each workshop has been calculated and compared if students are above (+1 points) or below the average (0 points). This makes it possible to evaluate projects of different workshops or tasks. However, in this way workshops cannot be compared with other workshops. Further, we provide a decomposition of the programs and show if and how often **programming concepts** (e.g., broadcast messages, object properties, functions) were used. These were defined with the storyboards **diversifiers**.

In total, all projects together used 450 scenes (5.2 on average), 4,388 scripts (45 on average), 13,892 bricks (164.8 on average), 2,338 objects (26.8 on average), 2,606 looks (29.8 on average), 115 sounds (1.2 on average) and 508 variables (8.4 on average). All projects used on average 4.8 different event bricks, 5 different control bricks, 5 different motion bricks, only 1 sound brick, 4 different look bricks, and 2 different data bricks.

For the analysis of the complexity of the games, only the applied programming concepts were counted. Thus students could get 3 points at most. A percentage of 27.38% got all 3 points, 7.14% did not apply any concept (on average 1.90). In the future we plan to include more CT concepts, see outlook section.

With these different components, the games received a **total score**. The maximum score reached was 18, the minimal score reached was 6. On average projects received a score of 11.16 points. Table 4 summarizes the results.

Table 4: Summary of each workshop that used textual/graphical storyboards

	# children	**Design Elements**	**Game Elements**	**Complexity**	**Total Score**
Workshop 1: mixed summer workshop	7 girls, 13 boys	73.33%	60%	76.67%	**67.22%**
Workshop 2: girls' coding week	28 girls	76.19%	59.13%	58.33%	**64.68%**
Workshop 3: girls' coding week	13 girls	78.21%	50.43%	58.97%	**61.11%**
Workshop 4: mixed in-school course	6 girls, 5 boys	71.21%	36.36%	69.70%	**53.54%**
Workshop 5: girls-only school course	10 girls	75%	37.78%	50%	**54.44%**
	64 girls, 18 boys	**75% (SD = 1.36, min: 2, max = 9)**	**52.30% (SD = 2.75, min = 2, max = 13)**	**63.41% (SD = 3.29, min = 2, max = 16)**	**67.55% (SD = 5.61, min = 8, max = 36)**

5. Discussion & Conclusion

As explained above, the aim of the study is to present a valid assessment for structuring games created with visual coding languages. The novelty of the analysis relies in the fact that we did not only focus on MDAs, as previous assessments (Spieler and Slany, 2018b). We are aimed at describing the games at different levels. These levels vary from the very low level of brick statistics to a high level of conceptual and design elements. A unified framework to assess these games can serve as a measure of the output of coding courses, and in turn as a measure for coding competence. The proposed assessment template proved to be successful on these grounds: the projects showed a considerable amount of variance in design elements, game elements as well as in complexity (see Table 4). Such a multifaceted assessment template allows the exploration of how different individual and situational factors may contribute to coding success. Some pilot data is already available: we have successfully shown that design elements are associated to simple memory storage functions and basic mathematical abilities, whereas game elements relied on the creativity of children (Spieler et al., 2020).

Whereas the above cited pilot study provided empirical evidence to the multifaceted nature of computational thinking, we still need a larger study to validate our assessment template. While the design of the framework is based on previous theoretical works (Spieler and Girvan, 2020), and the pilot results are aligned to these theoretical assumptions, we have to provide further evidence from more participants to explore the relationship between design elements, game elements and complexity. Reliability of the measures is also a crucial issue. Test-retest reliability measures are still required to guarantee that the assessed measures in fact reflect the underlying concepts. The current study, on the other hand, provide indirect evidence in this direction: we have tested five different groups, and the descriptive statistics within these results are rather similar.

In the current study we have found a wide range across the analysed domains. This is in accordance with our aims in developing a tool to assess and evaluate projects. The use of such a tool CS education is beneficial, as teachers can track the development of their students, and can compare the performance of individuals on three different scales. At the same time, the principles of assessment can serve as a guideline for students, who can design their apps to conform assessment criteria.

6. Outlook

In our future work we plan to extend the scope of the current assessment. We plan to add a more detailed decomposition of projects and show if and how programming concepts are used. We also plan to include concepts of parallelism and logical thinking (among others). We are also aimed at assessing the validity and reliability of different types of project evaluation. The basic principle of "Dr.Scratch" is how often the concepts are applied, we analyse the variety of concept use instead. Our aim is to find out whether Dr Scratch-type of token-based analysis, or our category-based analysis characterizes computational skills better.

On the other hand, we have observed a great variance along the evaluation of the projects. This fits our aim, as we wanted to develop an evaluation that can differentiate between students. It is not clear, however, what predicts programming performance. In the next step, we are aimed at identifying how different cognitive and psychological factors predict coding performance. More generally speaking, the key question is what makes a good programmer. This is a complex question, and the basic prerequisite is an evaluation of programming ability. With the current assessment we are one step closer to our general aim.

References

Arseth, E. (2003) "Playing Research: Methodological approaches to game analysis". In Papers from spilforskning.dk Conference, pp 1–8.

Berland M and V. R. Lee (2011) "Collaborative Strategic Board Games as a Site for Distributed Computational Thinking", International Journal of Game-Based Learning (IJGBL), Vol 1, No. 2, pp. 65-81.

Brathwaite, B. and Schreiber, I. (2009) "Challenges for Game Designers", Course Technology PTR: Stacy L.

Committee on European Computing Education (CECE): Informatics Education in Europe: Are We All In The Same Boat? Report by The Committee on European Computing Education. Jointly established by Informatics Europe & ACM Europe, https://www.informatics-europe.org/component/phocadownload/category/10-reports.html?download=60:cece-report. Last accessed 16 May 2020

Co, P. (2006) "Level Design for Games, Creating Compelling Game Experiences", Berkeley, New Riders Games.

Eberhardt, R. (2016) "No One Way to Jam: Game Jams for Creativity, Learning, Entertainment, and Research", In Proceedings of the International Conference on Game Jams, Hackathons, and Game Creation Events, pp 34-37.

Hunicke, R., Leblanc, M. and Zubek, R. (2004) "MDA: A Formal Approach to Game Design and Game Research", Proceedings of the AAAI Workshop on Challenges in Game AI, Vol 4.

Lankoski, P. and Björk, S. (2015) "Game Research Methods. An Overview", ETC Press 2015.

Lee, J. H. et al. (2014) "Facet Analysis of Video Game Genres", Proceedings of the iConference 2014, pp 125-139.

Lye, S. and Koh, J. (2014) "Review on Teaching and Learning of Computational Thinking through Programming: What is next for K-12?", Computers in Human Behavior, Vol 41, pp 51–61.

Kafai, Y. and Vasudevan, V. (2015) "Hi-Lo tech games: crafting, coding and collaboration of augmented board games by high school youth", Proceedings of the 14th International Conference on Interaction Design and Children, pp 130–139.

Kafai, Y. and Burke, Q. (2013) "Computer programming goes back to school". In Phi Delta Kappan, Vol 95, No 1, p 61.

Kazimoglu, C. et al. (2010) "Developing a Game Model for Computational Thinking and Learning Traditional Programming through Game-Play" J. Sanchez and K. Zhang, (eds.), World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education, pp 1378-1386.

Moreno-Leon, J., Robles, G. and Roman-Gonzalez, M. (2016) "Examining the Relationship between Socialization and Improved Software Development Skills in the Scratch Code Learning Environment", Journal of Universal Computer Science, Vol 22, No 12, pp.1533-1557.

Penn, G. (2005) "Mightier than the Sword; Developed Down Under - Why the Videogaming World wants a Piece of Australia", Retrieved February 6, 2020 from: https://archive.org/stream/Edge-AUS-04-2005-01#page/n0/mode/2up

Rollings, A. and Adams, E. (2003) "On Game Design", Indianapolis: New Riders.

Seaborn, K. and Fels, D. (2015) "Gamification in Theory and Action: A Survey", International Journal of Human-Computer Study, Vol 74, pp 14–31.

Papert, S. (1985) "Mindstorms. Children, Computer, and Powerful Ideas." In Basic Books Inc. (1985).

Shi, Y. R. and Shih, J. L. (2015) "Game Factors and Game-based Learning Design Model", International Journal of Computer Games Technology, Vol 11, No 1.

Slany, W., Luhana, K., Müller, M., Schindler, C. and Spieler B. (2018) "Rock Bottom, the World, the Sky: Catrobat, an Extremely Large-scale and Long-term Visual Coding Project Relying Purely on Smartphones", In Proceedings of Constructionism 2018.

Spieler, B., and Girvan, C. (2020, in press). Das PECC-Framework: Gender-Sensibilität und spielerische Programmierung in der informatischen Grundbildung. 18. Fachtagung Bildungstechnologien der GI Fachgruppe Bildungstechnologien (DELFI 2020). September 14-18, 2020.

Spieler, B. and Slany, W. (2018a) "Female Teenagers and Coding: Create Gender Sensitive and Creative Learning Environments", Proceedings of Constructionism 2018, pp 625–636.

Spieler, B. and Slany, W. (2018b) "Game Development-Based Learning Experience: Gender Differences in Game Design", 12th European Conference on Games Based Learning, pp 616–625.

Sung, K. et al. (2010) "Game-Themed Programming Assignment Modules: A Pathway for Gradual Integration of Gaming Context into Existing Introductory Programming Courses", IEEE Education Society, Vol 54, No 3, pp 416-427.

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