GameCourse - The Next Level

CATARINA GONÇALVES, Instituto Superior Técnico, Portugal

Gamification can be defined as the use of game elements and game design elements in non-game contexts. Throughout the years, this concept became increasingly popular due to its benefits when it comes to motivating students to be active in a course. The Multimedia Content Production (MCP) course at Instituto Superior Técnico uses GameCourse, a platform that aims to motivate students through gamification. The current system has a variety of game-like components that make the experience more engaging, encouraging students to learn the course content. To award these game elements, the system relies on rules that specify what are the requirements to give a certain award. However, the system had some performance issues related to the rules that needed to be dealt with and new game elements that could be integrated. The purpose of this thesis is to improve the system's performance, and increase game element diversity in the system. Three new modules were created: Streaks, Teams and Virtual Currency. This also included creating a new type of leaderboard, suitable for teams. From the results obtained from user testing, we concluded that we were successful in implementing and integrating the new modules. Additionally, we were able to find relevant bugs and collect feedback from the students enrolled in MPC 2021/2022, which rated these new game elements to be positive for their learning experience. Finally, the results of the performance tests made show that we improved the the system's performance.

Additional Key Words and Phrases: Gamification; Education; Learning; Tailored Gamification; Player Profiling; Game Elements.

1 INTRODUCTION

Education has an enormous and powerful impact on one's life. However, as important as it may be, students' lack of interest and motivation is still a big and ongoing concern. To improve students' engagement and performance and, consequently, overcome this struggle, teachers have been trying to adapt their teaching methods. With this need, the concept of Gamification emerged and became increasingly popular throughout the years.

Gamification can be defined as "the use of game elements and game design elements in non-game contexts" [Deterding et al. 2011]. Studies have shown that, in comparison to non-gamified systems, students become more engaged and motivated in courses that use gamified systems [Zainuddin et al. 2020]. There are several examples of successful applications of gamification in learning environments. One example is the Multimedia Content Production (MCP) course at Instituto Superior Técnico, University of Lisbon. This gamified MSc course rewards students with Experience Points (XP) for the accomplishment of certain tasks such as skills concluded or badges earned. Each student can earn up to 20000 XP and, at the end of the course, the amount of XP accumulated is converted into a 0 to 20 score. Moreover, MCP has been the main focus of several studies and theses since it uses GameCourse, a platform that has been developed and improved throughout its years in use.

This gamified platform contains several game elements that contribute to a more engaging experience. Furthermore, to escape the common "one-size-fits-all" approach [Pratt 2002] and be able to take into consideration each user's preferences and needs, a way of profiling users was implemented and incorporated into the system with the intent of assigning them to a specific player profile. This new feature allows the existence of an adaptive gamified system where we can adapt the way each student profile views a certain parts of the system.

GameCourse was already a well-accomplished system with several functionalities that allows users to freely interact with it, however there still is room for improvement. As such, our goal is to polish the system by solving bugs and performance issues the system has, and incorporate new game elements that can improve the gamified experience and increase game element diversity.

Author's address: Catarina Gonçalves, Instituto Superior Técnico, Lisboa, Portugal, catarina.q.goncalves@tecnico.ulisboa.pt.

2 RELATED WORK

Gamification, defined as "the use of game elements and game design elements in non-game contexts" [Deterding et al. 2011], has become increasingly popular throughout the years due to its positive impacts and potential to improve motivation, engagement, and social influence. It may pass unnoticed, but teachers rewarding their students with stickers after completing a certain task or even companies or supermarkets giving out stamps for their clients to collect to achieve a certain prize, counts as the use of gamification.

The exponential increase in the interest in gamification, particularly in the fields of education, wellness and business, has been the catalyzer to the making of several studies regarding this topic. In all contexts, the use of gamification has shown to be an advantage in increasing user engagement [Zainuddin et al. 2020], improving behavior [Metwally et al. 2021] and knowledge retention [Putz et al. 2020]. When it comes to education and learning, the goal is to motivate students to engage and be active in a course to maximize their success. This is usually done by implementing a system of rewards or providing feedback by indicating their level of performance [Furdu et al. 2017].

Nearly all gamified platforms are built on the use of rewards as its main dynamic by awarding game elements like badges, trophies, and points to their users after they achieve a certain goal or finish a certain task. These achievement-related features [Xi and Hamari 2019] are visual displays of the users' progress which gives them immediate feedback [Krath et al. 2021; Kyewski and Krämer 2018], helping them assess their performance and accomplishments. In addition, to allow users to keep track of their own progress as well as their peers, gamified platforms usually incorporate levels and leaderboards, the latter considered one of the most engaging game elements [Garone and Nesteriuk 2019] since it positively impacts social comparison between users [Krath et al. 2021]. Furthermore, to promote cooperation, collective game elements are also implemented. Teams and, subsequently, team leaderboards, have shown to have many positive benefits, promoting social connection amongst users, making them engage in strong social connectivity via competition and comparison of points and scores [Chang and Wei 2016].

There are several examples of well-succeeded educational gamified applications, such as Class-Dojo, Duolingo, Kahoot!, Khan Academy and SoloLearn. Each one of these systems uses different game elements that further motivate its users to be active and stay

engaged. Duolingo, for instance, was created with the intent of providing the best universally available online education so that its users can learn a new language while having fun. To keep users motivated and engaged with the courses, Duolingo takes advantage of the use of game elements such as leaderboards, levels, points, progress bars, streaks and lingots (in-game currency). With a staggering total of 500 million users and around 40 million monthly active users, this approach can be, without a doubt, considered extremely successful. Other implementations concerning gamification follow a similar design, focusing on a single and generalized approach in which the game elements used are the same for all users.

Taking into consideration that individual needs are a major and crucial factor in gamification, they are, at the same time, one of the main causes for its applications to fail. As such, identifying user needs and preferences and discovering the best ways to use this information as an advantage in the implementation of gamified systems could help create a more inclusive version of gamification.

2.1 Tailored Gamification

The current gamified applications are designed based on a "one-size-fits-all" idea. However, studies have indicated that treating individuals as a homogeneous group is not an optimal design approach since everyone has its own preferences and needs. Researchers concluded that using personalized gamified learning experiences helps raise students' performance, achievements, and gamefulness experiences [Metwally et al. 2021].

Tailored gamification emerged as a means to improve the effectiveness of gamification [Rodrigues et al. 2020]. It can be described as the personalization of gameful design elements, the interaction mechanics, the tasks, or the game rules for each user, according to their preferences [Tondello and Nacke 2020]. This personalization can be achieved by customization, allowing the user to select the elements to be used, or by adaptation, where the system selects the elements for each user.

One particular promising approach to adaptation is understanding the relationship between player types and personality types and traits in relation to game elements and mechanics. This can lead to more appropriate and meaningful choices for gamified systems. Therefore, researchers became interested in implementing models that have the potential to improve the gamified experience, by automatically tailoring it to each individual. For that effect several models have been developed and tested.

Some of the conducted studies tested several models and found that, for each one, certain game elements are better suited for a particular trait or type of player. This means that all personalized gamified experiences depend on player profiling focusing either on player type models such as Hexad or Bartle's or on personality-based models such as the Five Factor Model (FFM), also known has the "Big Five" Model.

2.2 Player Profiling

Classifying users into groups that best fit their characteristics as players can help in tailoring the system and discarding the "one-size-fits-all" approach. This can be achieved by taking into consideration the users' personality traits or the player type of a specific model the users are assigned to.

The importance of personality traits has been widely acknowledged since it has a significant impact in daily tasks like working, learning [Codish et al. 1989] and, consequently, on academic achievement. The FFM, the most used model when it comes to personality, characterizes individuals based on their personality traits. This is a complete model that provides a coherent taxonomy [Buckley and Doyle 2017], being extremely helpful with player profiling. Due to its popularity, newer and revised versions of the FFM have been developed, for instance, the Ten Item Personality Measure (TIPI). However, since TIPI is a briefer measure, it is considered less reliable.

When it comes to player types, most studies rely on player or user typologies like BrainHex, Hexad and Bartle's player type model, one of the oldest and most frequently used. Bartle's player typology was created based on a specific game type, the Multi-User Dungeon (MUD), and divides users into eight player. However, this model provides a way to specifically classify MUDs' players which can have some application limitations since it may not work in other contexts. Another well-known player model is BrainHex, a top-down approach based on neurobiological insights [Nacke et al. 2014]. It categorizes users into seven player types: Achiever, Conqueror, Daredevil, Mastermind, Seeker, Socializer, and Survivor, each with distinct motivations.

Finally, Marczewski [Marczewski 2015] introduced the Hexad typology, specifically developed for gamification. This model was initially based on the self-determination theory (SDT) since the proposed user types differ based on how they can be motivated by either intrinsic (which refers to "[...] doing something because it is inherently interesting or enjoyable [...]" [Ryan and Deci 2000]) or extrinsic motivation (which refers to "[...] doing something because it leads to a separable outcome" [Ryan and Deci 2000] like receiving rewards).

Studies that employed player profiling was concluded that gamification is more effective when the interests and needs of its users are put into consideration. Regarding the relation between player typologies and game elements, results with Hexad were the most consistent with the definitions of its user types, and that its types had more influence on the perceived user motivation than those from previously discussed models [Hallifax et al. 2019]. It was also concluded that tailored gamification leads to higher performance than generic gamification. Thus, tailoring the experience to each type of player is an advantage when it comes to engaging users and improving their performance.

Concerning game elements, studies found that players that are motivated by extrinsic rewards enjoy leaderboards as well as virtual economy while players that wish to excel in the gamified system prefer levels, challenges and time pressure. Moreover, signposting, competition and guilds were suggested for players that enjoy interacting with others [Klock et al. 2020; Tondello et al. 2019]. However, most gamified systems and studies made use the same game elements leaving others unexplored.

There are several works that focused only on the most popular and common gamified components, such as badges, levels, leaderboards

and points. Thus, there are strong conclusions regarding these elements whilst there is little evidence for others, like teams, signposting or easter eggs [Klock et al. 2020], that could be particularly beneficial to use in gamification. When it comes to leaderboards, for instance, results showed that most students enjoyed the competitive environment it creates. However, a small group of the participants did not enjoy them which can suggest that the implementation of different types of leaderboards, including one for teams, could be an advantage [Aldemir et al. 2017]. On the other hand, Streaks, a not so commonly used game element in the existing literature, have shown promising results for Duolingo, Kahoot! and other gamified systems [Jiang 2018]. Kahoot!, for example, introduced a streak that rewards users for consecutive correct answers with the intent of preventing them from randomly clicking on answers just to get the next question. They found out that users cared more about their streak than overall score in points [Jiang 2018].

3 GAMECOURSE

GameCourse is a framework that courses can use to provide a gamified environment to their students. Its architecture is composed of several components that support its features, such as Course, CourseUser, Role, Modules and Views. Each user in the system is referred to as GameCourseUser and they have can have different Roles within a Course, the default ones being "Teacher", "Student" and "Watcher" but custom ones can be created. Moreover, GameCourseUsers can authenticate through Fénix, Google, Facebook or LinkedIn. A GameCourseUser can have Admin permissions which allows them to create new Courses, activate or deactivate users, and change Admin permissions.

The system also offers a repository with several Modules (Figure 1) that enable game elements, like Skills and Badges, and important tools, like retrieving data from external sources. All modules use the same vocabulary, that we refer to as **Expression Language (EL)** that enables semantic operations within the system. Additionally, when created, each module has its own library containing specific functions and variables that is stored in a global **Dictionary**. Each module can be enabled or disabled within a specific Course following some dependencies that may exist. It is important to note that every module in GameCourse depends on the Views module.

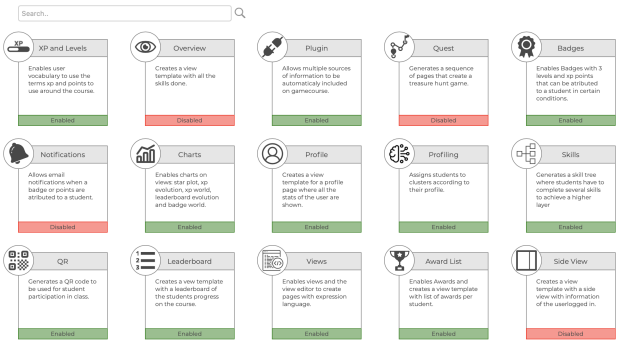

Fig. 1. Modules Page.

Views are the contents displayed to users, and they can be displayed differently for students with different Roles. For instance, within GameCourse, there are two types of Leaderboard, each being displayed to specific Roles. Additionally, modules can have view parts associated to them that are integrated in the system when they are enabled. Example of this is are the Leaderboard module that incorporates the Leaderboard Page, and the Skills Module that incorporates the Skill Tree, that displays the several skills students can complete.

Furthermore, through the Plugin Module, GameCourse is able to support the functionality of retrieving data from external sources, like Moodle, GoogleSheets and ClassCheck. This data is inserted as Participations, and is later transformed into awards by the component responsible for running the established rules, the Rule System.

Rules are a text-based rule written in Python with *when* and *then* clauses (Figure 2) that are run for the students that made new Participations within a course. In the *when* clauses are the preconditions that, when met, will make the actions in the *then* clause to be performed when the system runs. The rules can be created to perform any action the user might need and they may vary depending on the course. Consequently, they are stored in different text files according to their context and course. In MCP, for instance, there is a file for each game element for which rules are needed, one for badge-related and another for skill-related rules. Due to the rules-related dependability some modules may have, the Rule System also provides the automatic generation of rules based on a given template. However, this is only available for the Skills Module and only happens when a user creates a skill. As such, if a skill is edited, no change will happen to its rule.

```
rule: Quiz Grade
# give grades from quizzes

        when:
                logs = GC.participations.getParticipations(target, "quiz grade")
                flogs = filter_quiz(logs, "Dry Run")
                len(flogs) > 0

        then:
                award_grade(target, "Quiz", flogs)
```

Fig. 2. Example of a rule.

Additionally, GameCourse offers a Rule Editor that allows users to perform multiple actions related to the rules. In the Rules page, the users views a list of the existent rules within the course. In a rule's editing page, the user can establish the name and description of the rule and then write it. The page also contains a mechanism that provides tailored suggestions to the user. This is done in three different ways: by automatically suggesting a function while the user types, which is an autocomplete functionality that displays all the suggestions that match the syntax typed, by displaying those same suggestions on the function suggestion box to the right of the page and by listing the available metadata variables.

Another important mechanism within the Rule System is the logging mechanism. It is responsible for writing into individual files, one for each of the existing courses in GameCourse, the dates at which the rule system started and finished running as well as any errors that might have occurred while the Rule System was running.

Considering that this system can crash or return incomplete results due to any errors that can appear, this logging mechanism has proven to be extremely helpful since it allows us to understand the cause and locate the root of an error that occurred while the rule system was running.

4 IMPROVING GAMECOURSE

Our work started with the Plugin Module that, as previously mentioned, allowed multiple sources of information to be automatically included on GameCourse. However, since all the data sources were contained in single module, enabling it would include all of them in the system, even if the user did not require them all. Additionally, there were no visual indicators of what were the different sources that can be enabled for their data to be included in the system. To find out, a user had to first enable the module and then visit its configuration page.

As such, we divided this module into four new ones, each corresponding to a different data source. The configuration pages of these new modules were also revised to have a cleaner look and improve understandability. Code-wise, each data source was already being handled individually, having a script responsible for retrieving the data in question. However, the functions responsible for storing and editing the variables of each data source were all within the same file, which we had to compartmentalize.

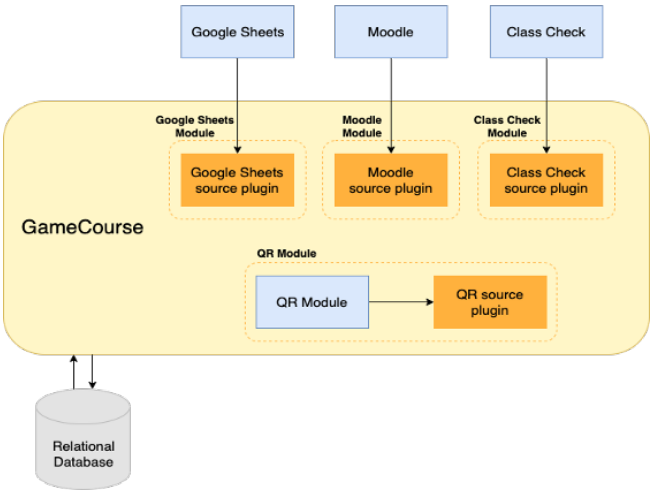

Fig. 3. GameCourse's data sources new architecture.

The page where the modules are displayed also lacked organization since all the available modules appeared in alphabetical order. As a consequence, to separate game element modules from data source modules, two sections were created in the Modules page. In order for the system to know which section each module belongs to, changes in the modules' creation were made. Now, when creating a new module, its type needs to be specified: "DataSource" or "GameElement". Later on, we decided to add a new type called "Tools" since modules like Fenix, Profiling or Notifications, did not really fit in any of the already existing types. Now, the Modules Page within GameCourse has three different sections, as seen in Figure 4, which culminated into a better organization of the modules.

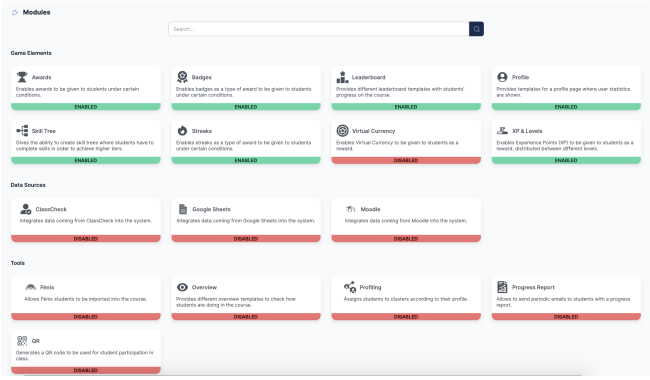

Fig. 4. New Modules Page.

4.1 New Game Elements

We implemented and integrated a total of three new game elements to the system: Virtual Currency and Streaks, both used in MCP 2021/2022, and Teams. To do so, we had to create the respective modules that would encapsulate their functionalities. When it comes to their actual implementation, there are three commonPrequirements that must be met. As such, for each module, we created a configuration page to support its functionalities, added the necessary tables in the database, and created all the rules that allow these new modules to have the desired effects, including the functions needed for the rule.

4.1.1 **Virtual Currency**. The first game element implemented was the Virtual Currency. We wanted to create a concept similar to shopping where students could earn tokens after completing a task that would then be saved on their wallet and could later be spent to perform certain tasks. Since this game element was going to be used for MCP 2021/2022, our first approach was to understand how the course professors intended on using it and how the students would earn and spend their tokens.

Students would earn them by peer-grading their colleagues as well as completing Streaks, a new game element that we discuss later in Section 4.1.2. Furthermore, they could only spend tokens on retrying a skill or unlocking a wildcard, a type of skill that is not bound to dependencies or levels. In addition, it should also be possible for them to exchange their tokens at the end of the course.

First, we created a prototype of how the this new module was going to be displayed for the students. The total number of tokens a students possesses is shown next to the student's number and in the Profile Page (Figure 5). Additionally, the cost of a skill and the total number of attempts made are shown right next to it, as can be seen in Figure 6.

Then, five tables were added to the database allowing the system to store all the meaningful information related to this module. One of the tables stores the general attributes, like the name of the tokens and tokens-to-XP ratio. The others store each student's total number

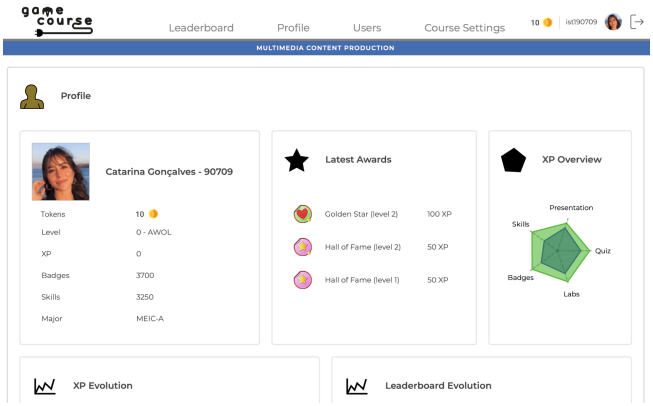

Fig. 5. Virtual Currency Prototype - Total amount of tokens.

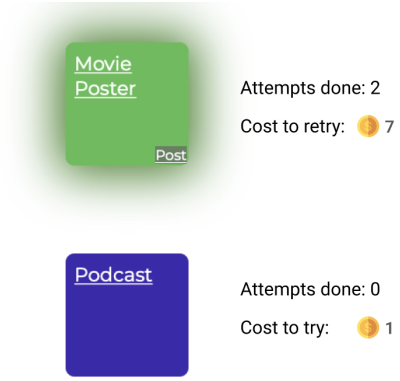

Fig. 6. Virtual Currency Prototype - Currency in the Skill Tree.

of tokens, the specific Participations they performed that cost them tokens, and the Participations the system should award or remove tokens. An example of the latter is, for instance, if the Professors of a course want to award the students each time they perform a peer-grade assessment or participate in a lecture. To finalize, we created all the functions responsible for inserting and updating this data in the tables.

Finally, we created functions in the Rule System that are responsible for awarding or removing tokens and updating a student's wallet. This allowed the creation of rules responsible for performing these changes in the system.

4.1.2 **Streaks**. In order to keep track of the tendency towards consecutive behaviors and awarding students for it, the Streaks module was implemented. Similar to the Virtual Currency module, our work began with understanding how this new addition was going to be used for MCP 2021/2022 including what was required be displayed within the module's configuration page and within the system.

Nonetheless, Streaks are more complex than an in-game currency. For a streak to be properly implemented, specific information must be stored. First, each streak should have a **name** and **description** so that a user can easily understand its goal and requirements. Then, an **accomplishment count** must be stored so that the system knows when to award it as well as a counter of valid Participations for each student, representing the **progression** made to complete a streak. Finally, streaks can be time-based, repetition-based, or both. As such, this should also be stored, including the **periodicity** that must be respected, so that the system can perform the correct verifications.

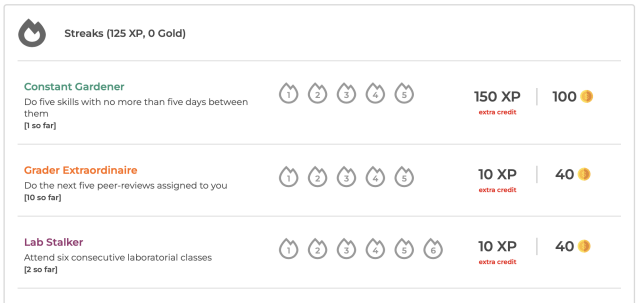

Fig. 7. Streaks within a student's Profile page.

In the configuration page of this module, there is a section for the general attributes and below that another containing a detailed list of all the existing streaks in the course. Here the user can perform a set of actions, such as creating a streak, editing, duplicating, or removing an existing one as well as importing and exporting the streaks. In the creating and editing streak dialog, a user can establish all the information mentioned above as well as the streak's color and the amount of XP or tokens that should be given to a student once the streak is completed.

Once this was done, we implemented the functionality required for the module to work as expected, including the addition of four new tables in the database. Two of them are responsible for storing the general attributes of the module and all of the streaks with the information mentioned above. In the other two, the progression and Participations made for each streak and student are stored.

This was followed by a crucial part of this module's development, its integration in the Rule System. For the streaks to be correctly awarded, new rules were created. However, there were no functions in the Rule System that could perform the verifications needed. As previously mentioned, streaks could be time-based, repetition-based or both. A repetition-based streak checks for consecutive behaviors, *like attending four consecutive lectures*, while a time-based streak performs time-verifications between behaviors, like *doing a total of three submissions, one every five hours*. Additionally, the streaks can also be awarded repeatedly, which means that once an award for that streak is given to a student, the streak resets to zero. Therefore, we created functions responsible for validating, according to the type of streak and its attributes, the participations each student made.

The functions created cover all the cases of streaks that were used in MCP 2021/2022 and other possible verifications that could be used in future. When it comes to time-verifcations the functions cover the comparison of timestamps for minutes, hours, days and weeks, only checking if the participations made respect the streak's established

periodicity. On the other hand, for repetition-based streaks, we had to be more careful and understand how this information is stored in the database since each external data source inserts data in the system with specific information. For instance, to check for consecutive lab or lecture attendance, we have to use the *description* column since it holds the number of the lab/lecture (1, 2, 3, and so on), as can be seen in Figure 8 . Furthermore, a function responsible for awarding a streak to a student was also created.

id	user	course	description	type	post	date	rating	evaluator
83	7	2	1	attended lecture	NULL	2022-06-10 18:51:31	NULL	6
84	7	2	2	attended lecture	NULL	2022-06-14 16:34:19	NULL	6
85	7	2	3	attended lecture (late)	NULL	2022-06-17 20:46:28	NULL	6
86	7	2	1	attended lab	NULL	2022-06-09 15:12:37	NULL	6
87	7	2	2	attended lab (late)	NULL	2022-06-13 18:29:49	NULL	6
88	7	2	3	attended lab	NULL	2022-06-16 21:08:10	NULL	6
89	7	2	Quiz 1	quiz grade	NULL	2022-06-10 13:31:47	1000	6
90	7	2	Quiz 2	quiz grade	NULL	2022-06-14 15:45:09	1000	6
91	7	2	Quiz 3	quiz grade	NULL	2022-10-22 19:21:41	475	6

Fig. 8. Examples of consecutive Participations.

4.1.3 **Teams.** Last but not least, the Teams Module was created. Implementing a collaborative game element was bound to improve the diversity of elements in GameCourse since most of them are for individual practices like skills, streaks and badges. As such, students can now be assigned to a team and perform evaluations as one. Using MCP as an example, a course that supports group evaluation, there is the group presentation, but there are also certain laboratory classes where the evaluation needs to be done by the whole group. This grade is given individually to each student but, it can now be awarded to the team as well.

In the configuration page, a user is able establish the maximum number of elements a team can have as well as allow teams to have a name. Creating teams can either be done manually or by importing a Comma-Separated Values (CSV) file with a specific format. Additionally, the create or editing page of a team contains a search bar that where a user can write the name of a student. Below that, there is a section that lists all the course users that are not yet inserted in a team and, if anything is typed by the user, that list will be filtered accordingly. Since there is a maximum number of elements per team, once that maximum is reached, a warning will appear on the create/edit team modal while the button to add an element will disappear (Figure 9). This way the user will not be able to add any other elements to the team.

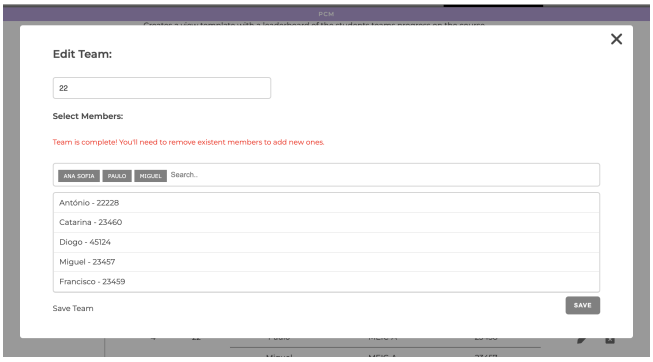

Fig. 9. Edit Team Modal.

The teams and respective team members are stored in the two distinct tables in the database. Additionally, we created two other tables where the total amount of XP and tokens a team possesses are stored, the teams_xp and teams_wallet, respectively.

To finalize the integration of this new module in the system, we implemented rule system functions that could be needed in the future and wrote template rules to be used as references. One of them takes into consideration MCP and its group presentation, awarding the XP earned to each team. Nevertheless, new rules can be easily written due to the suggestion mechanisms of the Rule Editor or they can also be written based on the existing rules that award prizes, grades or game elements to users individually.

Additionally, when enabled, this module allows the existence of a leaderboard specifically for teams. As it can be seen, the Teams Leaderboard displays the team members as well as the team's number, total XP and current level.

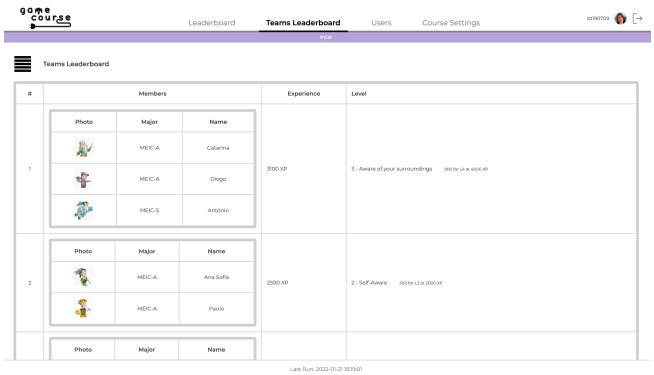

Fig. 10. Team Leaderboard.

4.2 Rule System Improvements

Our work within the Rule System began while the MCP 2021/2022 course was being taught. Once students started to do skills that had dependencies, the rule system would not award them even if all the preconditions of the respective rule were met or it would crash trying. For instance, to unlock the "reTrailer" Skill, a student must first unlock both the "Looping GIF" and "Publicist" Skills. Using this example, to award a dependant skill like "reTrailer", its rule checks, through the rule_unlocked function, if the rules responsible for awarding the "Looping GIF" and "Publicist" Skills have been unlocked since this would mean that the awards for those skills were given.

However, even though this function had worked in the previous iteration of the course, it no longer did. As such, the system would not award any skill with dependencies which was a big concern considering there were 18 different dependent skills in the Skill Tree. To fix this, we implemented a new function to replace the existing one that accesses the award table to check if the skills in question had been awarded to that user.

While exploring the system to check if there were any other bugs, we discovered that when a game element, like a badge or skill, is deactivated, its respective rule is not. This meant that the rules of

the deactivated game elements would still be run by the system, which should not happen. A rule is not active when it contains the word INACTIVE right below its name, as explained in Section 3. Furthermore, rules are grouped and stored into separate files, each for a specific game element. There are also rules that do not belong in these files, performing other actions that are not to award a game element, and, as such, are stored in a different file. This organization allows us to easily find where the rule is stored according to its purpose.

At this stage, the rule name had to match the one of the game element so that the system was able to correctly retrieve the rule. Therefore, we created a function that receives as arguments the type of the game element (badge, skill, streak, and so on) and the status the rule should have, i.e., active or not. This way the system is able to open and read the correct rule file and change the rule in question. This function is called when the user clicks the toggle button responsible for activating or deactivating the game element, which adds or removes the INACTIVE tag from the rule.

4.2.1 **Performance Improvements .** During MCP 2021/2022, we also observed a poor performance in the Rule System, more specifically, the time it took to run. With more data being inserted into the database every time it ran, this problem would aggravate, and as we got closer to the end of the course, the system would take at least an hour to finish running or would crash trying. We started by deactivating the rules that were no longer needed to decrease the number of verifications done each iteration, thus, decreasing the time it took. Once the course ended, we began exploring the problem and found three main issues:

- **Autogame would not restart on its own** if an error was thrown. This meant that when it crashed it would remain stuck in an unsuccessful loop of trying to run until a user manually fixed this problem.
- **A connection to the database per function** in the RuleSystem. As such, each time the Rule System ran, several connections would be made which can be extremely time consuming.
- **Repeated MySQL queries** executed throughout the code that, even when their results remain unchanged during the system's execution.

Nevertheless, using the Python time and logging modules, we measured the time each rule and its functions would take to run for a single student. We discovered that the Dictionary functions responsible for retrieving data were taking the longest to execute, with some of them taking almost five seconds. These functions are stored in specific libraries and the system retrieves them through a socket, which could be the reason they were taking so much time. In addition, the functions responsible for awarding a particular game element, grade, or prize were also very time-consuming, with the award skill and award streak taking the longest. These functions are of extreme importance to the system since most rules within a course rely on them to gather all the specific logs of a certain target and produce the desired and expected outcome like, for instance, awarding a streak after its requirements are met. As a consequence, rules that were dependent of these functions ended up taking almost eight seconds in total to execute which, for a course with around 100 users, would culminate in an excessive amount of time to run the Rule System, something that we had already experienced in MCP 2021/2022.

To fix this, we replaced the functions in question with new ones that access the database directly to retrieve data according to the arguments given. We also performed a refactor of the functions responsible for awarding the game elements, grades or other prizes. The award_skill and award_streak functions had a lot of conditional statements that should be done in a different function and called as a precondition in the rules. After their implementation, we ran the Rule System once more to measure the time these new functions took. A notable difference difference in times was observed with some rules taking, approximately, one-fifth of the time they did before.

Then, we replaced all the connections to the database that were made and closed in each function by a single connection made once the Rule System starts that is closed once the system finishes running. Additionally, we created a data broker to store queries and its results. The data broker consists of two nested Python dictionaries, one for course-related queries and the other for students. We chose to use dictionaries over the other data structures since they are faster, having a constant time complexity $O(1)$ for most of the operations to be performed. As such, a query whose results remain unchanged throughout the Rule System's execution are stored in the data broker as a key-value pair and the system only needs to retrieve the result, avoiding unnecessary query executions.

To conclude, we fixed the issues related with AutoGame not restarting on its own taking advantage of the logging mechanism. Each time something is written to the log file of a course, a separator before and after is added. As such, when the Rule System finishes running, the file ends with the separator, as can be seen in Figure 11. However, when an error occurs this does not happen. This allows us to check whether an error as occurred or not. Consequently, we created a function responsible for reading the last line of the file and, if it did not contain the separator, the Rule System needed to be restarted.

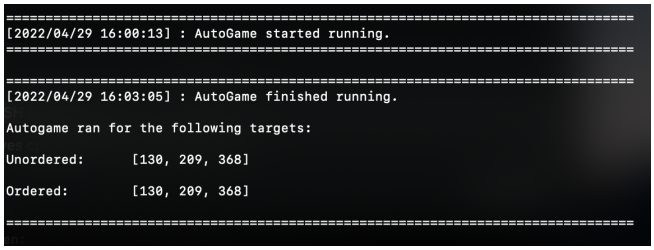

Fig. 11. Log file section separator.

4.2.2 **Automatically Generated Rules .** Another important implementation made and integrated in the system was the automatic generation of rules. This functionality, although already implemented for the Skills, was incomplete and faulty. When a user created a new Skill, the system was able to create the respective rule that would award it. However, if a user edited a skill, the rule would remain unchanged. As a consequence, the results of firing this rule

would most likely be wrong and produce, for instance, unwanted awards.

The automatic generation of rules requires the creation of templates for the system to use. To create rule templates for a game element, we need to understand what information they should hold and how many are needed. For instance, for skills, there are two templates where the preconditions vary: one is for skills that have normal dependencies and the other for skills that depend on wildcards.

As such, we created the necessary templates for the only remaining game elements that need rules, which are badges and streaks. Then, the functions responsible for using and changing the templates according to the element were created. Now, when a user creates a new game element, its rule is automatically created, and, if a game element is changed, the respective modifications are made to the rule. Additionally, we decided to keep a record of the changes made to the rule by commenting what is going to be changed and writing the updated version below. This allows a user to know what was changed.

Furthermore, if the Virtual Currency module is enabled, some rules would need to be updated to include currency-related functions. However, even though these functions are implemented in a way that does not require any changes to be made, we cannot guarantee that the user would want these functions. As a consequence, we decided to keep them commented within each template to allow the users to freely decide whether to include it or not.

```
rule: reTrailer

        when:
                #CHANGED:
                #combo1 = rule_unlocked("Course Logo", target) and rule_unlocked("Movie Poster", target)
                #combo2 = rule_unlocked("Album Cover", target) and rule_unlocked("Publicist", target)
                #combo1 or combo2

                wildcard = GC.skillTrees.wildcardAvailable("<skill-name>", "<tier-name>", target)
                combo1 = rule_unlocked("Publicist", target) and rule_unlocked("Album Cover", target)
                combo2 = wildcard and rule_unlocked("Course Logo", target)
                combo1 or combo2

                skill_based = combo1
                use_wildcard = False if skill_based else True

                logs = GC.participations.getSkillParticipations(target, "reTrailer")
                rating = get_rating(logs)
                rating >= 3

                # NOTE:
                # Virtual Currency Enabled ? Uncomment code below : Delete

                # valid_attempts = get_valid_attempts(target, "reTrailer")
                # valid_attempts > 0
                # (new_total, removed) = get_new_total(target, valid_attempts, rating)
                # new_total >= 0

        then:
                #CHANGED:
                #award_skill(target, "reTrailer", rating, logs)

                award_skill(target, "reTrailer", rating, logs, use_wildcard, "Wildcard")
                # update_wallet(target, new_total, removed, logs)
```

Fig. 12. reTrailer Rule after changing the skill dependencies.

Finally, since there are rules that require a participation type to be specified, once game element is created or edited, the user is redirected from the current page to the editing page in the Rule Editor, displaying the rule. This way, a user can check if the rule was correctly written and perform any changes needed.

5 EVALUATION

The Streaks and Virtual Currency modules were developed and integrated within GameCourse in time for them to be used during the Multimedia Content Production 2021/2022 course. During the seven- week period in which the course was taught, we were able to gather feedback and improve the developed functionalities by fixing any relevant bugs that were found. Before the course began, we created a GameCourse test environment where we could test the modules beforehand to guarantee they were properly working. This was done by testing if the changes made within the configuration pages were having the proper effect in the database as well as verifying whether the new rules were correctly implemented and awarding streaks and tokens.

We were able to find and fix a few bugs, most of them related to the time verifications for time-based streaks. Since the streak-related rules are complex and of extreme importance, it was clear that manually populating the database and running the Rule System to test them was not an optimal approach. As such, we created a PHP script that covers eight different streaks and that the users can simply run to check if the streaks are properly working. So that a user does not have to worry about the database, the script is responsible for both populating the database to set up for the tests as well as cleaning up afterward, which guarantees that it starts with a clean environment every time it is run. After setting up the environment, the script makes a call to the function that runs the Rule system so that the necessary rules are fired. Afterwards, it checks if the correct awards were given.

Even though we tried to cover all the possible error scenarios, some streak-related bugs within the Rule System functions appeared throughout the course and were pointed out to us by the students discovered. The first bug that appeared was related to the streaks' progression: if a student only had one valid Participation for a streak, it was not being counted as a Participations for the progression. This was due to the fact that, within the code, the system would iterate through the received Participations in pairs. Therefore, in the beginning of the functions responsible of their validation, we added a verification regarding the number of Participations received as argument. If there was only one, we would add it to the progression of the streak since the first Participation is the one used for the first comparison and, as such, is always valid.

Another bug that was caught during MCP was that the rules were being fired for professors that had previously been students in a different course within the system. IN MCP 2021/2022, there were two professors that were enrolled as students in MCP 2020/2021, a course that, although inactive, still existed in GameCourse and, subsequently, in the database. We detected that rules were being fired for those two Professors. The root of this problem was that the queries responsible for retrieving the Rule System targets did not take into consideration the existence of several courses within the system. As such, it would just retrieve users with the Student role without specifying the course. To fix this, we had to filter the results by course, adding this verification to the query.

5.1 User Tests

To evaluate the success of our implementation regarding the new game elements, we resorted to user testing since we could observe real users attempt to complete a set of tasks. This allowed us to comprehend if the new modules were easily understood by the users and, if not, what the sources of confusion were. We came up with a set of 14 tasks that covered all three modules. Before giving out the list of tasks to the users, we randomized its order for each

user to assure that there was no influence caused by the learning curve.

We conducted tests with 21 participants, which is a number above the minimum number required for a summative analysis. The most common age range of the participants was 21 to 25 years old. However, there were some participants' were between the ages of 16 to 20, 36 to 40, and 56 to 60. Six participants had previously used GameCourse as students while the remaining fifteen were not acquainted with the system.

Once a participant was ready to start a task, we would start the timer. After the task was complete, the participants were asked to rate that task in a scale of **1 (Very Difficult)** to **7 (Very Easy)** and also had to answer a questionnaire whose answers would allow us to calculate the NASA Task Load Index. Finally, once the all the tasks were done, the participants had to answer a final questionnaire where they were asked about their overall experience and if they had any suggestions to improve the system.

As it can be seen in Table 1, participants did not have difficulty in performing 12 out of 14 of the defined tasks. On the other hand, tasks number 5 and 6 had a success rate of 76,19% and 47,62%, respectively. We can also observe that the largest number of errors occurred in tasks 5, 6, 12, 13 and 14, two of them having the lowest success rate. These tasks either required the creation of two distinct streaks, each with different properties, or required the creation, modification or import of a teams into the system. Additionally, when it comes to the time each participant spent of a task, we can conclude that tasks 5 and 6, the ones with the lowest success rate, were the most time consuming.

Additionally, we calculated the NASA-TLX so that we could better understand the amount of mental and physical effort the users had to apply to perform each task. The average score obtained was 18.04, which is a relatively low score considering it can go from 0 to 100. Consequently, this reflects that the users' perceived mental workload when performing the tasks was low. Moreover, at the end of the questionnaire given to the users, all the 10 questions of the System Usability Scale (SUS) were asked so that we could calculate the system's overall SUS score. After calculating the sum of all the individual scores, we obtained a 89.88 average SUS score, which is considered an excellent result.

Table 1. Success Rate, mean ($\overline{x}$) and standard deviation (σ) values of the information collected from each task.

Task	Success Rate (%)	Time (s)		Nr. of Clicks		Nr. of Errors	
		$\overline{x}$	σ	$\overline{x}$	σ	$\overline{x}$	σ
1	90.48	20.0	6.5	3	0.45	0.10	0.30
2	100	31.4	16.5	6.86	0.57	0	0
3	100	27.6	11.0	5.24	0.44	0	0
4	100	6.4	5.5	2	0	0	0
5	76.19	50.4	15.7	8.76	0.94	0.29	0.56
6	47.62	42.9	20.9	9.86	1.62	0.71	0.85
7	100	23.6	11.9	4.81	0.75	0	0
8	100	7.8	10.7	2.14	0.65	0.05	0.22
9	100	4.2	3.8	1.38	0.80	0	0
10	100	6.5	6.5	1.19	0.60	0	0
11	100	11.4	8.2	2	0	0	0
12	100	25.3	9.3	6.33	0.66	0.76	0.46
13	95.24	19.6	10.8	5.10	0.70	0.17	0.51
14	100	21.7	4.6	5.05	1.02	0.33	0.48

5.2 Performance Testing

Finally, performance testing was conducted to allow us to comprehend if the changes made in the Rule System had indeed improved its performance. We had already tested out the time each function and rule took to individually run for a single student, however, we had to apply it to several students firing a rule for each one, which resulted in awards being given.

We performed tests for 20, 40, 60, 80, and 100 students, awarding a game element per student. Since the rules responsible for awarding streaks and skills had proven to be the most time-consuming and were the main focus of the changes made in the Rule System, we only fired those specific rules (one for each student). We can observe, in Figure 13, the execution times of the oldest and newest versions of the Rule System took to fire the same rules for a different number of students. As can be seen, for smaller groups of students the differences between the execution times are smaller. Nonetheless, the new version is indeed faster than the old one, and, for groups with more than 40 students, it takes less than half the time the old version took, which is a significant improvement.

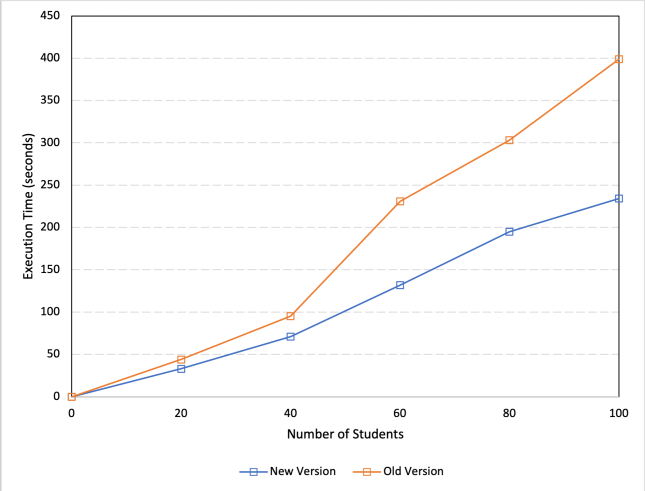

Fig. 13. Rule System execution times by number of students.

The results from these tests were positive since the execution times for different users and awards given had significantly decreased, which matched our expectations.

6 CONCLUSIONS AND FUTURE WORK

Even though GameCourse offers several functionalities that contribute to a better gamified experience, our work aimed to improve the existing ones as well as implement and integrate new modules to increase game element diversity. Additionally, we also intended to improve the Rule System's performance, a crucial part of GameCourse, that was taking a long time to execute.

With those goals in mind, our work, GameCourse - The Next Level, added three new game elements, Streaks, Teams and Virtual Currency. The use of the Streaks and Virtual Currency Modules in MCP 2021/2022 revealed some vulnerabilities in their implementations that we did not forsee, but were able to quickly fix. Additionally, the feedback obtained from the students at the end of the course, allowed us to conclude that, between these two new game elements, Streaks were considered more engaging. Although there is still room for improvement, we can conclude that these new game elements were valuable additions to GameCourse.

Additionally, to understand if the new modules were correctly integrated, and were of easy understandability for a user, we conducted user tests with 21 participants. The tests gave us the needed insights regarding any areas of confusion and weaknesses of our work. Overall, the user tests allowed us to further improve the modules' functionalities and usability.

Lastly, we improved the Rule System's performance by tackling the four main issues identified. To evaluate what were the effects of our work in the system's performance, we conducted performance tests. The results of the tests performed showed that the new version takes less than half the time the old version took, which is a significant improvement. This allowed us to confirm that we improved the poor performance of the Rule System, achieving our goals.

However, there still is room for improvements. When it comes to the Teams module, a way of automatically generating teams and randomly assigning students to them should be explored and implemented since it may suit certain courses better. In addition, the system could be further improved to allow a more complete integration of the teams since the whole system is extremely tailored to individual Participations. This means replicating how the users data is dealt with and applying it to teams.

In addition, regarding the Rule System, there are changes that can be made. The connector module could be separated into files, each containing the functions used in the same context, which would culminate in a better organization of this part of the system. Additionally, the automatic generation of the rules should be a global functionality that each module can simply use. As of now, if a user wants to have this functionality within a new module, it needs to replicate the existing code to then apply it to the module.

Moreover, the Rule Editor could have a highlight functionality that could, for instance, give a specific color to a certain part of the rules like what is done in text editors. This would be extremely advantageous for users to be able to visually differentiate from the actual parts of the rule that are going to be run and the ones that are commented.

REFERENCES

Tugce Aldemir, Berkan Celik, and Goknur Kaplan. 2017. A Qualitative Investigation of Student Perceptions of Game Elements in a Gamified Course. *Computers in Human Behavior* 78 (10 2017). https://doi.org/10.1016/j.chb.2017.10.001

Patrick Buckley and Elaine Doyle. 2017. Individualising gamification: An investigation of the impact of learning styles and personality traits on the efficacy of gamification using a prediction market. *Computers & Education* 106 (2017), 43–55.

Jen-Wei Chang and Hung-Yu Wei. 2016. Exploring engaging gamification mechanics in massive online open courses. *Journal of Educational Technology & Society* 19, 2 (2016), 177–203.

David Codish, Israel Beer-Sheba, and Gilad Ravid. 1989. PERSONALITY BASED GAMIFICATION: HOW DIFFERENT PERSONALITIES PERCIVE GAMIFICATION Research in Progress. *No. Davis* 2012 (1989).

Sebastian Deterding, Dan Dixon, Rilla Khaled, and Lennart Nacke. 2011. From game design elements to gamefulness: defining" gamification". In *Proceedings of the 15th international academic MindTrek conference: Envisioning future media environments*. 9–15.

Iulian Furdu, Cosmin Tomozei, and Utku Kose. 2017. Pros and cons gamification and gaming in classroom. *arXiv preprint arXiv:1708.09337* (2017).

Priscilla Garone and Sérgio Nesteriuk. 2019. Gamification and learning: A comparative study of design frameworks. In *International Conference on Human-Computer Interaction*. Springer, 473–487.

Stuart Hallifax, Audrey Serna, Jean-Charles Marty, Guillaume Lavoué, and Elise Lavoué. 2019. Factors to consider for tailored gamification. In *Proceedings of the Annual Symposium on Computer-Human Interaction in Play*. 559–572.

Tiffany Jiang. 2018. Research: In-game streaks. https://blog.prototypr.io/research-in-game-streaks-92bfb229e776

Ana Carolina Tomé Klock, Isabela Gasparini, Marcelo Soares Pimenta, and Juho Hamari. 2020. Tailored gamification: A review of literature. *International Journal of Human-Computer Studies* 144 (2020), 102495.

Jeanine Krath, Linda Schürmann, and Harald FO von Korflesch. 2021. Revealing the theoretical basis of gamification: A systematic review and analysis of theory in research on gamification, serious games and game-based learning. *Computers in Human Behavior* 125 (2021), 106963.

Elias Kyewski and Nicole C Krämer. 2018. To gamify or not to gamify? An experimental field study of the influence of badges on motivation, activity, and performance in an online learning course. *Computers & Education* 118 (2018), 25–37.

Andrzej Marczewski. 2015. Even Ninja Monkeys like to play. *London: Blurb Inc* (2015).

Ahmed Hosny Saleh Metwally, Lennart E Nacke, Maiga Chang, Yining Wang, and Ahmed Mohamed Fahmy Yousef. 2021. Revealing the hotspots of educational gamification: An umbrella review. *International Journal of Educational Research* 109 (2021), 101832.

Lennart E Nacke, Chris Bateman, and Regan L Mandryk. 2014. BrainHex: A neurobiological gamer typology survey. *Entertainment computing* 5, 1 (2014), 55–62.

Daniel Pratt. 2002. Good Teaching: One Size Fits All? *New Directions for Adult and Continuing Education* 2002 (03 2002), 5 – 16. https://doi.org/10.1002/ace.45

Lisa-Maria Putz, Florian Hofbauer, and Horst Treiblmaier. 2020. Can gamification help to improve education? Findings from a longitudinal study. *Computers in Human Behavior* 110 (2020), 106392.

Luiz Rodrigues, Armando M Toda, Paula T Palomino, Wilk Oliveira, and Seiji Isotani. 2020. Personalized gamification: A literature review of outcomes, experiments, and approaches. In *Eighth International Conference on Technological Ecosystems for Enhancing Multiculturality*. 699–706.

Richard M Ryan and Edward L Deci. 2000. Intrinsic and extrinsic motivations: Classic definitions and new directions. *Contemporary educational psychology* 25, 1 (2000), 54–67.

Gustavo F Tondello, Alberto Mora, Andrzej Marczewski, and Lennart E Nacke. 2019. Empirical validation of the gamification user types hexad scale in English and Spanish. *International Journal of Human-Computer Studies* 127 (2019), 95–111.

Gustavo F Tondello and Lennart E Nacke. 2020. Validation of user preferences and effects of personalized gamification on task performance. *Frontiers in Computer Science* 2 (2020), 29.

Nannan Xi and Juho Hamari. 2019. Does gamification satisfy needs? A study on the relationship between gamification features and intrinsic need satisfaction. *International Journal of Information Management* 46 (2019), 210–221.

Zamzami Zainuddin, Samuel Kai Wah Chu, Muhammad Shujahat, and Corinne Jacqueline Perera. 2020. The impact of gamification on learning and instruction: A systematic review of empirical evidence. *Educational Research Review* 30 (2020), 100326.