A User Experience and Usability Test on Creating Games Specified as Graph Grammars

Júlia Veiga da Silva[1], Braz Araujo da Silva Junior[1], Luciana Foss[1], Simone André da Costa Cavalheiro[1]

[1]Fundamentals of Computing Laboratory – Federal University of Pelotas (UFPel)
CEP 96.010-610 – Pelotas – RS – Brazil

{jvsilva, badsjunior, lfoss, simone.costa}@inf.ufpel.edu.br

Abstract. *GrameStation is a game engine based on Graph Grammar that allows creating and playing games modeled with this language. Graph Grammar is an intuitive formal language, but people who do not have previous experience with it may have difficulty understanding/specifying games in this way. Therefore, we propose an experiment to analyze the support provided by GrameStation during the creation of games modeled as Graph Grammars in order to facilitate this task. We analyzed three groups of people with different levels of knowledge about Graph Grammar. They created a game following provided instructions and answered a questionnaire. It is concluded that GrameStation is suitable for those who have experience with it, but presents gaps for people without experience. These results will guide the implementation of pedagogical agents in the tool.*

1. Introduction

Being introduced to the school curriculum in several countries, computing can bring challenges to teachers. Thus, learning appropriate pedagogies for teaching content such as algorithms, programming, and **C**omputational **T**hinking (**CT**), becomes fundamental [Sentance and Csizmadia 2017]. In this context, digital games are commonly used as an educational resource because, interactive and visually attractive, they provide fun and stimulate students' creativity. Among these skills, CT has been intensively explored in the scientific community. It is a problem-solving process based on **C**omputer **S**cience (**CS**) that should be learned by everyone, not just CS professionals [Wing 2006]. Thus, focusing on this process and proposing a new approach for CS education, mixing educational games, active methodologies, and creative learning, a game engine called GrameStation [Silva Junior et al. 2021] was proposed in which Graph Grammars should be used to create games.

GrameStation was developed on the Unity platform [Technologies 2022] using C# codes and it is based on a mathematical formalism called **G**raph **G**rammar (**GG**) – a visual and formal language used to describe systems and verify properties. Therefore, GrameStation allows making, editing, and running Graph Games. It means that creating a game corresponds to specifying a GG in the tool. However, although intuitive, specifying a GG may be a completely new ground for many. That means they are likely to have a lack of knowledge about fundamental concepts and terms from GG. Thus, in order for GrameStation to reach the general public, it should be able to provide the proper support for presenting this language in a friendly and engaging manner.

Facing this challenge, pedagogical agents were proposed as an effort to help provide direction to users on GrameStation [Silva et al. 2021]. Hence, an experiment was conducted to collect the main feedback from users regarding GrameStation as a way to guide the future implementation of these agents. An experiment, previously conducted, with a focus on analyzing the support given by the tool during the game execution, was presented in [Silva et al. 2022]. In this experiment, it was concluded that the participants did not have difficulty using GrameStation, but rather in understanding how the games should be played. Consequently, the addition of tutorials, as a complement to the visual feedback already provided by the tool, was suggested by the participants.

Therefore, this paper presents a usability and user experience test of GrameStation aimed at analyzing the tool's support during the creation of games (GG specification) and identifying areas for improvement. We adopt the same empirical strategy suggested by Nielsen (1994) to inspect the usability, which was also used in the first experiment [Silva et al. 2022], conducting tests with real users. Nineteen individuals with CS background were selected to participate in this experiment and each of them created a game in GrameStation. Subsequently, they assessed the platform through the **S**ystem **U**sability **S**cale (**SUS**) [Brooke 1996] to measure the usability, and the AttrakDiff questionnaire [Hassenzahl et al. 2003] to evaluate the user experience. These resources have already been extensively employed as evaluation methods for educational games [Geraldes et al. 2019, Cardoso et al. 2016, Tolentino et al. 2011].

The rest of this paper is organized as follows. Section 2 provides basic notions of GG. Section 3 presents the GG-based game engine GrameStation and how a user can build games using the tool. Section 4 details the activity applied in this experiment. Section 5 gives an overview of obtained results and highlights some of them. Section 6 discusses the main results of this experiment. Section 7 concludes the paper, discusses research directions, and indicates some future works.

2. Graph Grammar

GG is a formal language [Ehrig et al. 1997] that can be seen as a generalization of Chomsky grammars, replacing strings by graphs, or Petri Nets, with dynamic changes over the system topology and references between tokens [Ribeiro 2000]. In other words, it is a visual way of specifying systems. This language represents the states of a system as graphs and describes its events (transitions between states) with graph transformation rules. Graphs are structures essentially composed by vertices and edges usually represented by points and arrows, respectively. The GG definition used by GrameStation specifies a **type graph**, that declares the kind of the system elements (vertices and edges); a **start graph**, that indicates the initial configuration from which the rules can be applied; and a set of **rules** that change the current state (graph). A rule is composed of two graphs, the **L**eft **H**and **S**ide (**LHS**) and the **R**ight **H**and **S**ide (**RHS**), as well as a morphism that maps the LHS into the RHS, defining what should be consumed, preserved, or created during the rule application. Elements on the LHS that are related by the **morphism** must be preserved, while those that are not related must be deleted. On the other hand, elements on the RHS that are not in the morphism image must be created. The LHS expresses a condition for applying a rule and the RHS expresses a consequence of its application. For a rule to be applied in a state graph, its condition must be satisfied, that is, it must be possible to map each element on the LHS of the rule to an element of the state graph.

This mapping is called **match** and must respect the type of elements, as well as the source and target of each edge. Thus, the application of a rule changes the state graph by excluding the elements in the match image associated with the deleted elements and adding the elements that must be created by the rule.

Figure 1 illustrates the type graph and the start graph of the Pac-Man game as a GG. The type graph (shown on the left of the figure) declares the existence of Pac-Man, ghosts, fruits, places (grey dots), a counter (pink triangle), and the relations between these elements. The start graph (depicted on the right of the figure) shows one Pac-Man, one ghost, and three fruits in a 3x4 map of places, while the counter indicates that no fruits have been eaten yet.

Figure 1. Type graph (left) and start graph (right) for the Pac-Man game

The set of rules (Figure 2) includes: *Pac Move*, *Ghost Move*, *Pac Eat*, and *Ghost Eat*. The rules are represented by a pair of graphs linked by an arrow. In Pac Move rule, for instance, the LHS defines the condition to apply the rule: having a Pac-Man in a place that has a way to another one. The RHS, on the other hand, defines the consequence of this rule: the conection of the Pac-Man is removed from its original place and is restored at the next one.

Figure 2. Rules *Pac Move* (left, top), *Ghost Move* (right, top), *Pac Eat* (left, bottom), and *Ghost Eat* (right, bottom)

3. GrameStation

GrameStation is a GG-based tool used for creating and running games modeled according to this formal language. Since games are represented as GG, it also promotes the development of skills related to CT. These skills are developed both by the person who creates a game (specifies a GG) and by the person who runs a game (simulates a GG). Silva Junior (2020) detailed how concepts such as data representation, problem decomposition, abstraction, algorithms and processes, and parallelism are developed through the design and simulation of a game (GG). GrameStation is divided into three modules: Grame Builder, Grame Player, and Grame Explorer. These modules allow users to create, run, and find games, respectively.

When specifying a GG in GrameStation, it is possible to import external resources such as images and sounds or use the original resources available in the tool. The user should also create the type graph, which corresponds to a declaration area in the platform, and the start graph, which shows the organization of the game at the beginning. Finally, the rules which define the actions of the game should be created. All games begin with an empty type graph and an empty start graph. In order to specify elements in the type graph, users must define properties such as name, appearance, color, width, height, and rotation (on the x, y, and z axis). In the start graph, users can create new elements by instantiating the ones defined in the type graph. To do this, type and name of the element must be specified. Finally, to specify rules, the user must indicate whether the element to be added will be preserved, created, or deleted by the rule, as well as its type and name. In all cases, source and target are also required if the element is an edge, and the position (coordinates) if the element is a vertex.

To play a game (using the Grame Player module), the user can select, map, and apply the specified rules during the execution. When a rule is selected, the LHS and RHS graphs are shown, and the user should find a match by clicking on LHS elements and then their corresponding elements in the state graph. Besides, GrameStation signals when a match is correct or incorrect.

4. Methods

A total of 19 individuals (with and without prior experience in GG) were invited to participate in this second experiment, of which 11 had previously participated in the first experiment [Silva et al. 2022] and agreed to participate in this one as well. Participants were instructed to create a game using GrameStation and then answer questions regarding their experience with the platform. Two questionnaires were administered: the SUS [Brooke 1996] to collect data on the platform's usability, and AttrakDiff [Hassenzahl et al. 2003] to gather data on the user experience. The activity was remotely, asynchronously, and individually conducted. All participants had a CS background: 1 doctoral student, 1 master's graduate, and 17 undergraduate students. The students were enrolled in three different courses (System Analysis and Development, Computer Science, and Computer Engineering) from four different higher education institutions in the state of Rio Grande do Sul: Universidade Federal de Pelotas, Universidade Federal do Rio Grande do Sul, Universidade Federal do Pampa, and Instituto Federal Sul-rio-grandense. GrameStation is not restricted to a specific target audience, but we chose to keep the same group of people with a CS background from the first experiment for the second stage of the analysis to compare the results at the end of both experiments. Additionally, we assumed that the participants, who have a CS background, could focus on the investigated issues in the current state of GrameStation, as they tend to be familiar with digital tools/platforms, graphs, formal languages, and understand actions through automated feedback. Before the activity, all participants read and signed the Free and Informed Consent Form (ICF), indicating that they decided to participate in the research of their own free will and are aware of the use of their data.

Among the 19 invited people, 6 had some previous experience with GG, so we divided the participants into three groups: a group composed of those who had previous experiences with GG (group 1) and two groups (with 7 and 6 people) composed of those who had no knowledge in GG (group 2 and group 3, respectively). In order to determine which groups the participants would fit into, an initial meeting was conducted. During this meeting, we provided an overview of the activity and asked the participants if they had any prior experience with GG. After the meeting, 2 people from each group dropped out of participating in this second phase, even though they had participated in the first experiment. In this experiment, group 2 was presented with a video introducing the main concepts of GG and an example of creating a game modeled in this language, in order to facilitate the understanding of these concepts. Group 3, on the other hand, received a tutorial briefly describing the main concepts of GG and presented figures of a game created as GG, but without explanations about its creation. We did this so that we could compare the impact of this little experience on creating games in GrameStation. Additionally, all three groups received a document with a briefing for creating the game – that is, instructions on what should be created – and an explanation of the basic actions of GrameStation (such as how to create an element in the type graph and how to edit an element). All three groups did the same activity (created the same game and answered the same questions) and had 16 days to complete it.

Participants were tasked with creating a simplified version of the classic Pac-Man game from the 80s, which involves eating fruits to gather points on a map while controlling the main character and avoiding four ghosts. The instructions for the activity required participants to include at least one Pac-Man character, one ghost character, places on the map, and an objective (a fruit to eat or an object to catch). Additionally, they should create at least one movement rule, one interaction rule between Pac-Man and the ghost, and one interaction rule between Pac-Man and the objective. The Pac-Man game was chosen due to its popularity as most participants would already be familiar with the mechanics of the original game and could analyze and compare how it can be modeled in GrameStation. After completing the activity, participants were asked to upload their created games for evaluation. Each game was individually evaluated based on five components related to the minimum requirements: 1) type graph; 2) start graph; 3) movement rule of Pac-Man/ghost; 4) interaction rule between Pac-Man and ghost (typically leading to defeat); and 5) interaction rule between Pac-Man and the objective (typically leading to victory). Each component was scored as follows: 0 for not completing the requested task; 1 for partially completing the requested task; 2 for completely completing the requested task; 3 for creating something functional beyond what was requested.

Finally, each participant evaluated the experience through a form[1] divided into three categories: demographic information, usability, and user experience. For the demographic information category, we used the MEEGA+ (Model Evaluation of Educational Games) questions [Petri et al. 2016], only replacing the "Subject" field with the "Semester" field, which indicates the current semester of the students. The SUS [Brooke 1996] questionnaire was used to measure usability. It is a simple 10-statement

[1] Available (in Portuguese) on: https://curt.link/0vieCZ.

scale that provides a global view of subjective usability assessments, returning a result between 0-100. Each item is scored from 1 to 5 points, with higher scores indicating greater agreement or disagreement with the statements. For statements 1, 3, 5, 7, and 9, the higher the score, the greater the agreement. For the remaining statements, the higher the score, the greater the disagreement. For example, for statement 1, “I think that I would like to use this system frequently”, a score of 5 indicates strong agreement, while for statement 2, “I found the system unnecessarily complex”, a score of 5 indicates strong disagreement. We followed the original scoring rules of the SUS, only translating the statements into Portuguese and replacing “the system” with “GrameStation” where appropriate.

As for the user experience category, the AttrakDiff user experience form was used. AttrakDiff [Hassenzahl et al. 2003] is a questionnaire composed of 28 pairs of antonyms that analyze the user experience based on 4 dimensions: Attractiveness (ATT), Pragmatic Quality (PQ), Hedonic Stimulation Quality (HQ-S), and Hedonic Identification Quality (HQ-S). The ATT measures the global judgment of the system and the user’s perception of quality. The PQ addresses the usability, functionality, and usefulness of the system, to determine if users are achieving the goals using it [5, 15]. HQ-S emphasizes psychological well-being, in addition to evaluating characteristics related to innovation, whether the system arouses interest or has interaction, content, and presentation characteristics. HQ-I analyzes product identification in a social context [5, 15]. All questionnaire items were used according to the original model, only translated into Portuguese. The response format for each item is based on a 7-point Likert scale [DeVellis 2003], with alternative responses ranging from -3 (“strongly disagree”) to 3 (“strongly agree”). We also added a section for participants to discursively answer the questions “What did you like most about creating games on GrameStation?” and “What limitations did you notice in the tool? What could be improved?” as well as an area for general comments. Finally, we asked participants to upload the game created on GrameStation.

5. Results

Among the 19 invited students, 16 finished the activity, and we considered all their answers for the analysis. Regarding demographic information, 11 (68.75%) were male and 5 (31.25%) were female. Fourteen (87.5%) were aged between 18-28 years old, while 2 (12.5%) were aged between 29-39 years old. The rest of the information, such as institutions, courses, and semesters, as well the participants’ familiarity with digital and non-digital games are also available[2]. Figure 3 (left) shows the score assigned to each component[3]. P1-P6 (in blue) represents the participants of group 1; P7-P11 (in yellow) represents the participants of group 2; P12-P16 (in pink) represents the participants of group 3. In group 2, P9 did not comply with what was requested in items 2 (initial graph creation), 4, and 5 (creation of rules), while P10 did not comply with items 3-5, i.e., did not create any rule. Regarding group 3, P13 only partially completed item 1 (type graph creation); P15 did not comply with items 3-5 (creation of rules); and P16 did not send us the game. The figure also illustrates, on the right, the chart representing the average obtained by each group in each component. Group 1 completed all the requested requirements, while group 3 had more difficulty in creating rules.

[2]Complete demographic information charts available on: https://bityli.com/v2O67.

[3]The games created by the students have been evaluated and can be found at: https://wp.ufpel.edu.br/pensamentocomputacional/gramestation-pt/jogos/.

	1	2	3	4	5
P1	3	3	2	3	3
P2	2	2	2	2	2
P3	3	2	2	2	2
P4	3	3	3	3	2
P5	3	3	2	3	3
P6	3	3	2	3	3
P7	2	2	2	2	2
P8	3	2	2	2	2
P9	1	0	1	0	0
P10	1	2	0	0	0
P11	2	2	2	2	2
P12	3	1	2	0	1
P13	1	0	0	0	0
P14	3	3	1	2	1
P15	1	2	0	0	0
P16	0	0	0	0	0

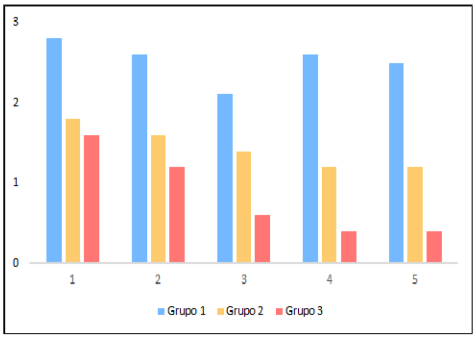

Figure 3. Score for each component assigned to each participant (left) and rating chart of created games (right)

Regarding usability, GrameStation had an average score of 75.4 according to group 1, on a scale between 0-100, while group 2 and group 3 had scores of 41.5 and 43.5, respectively. Figure 4 shows the average score of each group for each item of the questionnaire. Item 2 stood out, as groups 2 and 3 found GrameStation unnecessarily complex, particularly group 3. On the other hand, item 3 showed the opposite, as group 1 found GrameStation easy to use, while group 3 had more difficulty in this aspect. For item 4, groups 2 and 3 (with emphasis on group 3) mostly agreed that they would need help to be able to use GrameStation. In item 7, group 1 agreed that most people would learn to use GrameStation quickly, unlike groups 2 and 3. In item 8, groups 2 and 3 found GrameStation too complicated to use, unlike group 1. Finally, regarding item 10, group 3 was the one who most agreed that they would need to learn many things before starting to use GrameStation.

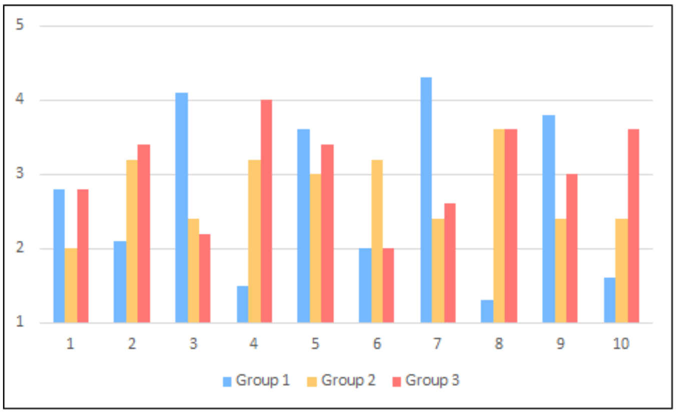

Figure 4. GrameStation usability chart

Figure 5 illustrates the average score of each group for each item (pairs of words) of the AttrakDiff questionnaire. We observed opposite results between group 1 and groups 2 and 3 for items 1, 2, 5, and 6 of the PQ category. In the HQ-S category, all three groups presented positive scores for items 16, 17, 18, and 21. However, for item 20, groups 2 and 3 found the tool to be challenging. Finally, in the ATT category, we noticed that group 1 had more positive results compared to the other groups for items 22, 24, and 27.

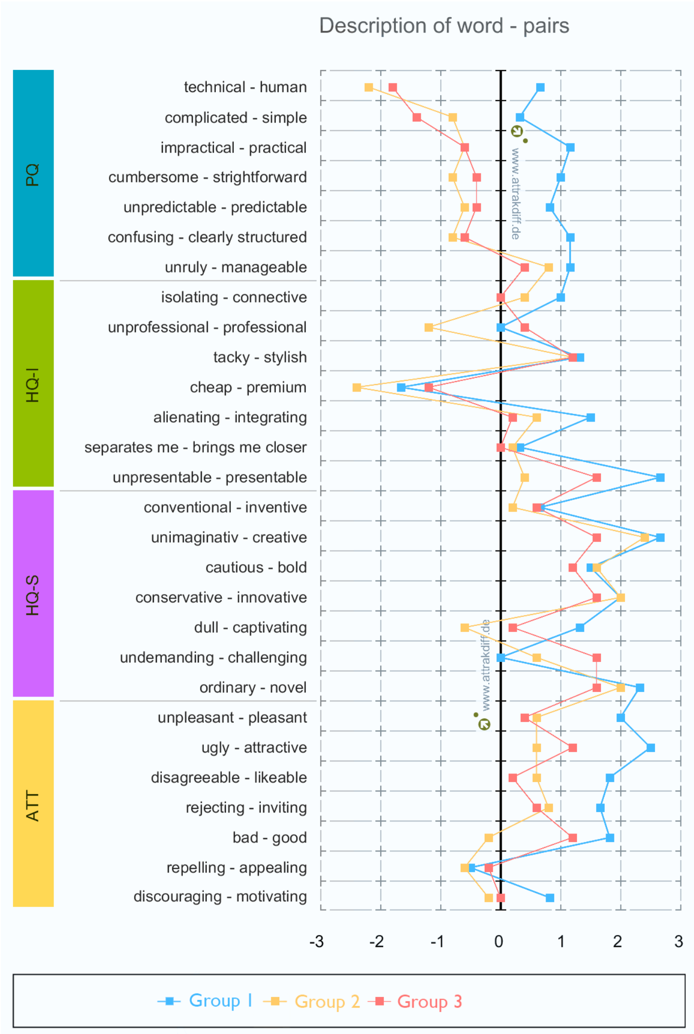

Figure 5. GrameStation user experience chart

Regarding the discursive questions[4], in Q1, 8 participants said they liked the visual aspects of GrameStation, such as the variety of colors and icon options, the soundtrack, and the possibility to create and customize elements in detail. Additionally, 2 participants found GrameStation intuitive and the creation of elements simple. Two other participants mentioned that they enjoyed the challenge of thinking in a different way when creating games.

In Q2, regarding what could be improved on the platform, 7 participants made comments related to interface improvements (such as a back button); 3 participants commented on the repetitive flow of the tool for creating items – often basic items (1 participant also commented about this in Q1); 4 participants commented about difficulties related to creating rules. Finally, 6 participants suggested the inclusion of tutorials, documentation, pop-ups, and example games as a way to help new users with the tool.

[4]Available (in Portuguese) on: https://curtlink.com/naMYA6M.

6. Discussion

We divided the participants into three groups according to their knowledge of GG as we had hypothesized that it would impact creating a game in GrameStation. Therefore considering this separation, the introductory video about GG seemed to make difference to group 2, as the responses of group 3 were more negative in all evaluations.

Regarding the games created by the participants, some design decisions were made due to a lack of knowledge or misunderstandings of the functioning of the GGs. For example, some participants duplicated the type that represents a place (home/position) in the game and all its relations, just to be able to link one place to another (Figure 6). However, this should be done using only one place type and an edge with a source and target in this same vertex. Additionally, some participants created the map in the type graph defining several types of places and relations between them without intending for them to behave differently (Figure 7). In fact, only one type of place and relation should be created, and this map assembly should be done in the initial graph by instantiating these two types. Figure 8 illustrates the type graph and initial graph of a participant of group 1 that successfully completed what was requested and created something functional beyond what was asked.

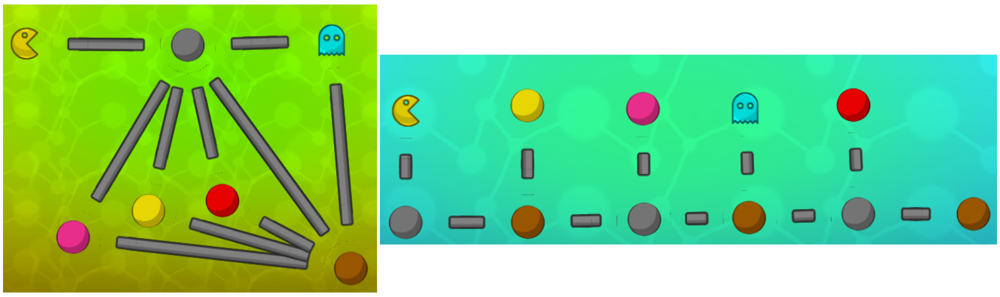

Figure 6. Type graph (left) and initial graph (right) of a participant of group 3

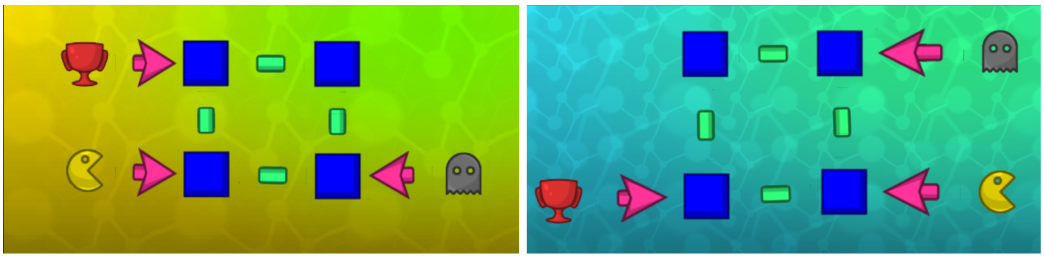

Figure 7. Type graph (left) and initial graph (right) of a participant of group 3

Concerning the usability category, items 3 and 7 revealed how easier creating a game in GrameStation was to group 1 (people with previous experience in GG). It contrasts with how difficult items 2, 4, 8, and 10 revealed that creating a game in GrameStation was to groups 2 and 3 (people without previous experience in GG). The score agrees with the suggestions left by the participants in the descriptive questions fields about the need of tutorials, documentation, or something similar to help new users in the platform. Most of these comments were written by participants in groups 2 and 3. Group 1's suggestions, otherwise, were related to technical aspects, such as the suggestion of NAC rules or comments about missing attributes.

Figure 8. Type graph (left) and initial graph (right) of a participant of group 1

Furthermore, we assume that group 1 found GrameStation simple, well-structured, user-friendly, and addictive, for example, precisely because of their familiarity with GG – as opposed to groups 2 and 3, who mainly responded negatively. Moreover, GrameStation is still under development, with several improvements planned to be implemented. Hence, suggestions related to the functional aspect of the platform were already expected. Despite the difficulties faced by groups 2 and 3, both seemed to agree on the quality of GrameStation in terms of its innovation, creativity, daring, and challenge. Lastly, in the HQ-I category of AttrakDiff, all participants agreed that the GrameStation was cheap. Cheap and premium have significantly different meanings in relation to "barato e caro" (in Portuguese). This pair of words is about how the platform feels, but due to the translation, may have been interpreted as the cost/price by participants.

Finally, considering the points raised by the participants in both experiments, the suggestions can be divided into three main points to be addressed/solved by the pedagogical agents. These points are related to: i) **assistance with the tool interface**, so that the agent can help users with questions regarding the GrameStation's usage mechanics; ii) the **addition of tutorials** to guide the steps of creating and running games in order to facilitate these actions; iii) the **presentation/explanation of terms related to the GG** to familiarize the users with the concepts of this formalism.

7. Conclusions

This article presented a usability and user experience test of GrameStation: a game engine based on GG. In this experiment, considering the creation of the games, participants in groups 2 and 3, who had no previous experience with GG, encountered greater difficulties with the tool. This corroborates the hypothesis raised about people with no experience finding it difficult to specify/understand a GG. In this sense, considering the performance (score of the created games) of group 1 and the usability and user experience evaluation, the GrameStation, in its current state, meets the needs of those who know GG. Additionally, although the number of participants in the activity was not significant, it was sufficient to identify improvements to be implemented. Future work will include increasing the number of participants in further evaluations.

Based on the result obtained from the two experiments, the next step in this work consists of implementing the pedagogical agents on the platform. After implementation, we aim to repeat the test with groups with similar profiles to group 2 and 3, as well as test the tool with other groups, such as people with no Computer Science background and children.

References

Brooke, J. (1996). SUS: A "quick and dirty" usability scale. *Usability evaluation in industry*, 189(3).

Cardoso, R. C., da Costa, V. K., Rodrigues, A. S., Tavares, T. A., Xavier, K. F., Peroba, J. A., Peglow, J., and Quadros, C. (2016). Doce labirinto: Experiencia de jogo utilizando interação baseada em movimentos da cabeça e recursos tangíveis. *XV Simpósio Brasileiro de Jogos e Entretenimento Digital*, pages 563–572.

DeVellis, R. F. (2003). Scale development: Theory and applications. *Sage London*, 13:0–176.

Ehrig, H., Heckel, R., Korff, M., Löwe, M., Ribeiro, L., Wagner, A., and Corradini, A. (1997). Algebraic Approaches to Graph Transformation. Part II: Single Pushout Approach and Comparison with Double Pushout Approach. In *Handbook of Graph Grammars and Computing by Graph Transformation. Volume 1: Foundations*. World Scientific Publishing Co., Inc.

Geraldes, W. B., Martins, E. R., and Afonseca, U. R. (2019). Avaliação da Usabilidade do Scratch utilizando o Método System Usability Scale (SUS). In *Anais da X Escola Regional de Informática de Mato Grosso*, pages 25–30. SBC.

Hassenzahl, M., Burmester, M., and Koller, F. (2003). Attrakdiff: Ein fragebogen zur messung wahrgenommener hedonischer und pragmatischer qualität. In *Mensch & computer 2003*, pages 187–196. Springer.

Nielsen, J. (1994). Usability Inspection Methods. In *Conference Companion on Human Factors in Computing Systems*, CHI '94, page 413–414, New York, NY, USA. Association for Computing Machinery.

Petri, G., von Wangenheim, C. G., and Borgatto, A. F. (2016). MEEGA+: An Evolution of a Model for the Evaluation of Educational Games. *INCoD/GQS*, 3:1–40.

Ribeiro, L. (2000). Métodos Formais de Especificação: Gramática de Grafos. *Escola de Informática da SBC-Sul*, 8:1–33.

Sentance, S. and Csizmadia, A. (2017). Computing in the curriculum: Challenges and strategies from a teacher's perspective. *Education and Information Technologies*, 22(2):469–495.

Silva, J. V., Junior, B. A. S., Foss, L., and da Costa Cavalheiro, S. A. (2022). A User Experience and Usability Test on Playing Games Specified as Graph Grammars in GrameStation. In *Anais do XXX Workshop sobre Educação em Computação*, pages 298–309, Porto Alegre, RS, Brasil. SBC.

Silva, J. V., Silva Junior, B. A., Cavalheiro, S. A. C., and Foss, L. (2021). Gramers: Agentes Pedagógicos para uma plataforma de jogos baseada em Gramática de Grafos. In *Anais do VI Workshop-Escola de Informática Teórica*.

Silva Junior, B. A. (2020). GGasCT: Bringing Formal Methods to the Computational Thinking. Master's thesis, Universidade Federal de Pelotas, Pelotas, RS, Brasil.

Silva Junior, B. A., Cavalheiro, S. A. C., and Foss, L. (2021). GrameStation: Specifying Games with Graphs. In *Anais do XXXII Simpósio Brasileiro de Informática na Educação*, pages 499–511, Porto Alegre, RS, Brasil. SBC.

Technologies, U. (2022). Unity real-time development pltaform – 3D, 2D, VR, and AR engine. Available at: `http://unity.com/`. Accessed: 2022-03-08.

Tolentino, G. P., Battaglini, C., Pereira, A. C. V., Oliveria, R. J. d., and Paula, M. G. M. d. (2011). Usability of Serious Games for Health. In *2011 Third International Conference on Games and Virtual Worlds for Serious Applications*, pages 172–175.

Wing, J. M. (2006). Computational Thinking. *Communications of the ACM*, 49(3):33–35.