Game Mechanics Generation. A Work In Progress

Jorge Ruiz Quiñones[1] and Antonio J. Fernández-Leiva[1]

University of Málaga Department Lenguajes y Ciencias de la Computación, Universidad de Málaga, Andalucía Tech, Málaga

Abstract. As part of a more general research, this paper presents a work in progress experiment for generating videogame mechanics and rules in a fully procedural way. A practical application of the detailed theoretical approach is also detailed as an experiment based on a genetic programming algorithm that allows to generate new video game specifications fulfilling the most important aspects highlighted by the game designers. To meet those requirements, we take advantage of the potential of the new features included in the last version of Xml-Based Video Game Description Language (XVGDL), termed XVGDL , and the corresponding advances in the XVGDL Game Engine (XGE). Taking advantage of both tools, the execution and the evaluation methods are about to be included in the experiment making the process to be performed without any human intervention. We expect to have new videogame specifications as the output of the process, created according initial requirements and designers' needs, with a minor effort of programming.

Keywords: Game Mechanics · Game Generation · Video Game Description Language · Video Game Engine · Video Game evaluation.

1 Introduction

This work in progress is framed into a more general research in the field of video game definition languages (VGDL) and could be considered as a particular case of procedural content generation (PCG) (although it will be focused on non-visualization components). Applying different computational techniques for generating game content directly related to the graphics or visualization–such as maps, levels or game components–is one of the most common approaches. In this research, we are focused as well in other non-visual aspects of the game like game mechanics, game rules or game end conditions. After developing a new VGDL called Xml Video Game Description Language (XVGDL) and the corresponding Game Engine supporting XVGDL game specifications (XGE), at the moment, just intended to be a basic game engine to allow to play XVGDL game specifications, the research is currently in a experimental phase in which some experiments are being performed to demonstrate and take advantage of the XVGDL and XGE features.

The interested reader is referred to [17], where more details and the advantages in usingXVGDL over other existing approaches are widely exposed.

At this point of the global research, we can highlight the following aspects:

- XVGDL is a complete VGDL with powerful features to take advantage of and contributing some improvements compared to other existing VGDLs implementations.
- XGE is a complete functional game engine with research purposes that allows to execute XVGDL specifications and provides great feedback to evaluate games' specifications.
- XVGDL and XGE are already implemented and presented. Used together, they are a powerful tool to develop real experiments that can definitely contribute during the game design phase, providing the game designers useful information and feedback during the game play and after game executions.

The new step in our research path is, using both XGE and XGE features, to provide a software that will be able to generate game mechanics. As a general concept, that can be understood as generating *new* games as they will be potentially different games one from each other. This is actually a previous stage to generate full games automatically, including some other concepts like graphics, maps/levels definitions, game objects, etc ...

This work in progress presentation is structured as follows. In section 2, the state of the art is exposed including some of the most relevant related works and also explains more in deep the XVGDL and XGE features that allow to evaluate and optimize video game specifications. Following, section 3 and its different subsections explain in detail the approach that we are following to generate game mechanics and how the genetic programming algorithm is developed to help on the main goal of the experiment. In particular, in subsection 3.3, the practical experiment approach is described including the requirements, the algorithm implementation and the expected outputs in the latest subsections. In subsection 3.6 we analyze then the expected outputs and how this experiment can contribute to the community. Last section 4, finally presents the conclusions and future work, mentioning the following steps to complete the global research project and the aspects to continue improving XVGDL and XGE .

2 State of the Art

Bio-inspired algorithms and, in particular, genetic programming (GP) has been widely used in many different research and engineering areas, including lots of many real examples and applications of this technique. Studies and researches in the computer games (videogames) field are not an exception in this sense, and we can find also existing works that take advantage of the GP features.

Focusing on computer games field, GP has been applied to agents [6, 11, 1], modeling [8, 4], machine learning [12, 19] or procedural content generation [5], to mention some of the existing works. This prolific production demonstrates that GP applied to any different area or research field in games or computer videogames is a really promising and powerful technique to proceed with experiments of different nature (different areas/sub-areas) and get an interesting and valuable output to be analyzed.

Getting deeper into game mechanics sub-area inside games research, many recent publications about this topic can be found, but at the moment, it is difficult to find works directly related to game mechanics generation. Some of the most interesting works in this particular area are related to game mechanics modelling and how they affect somehow to the players [2, 16, 14, 7] and also works more focused on practical applications like [15, 21, 20, 18]. Last, worth to mention [10] as a particular important contribution in the scope of our research, exposing tools to manage dynamics in n-player games of different nature.

As stated, there are many different approaches that contribute so much to the community, although it seems less common to find game mechanics (game design) generation approach using Genetic Programming (GP). As an example, [3] is focused on generating card games. This is a promising study to validate the approach of trying generating different kind of games using GP. Applying genetic programming to this particular field, would be a nice new contribution of this work.

3 Generating Game mechanics

As stated in the previous chapter, Game mechanics in general represent a very interesting and active game component to study. The research on game mechanics can be done from many different approaches, ranging from theory to practical applications. As aforementioned, we can not easily find works that are fully focused on game mechanics generation automatically and, as a consequence, there are just a few approaches to generate complete games, as an example [13]. We consider this aspect as a core point to finally be able to generate complete games in an procedural way, which is also in our research.

The main reasons and barriers that we find in order to develop a game mechanics generator include the following points:

- There is no standard VGDL that allows to extend the research in this and other related fields. This makes the advances to be so slow and so coupled to a concrete solution.
- There is no current approach that allows to test and experiment for games of different nature. The point here also is that the existing works in this field are heavily tied to a concrete example or implementation of a game, making it difficult to be extended.
- Evaluating the computational creations is not a easy task to be performed in a procedural way, and that is a key part of the whole process. A reliable way to evaluate games is needed to automate the game generation process and, in particular, the game mechanics generation.

The lack of having an standard way of describing game specifications or in particular, game mechanics, is in our opinion, the most important barrier to cross. In this sense, the publication of XVGDL has been great contribution as the majority of those problems, making possible to extend the research in fields like

game mechanics, without focusing on other points, like describing or specifying a game, an example to use, a game engine to execute, etc...

With previous experiments and researches, it has been demonstrated that XVGDL and XGE are a really useful tools to explore and can contribute so much with their potential and, with this new work, we want to give a new step forward in this field, making possible to create mechanics and/or games, taking advantages of the mentioned researches. In the next sub-section, the process of generating game mechanics using an XVGDL approach is fully detailed.

3.1 The XVGDL approach to generate game mechanics

Given the ability we have to specify very different parts of a game at the content, visualization or mechanics level, this work in progress presents a procedure that allows to generate parts, or even complete, videogames specifications using XVGDL . Chasing the idea of applying a genetic programming (GP) technique and taking advantage of the XVGDL nature, an specification can be easily interpreted as a tree, by the inherited representation of an XML itself. At first, it seems a feasible solution to apply GP to find new solutions (specifications) that represent new games.

For our case of study in particular, the XVGDL has by definition an structure which is really likely to be applied to a GP approach. XVGDL (based on XML language) can be easily transformed into a tree representation. Also, different nodes or even complete branches can be defined so they can be fit perfectly with the GP definition, being used in the different GP algorithm phase.

Being XVGDL a concrete implementation of the eXtensible Markup Language (XML), it can therefore easily be interpreted as a tree structure. The Document Object Model (DOM) is the foundation of XML documents have a hierarchy of units called nodes. DOM, is a way of describing the nodes and the relationships between them. A DOM document is a collection of nodes or pieces of information organized in a hierarchy.

If we interpret an existing specification of a game in XVGDL language taking into account these mentioned characteristics, we could easily represent the specification in the form of a tree and, in addition, separating the XVGDL nodes theoretically and by concepts as indicated in the Figure 1.

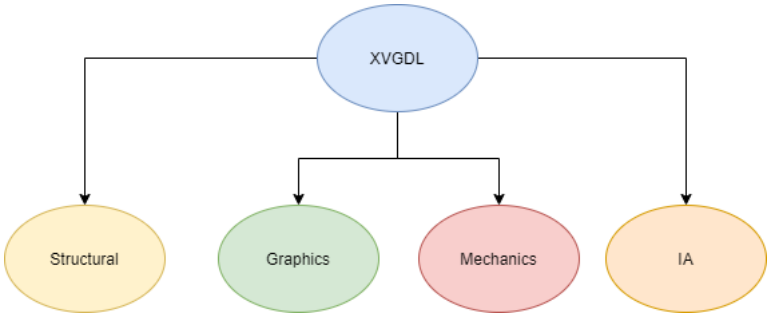

Fig. 1. Basic composition of an XVGDL specification represented as a tree.

As it can be appreciated in the Figure 1, all the information of a specification of a game in XVGDL would be lower in the hierarchy of each of the main nodes represented in the image. In addition to the main parent node, at the first level of the hierarchy, the detail of each of the levels represented is specified below:

- Structural Information: At this level, all XVGDL nodes related to the structure of the game, its maps, levels, screens or its controls, as well as the general properties that are established, are represented.
- Graphic Information: From this level, all XVGDL nodes related to graphic aspects would hang. We talk, for example, about details at the renderer level, the bitmaps associated with each of the game objects or the screen layout.
- Mechanics Information: All XVGDL nodes related to game mechanics. This node would be much more extensive in depth including game rules, end conditions, game states, the objects present in the game and events.
- Artificial Intelligence (AI): Considering it has importance enough to have its own node. AI sub-tree includes all the information and specifications related to agents and AI applied to any object in the game. It also includes all different AI implementations not directly related to an object, but affecting somehow during game play. As an example, an algorithm implementation to set the weather conditions during the game play or the game difficulty according to the player conditions and behaviour.

3.2 Representation of XVGDL elements in a tree.

We have seen, in general terms, how we could conceptually group XVGDL elements in four different dimensions. From these nodes, let's call them `main`, in general the intermediate nodes will represent concepts (existing elements/tags in XVGDL) as well as their properties. To complete the tree, the leaf nodes will represent specific values for each of the properties of the parent node. Each leaf node will have a value in relation to the parent node it represents.

As an example, in Figure 2, the representation of a bitmap element defined in XVGDL is detailed from the following code:

```
1  <bitmap id='PlayerRunning' file='/bitmaps/player-running.png' />
```

Fig. 2. XVGDL representation of nodes with attributes and values.

In Figure 2, the blue `Bitmap` node represents an element of the XVGDL . The red nodes named `id` and `file` represent properties of the parent node. The leaf nodes, in green, represent specific values of these properties.

Following the same color code represented in Figure 2 (blue for components, red for attributes and green for values), another bit more complex case is represented in Figure 3, which shows graphically the tree representation of the following XVGDL code: :

```
1  <rule name="eatSmallDot" type="collision">
2    <ruleAction objectName="pacman" result="score-up" value="100" />
3    <ruleAction objectName="smallDot" result="disappear" />
4  </rule>
```

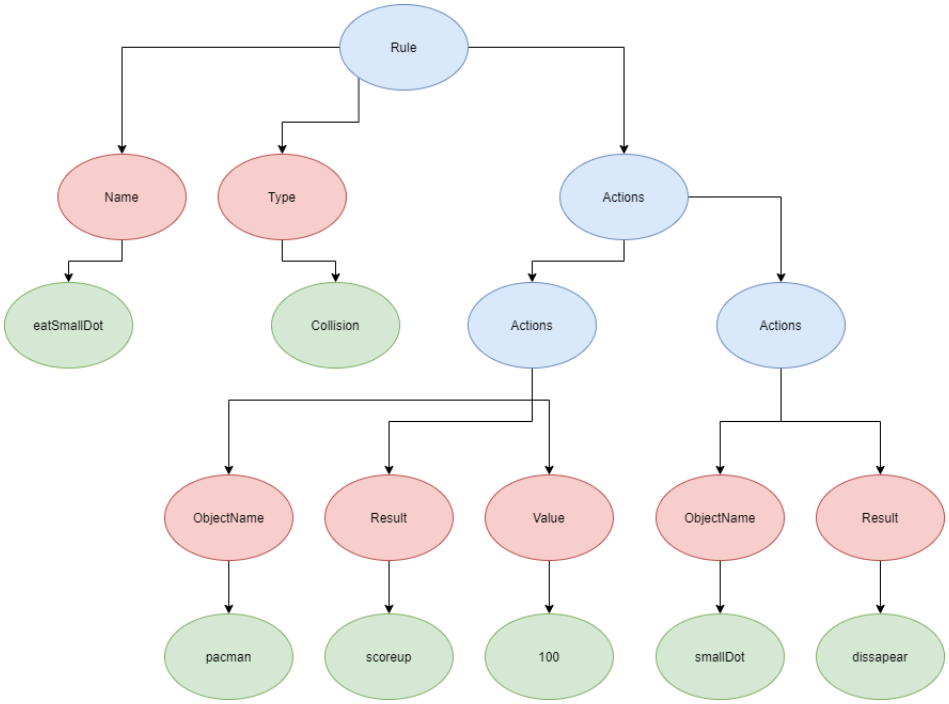

Fig. 3. XVGDL representation of complex nodes with sub-trees.

Following this type of representation and given the four main nodes from which the information hangs (structural, mechanical, IA and graphic), we can represent an entire specification in the form of a tree and work with it in a future genetic programming experiment.

3.3 Designing the experiment: Generating Game mechanics

Supporting the underlying research, a practical experiment was conducted to demonstrate the potential of this approach and also to be able to extract valuable information about the game design in a particular case, using an specification of the Pacman [9] videogame as test bench.

As aforementioned, the main idea of this global research is to generate full games from scratch (or at least, with a minimum effort of development), but at this point, we are focusing on the game mechanics and game rules. That means that the whole game generation is not covered.

This approach will allow us to create new game specifications including new mechanics. Practically, this means we can create new games. Not only that, but also if we apply the same solution to all the elements in the XVGDL tree representation, is easy to extrapolate it to different nodes in the tree, that is, covering not only mechanics but also other game components.

As discussed above, an algorithm can be perfectly applied to any part of the specification. As a future opportunity to explore, it would be a matter of deciding what elements we are going to consider in a future experiment to allow applying the same idea (GP approach) to generate other different videogame specification components. With the theoretical approach, the representation and division that we have detailed in the previous sections, we can easily proceed to other game components, being important in this particular work to limit the generation for the experiment. We have identified the components or sections where the algorithms can be applied, in such a way that we can complete the game generation, dealing with the following points globally or individually:

- Generation of game mechanics (Main mechanics node), covered by this work making possible to generate different games with a base set of components.
- Generation of game's components at a visual level (Main graphics node), covering all generation of graphics and visual components of a game.
- Generation of game's levels and/or maps (Main structure node), automating the creation of new game maps or levels.

3.4 Requirements

To carry out the generation of game mechanics we should have the following elements:

- A seed, that is to say, a base version of a game, from which we could generate the part that interests us. It could be as basic as we want, thus being the more complex generation process, or we could have a seed in which the structural and graphic information is defined and is immutable, and we apply the generation only at the level of nodes related to mechanics.
- A repository containing an initial set of rules, termination conditions and events.

At this point we need what we call a Primary Data store. This is important in order to make sure the generated games (experiment outputs) will be correct and valid in terms of game specifications. That means that the games will have sensible rules and mechanics, end conditions to make sure the game will finish and some basic definitions, assuring the game can be executed properly.

So, the main requirement now before starting the generation experiment will be to make sure we have a data store, keeping an initial set of rules (mechanics) and end conditions, as well as basic game definitions. Taking advantage of the current development, of XVGDL and XGE , we have created a simple process that fulfills the data store as needed. Different combinations of all possible values for rules and end conditions are performed in order to have the first base set of rules. As aforementioned, basic game specification (for Pacman in this case) has also been used to store a base game definition which will be evolved to a variant of the original video game using the GP approach. To make sure the data store is accessible from different sources, at the moment, data is stored using three different sources of information, a relational data-base, a local data set (in files)

and a secured access to a S3 bucket from AWS. Figure 4 expose graphically this procedure and the communication between different stacks involved in the videogame mechanics generation process.

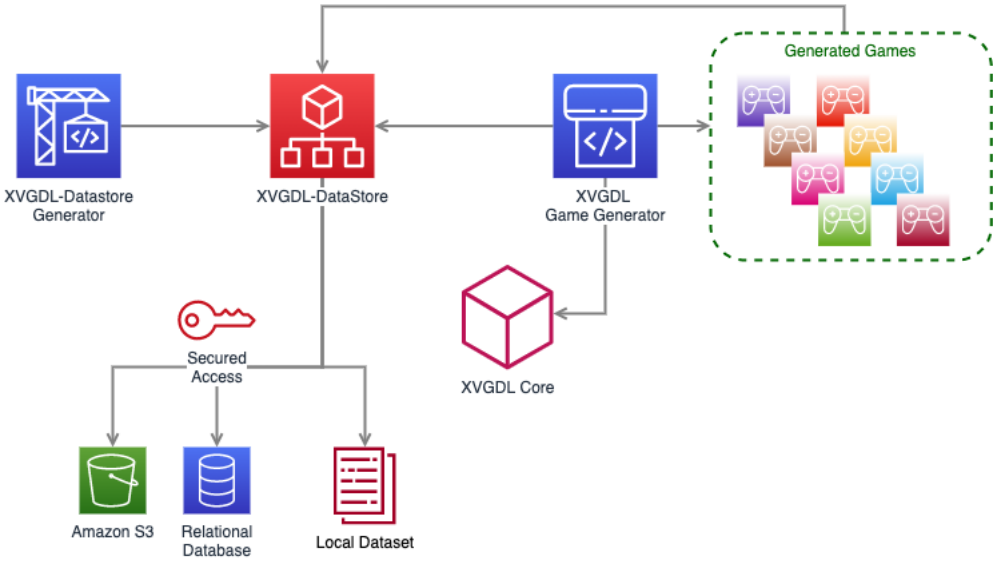

Fig. 4. The XVGDL Data store generation.

3.5 The algorithm

As already explained, the implementation approach is to use GP, implementing an algorithm that, starting from a seed, will find for different *interesting* videogames' specifications, according to previously defined requirements. The initial seed can be already described (configured) or even can be generated from the aforementioned repositories, at last, it will be the base game to evolve. The base seed will be slightly modified to make sure we have an interesting population to start the algorithm. This is performed using a simple software in charge of generating new values for the base specification defined nodes. Once the initial population of specifications are obtained, those must be transformed into the tree form, as detailed in sub-section 3.2 making possible then to apply the main steps defined by a genetic programming algorithm. In a general way, we have defined the following operations:

– **Selection**: We would apply an evaluation function to each of the solutions found. Is part of this experiment to define a fitness function to evaluate each possible solution, focusing in parameters that can be defined by the game designers before starting the experiment. Aspects like wining/loosing the

game, time to complete, number of lives remaining, etc... will be considered to create the fitness function.
Crossing: From the population of specifications that we have at a given moment, the crossing of individuals could be done by exchanging leaf nodes (values) for certain aspects of the game, or directly exchanging sub-trees that represent a specific structure, including elements and their values.
Mutation: We could drastically change a complete branch of a tree from the elements that we have in the repositories or change leaf nodes (values) for distant values (within the allowed ranges).

From the crossover and mutation operations, we would have solutions that we could evaluate thanks to the fitness function that we define and applying the characteristics of XVGDL .

Somehow, even not starting from specific game requirements, we could have very different games visually, structurally and mechanically, always starting from some generic objectives to be evaluated, such as the total playing time, assessment of victory player, difficulty level, etc.

3.6 Expected results

The main output of this experiment will be different videogames' specifications that, starting from a base game and given initial designers' requirement, are valid and complete, in such a way that they are playable games, using XGE , the game engine interpreter of XVGDL game specifications. It is important to highlight here the following aspects.

- During the experiment, we will apply a fitness function to every single solution guided by the designers' criteria in such a way that we will finally have videogame specifications that will maximize the designers' initial requirements.
- The output, which will be a population of videogames, will finally represent game specifications that can end up in completely different games one from each other, as eventually, the rules and mechanics will be different for each solution.

It is important to point as well that this work is not oriented to find the best solution (best game specification in this case) as there is no such solution. The main idea is to find a set of solutions, fulfilling some initial requirements, making sure the games are playable (correct). A solution is not better than other, but the output will help designers to define and describe new game specifications, according the ones obtained for the final population. Even those final specifications could be then gathered into a single game specification, just taking into account the most promising game mechanics and rules from each of them.

We have covered game mechanics and rules but as the XVGDL is easily transformed into a tree, the whole game generation is a feasible future opportunity to explore.

4 Conclusions and future works

In this work, we are taking advantage of the already implemented and presented XVGDL and XGE . With this experiment, we demonstrate the important contribution and a bunch of possibilities open using those tools. This research path is a really promising branch, opening many different new opportunities for future researches.

Although GP is widely used in videogames research fields, generating game mechanics automatically will be a great contribution as there is not so much work in the library in regards to this particular sub-area.

During the paper we have also mention how this proposal can be applied to other game specfications' components, not only considering game mechanics. The process detailed here can be extended to maps/levels, AIs and other graphical components. In this sense, we consider this work as a core point to finally be able to generate complete games in an procedural way, which is also in our research path. This, or a similar proposal, can be used for future implementations to keep the same approach in generating graphics or structural components of a videogame, completing the full automatic generation.

The evaluation of these computational creations is a key factor to consider here, as we need to ensure that the solutions accomplish the designers' specifications and requirements. Previous works optimizing game specifications with XVGDL and XGE , cleared the way to perform these evaluations and allow us now to perform this work with key tools, ideas and providing a good level of reliability.

Algorithm performance and data will be analyzed once the executions will be finished. Hopefully, GP will be a good approach to get the expected output, although applying some other search techniques or approaches will be evaluated for future experiments.

We consider this work as a core point to finally be able to generate complete games in an procedural way. While this idea needs to be investigated yet, this work will contribute so much to the automatic complete videogame generation, together with the other approaches mentioned in the paper.

Acknowledgements

This work has been supported by Spanish Ministry of Economy and Competitiveness (MINECO) projects: EphemeCH (TIN2014-56494-C4-1-P), and DeepBio (TIN2017-85727-C4-1-P) – Check: `http://blog.epheme.ch` and `http://blog.deepb.io` –, and Universidad de Málaga, Campus de Excelencia Internacional Andalucía Tech.

References

1. Dworman, G.O., Kimbrough, S.O., Laing, J.D.: On automated discovery of models using genetic programming: Bargaining in a three-agent coalitions game. J. Manag. Inf. Syst. **12**(3), 97–125 (1996), `http://www.jmis-web.org/articles/307`

2. Ferro, L.S.: The game element and mechanic (GEM) framework: A structural approach for implementing game elements and mechanics into game experiences. Entertain. Comput. **36**, 100375 (2021). https://doi.org/10.1016/j.entcom.2020.100375, https://doi.org/10.1016/j.entcom.2020.100375
3. Font, J.M., Mahlmann, T., Manrique, D., Togelius, J.: Towards the automatic generation of card games through grammar-guided genetic programming. In: Yannakakis, G.N., Aarseth, E., Jørgensen, K., Lester, J.C. (eds.) Proceedings of the 8th International Conference on the Foundations of Digital Games, FDG 2013, Chania, Crete, Greece, May 14-17, 2013. pp. 360–363. Society for the Advancement of the Science of Digital Games (2013), http://www.fdg2013.org/program/papers/short01_font_etal.pdf
4. Frade, M., de Vega, F.F., Cotta, C.: Modelling video games' landscapes by means of genetic terrain programming - A new approach for improving users' experience. In: Giacobini, M., Brabazon, A., Cagnoni, S., Caro, G.D., Drechsler, R., Ekárt, A., Esparcia-Alcázar, A., Farooq, M., Fink, A., McCormack, J., O'Neill, M., Romero, J., Rothlauf, F., Squillero, G., Uyar, S., Yang, S. (eds.) Applications of Evolutionary Computing, EvoWorkshops 2008: EvoCOMNET, EvoFIN, EvoHOT, EvoIASP, EvoMUSART, EvoNUM, EvoSTOC, and EvoTransLog, Naples, Italy, March 26-28, 2008. Proceedings. Lecture Notes in Computer Science, vol. 4974, pp. 485–490. Springer (2008). https://doi.org/10.1007/978-3-540-78761-7_52, https://doi.org/10.1007/978-3-540-78761-7_52
5. Frade, M., de Vega, F.F., Cotta, C.: Breeding terrains with genetic terrain programming: The evolution of terrain generators. Int. J. Comput. Games Technol. **2009**, 125714:1–125714:13 (2009). https://doi.org/10.1155/2009/125714, https://doi.org/10.1155/2009/125714
6. García-Sánchez, P., Fernández-Ares, A., Mora, A.M., Valdivieso, P.Á.C., González, J., Guervós, J.J.M.: Tree depth influence in genetic programming for generation of competitive agents for RTS games. In: Esparcia-Alcázar, A.I., Mora, A.M. (eds.) Applications of Evolutionary Computation - 17th European Conference, EvoApplications 2014, Granada, Spain, April 23-25, 2014, Revised Selected Papers. Lecture Notes in Computer Science, vol. 8602, pp. 411–421. Springer (2014). https://doi.org/10.1007/978-3-662-45523-4_34, https://doi.org/10.1007/978-3-662-45523-4_34
7. Gauthier, A., Jenkinson, J.: Designing productively negative experiences with serious game mechanics: Qualitative analysis of game-play and game design in a randomized trial. Comput. Educ. **127**, 66–89 (2018). https://doi.org/10.1016/j.compedu.2018.08.017, https://doi.org/10.1016/j.compedu.2018.08.017
8. Geng, S., Hu, T.: Sports games modeling and prediction using genetic programming. In: IEEE Congress on Evolutionary Computation, CEC 2020, Glasgow, United Kingdom, July 19-24, 2020. pp. 1–6. IEEE (2020). https://doi.org/10.1109/CEC48606.2020.9185917, https://doi.org/10.1109/CEC48606.2020.9185917
9. Inc, B.N.E.: Pacman official, http://pacman.com/en/, accessed: 2nd September 2021
10. Letcher, A., Balduzzi, D., Racanière, S., Martens, J., Foerster, J.N., Tuyls, K., Graepel, T.: Differentiable game mechanics. J. Mach. Learn. Res. **20**, 84:1–84:40 (2019), http://jmlr.org/papers/v20/19-008.html

11. Liu, F., Serguieva, A., Date, P.: A mixed-game and co-evolutionary genetic programming agent-based model of financial contagion. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2010, Barcelona, Spain, 18-23 July 2010. pp. 1–7. IEEE (2010). https://doi.org/10.1109/CEC.2010.5586243, https://doi.org/10.1109/CEC.2010.5586243
12. Mariño, J.R.H.: Learning strategies for real-time strategy games with genetic programming. In: Smith, G., Lelis, L. (eds.) Proceedings of the Fifteenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, AIIDE 2019, October 8-12, 2019, Atlanta, Georgia, USA. pp. 219–220. AAAI Press (2019), https://www.aaai.org/ojs/index.php/AIIDE/article/view/5249
13. Michael Cook, m.: Games by angelina, http://www.gamesbyangelina.org/, accessed: 2nd September 2021
14. Moll, P., Frick, V., Rauscher, N., Lux, M.: How players play games: observing the influences of game mechanics. In: Eg, R., van der Hooft, J., Raaen, K. (eds.) Proceedings of the 12th ACM International Workshop on Immersive Mixed and Virtual Environment Systems, MMVE@MMSys 2020, Istanbul, Turkey, June 8, 2020. pp. 7–12. ACM (2020). https://doi.org/10.1145/3386293.3397113, https://doi.org/10.1145/3386293.3397113
15. Navarro, D., Sundstedt, V.: Simplifying game mechanics: gaze as an implicit interaction method. In: Gutierrez, D., Huang, H. (eds.) SIGGRAPH Asia 2017 Technical Briefs, Bangkok, Thailand, November 27 - 30, 2017. pp. 4:1–4:4. ACM (2017). https://doi.org/10.1145/3145749.3149446, https://doi.org/10.1145/3145749.3149446
16. Nguyen, J., Ruberg, B.: Challenges of designing consent: Consent mechanics in video games as models for interactive user agency. In: Bernhaupt, R., Mueller, F.F., Verweij, D., Andres, J., McGrenere, J., Cockburn, A., Avellino, I., Goguey, A., Bjøn, P., Zhao, S., Samson, B.P., Kocielnik, R. (eds.) CHI '20: CHI Conference on Human Factors in Computing Systems, Honolulu, HI, USA, April 25-30, 2020. pp. 1–13. ACM (2020). https://doi.org/10.1145/3313831.3376827, https://doi.org/10.1145/3313831.3376827
17. Ruiz-Quiñones, J., Fernández-Leiva, A.: Xml-based video game description language. IEEE Access **8**, 4679–4692 (2020)
18. Siu, K., Zook, A., Riedl, M.O.: A framework for exploring and evaluating mechanics in human computation games. In: Deterding, S., Canossa, A., Harteveld, C., Zhu, J., Sicart, M. (eds.) Proceedings of the International Conference on the Foundations of Digital Games, FDG 2017, Hyannis, MA, USA, August 14-17, 2017. pp. 38:1–38:4. ACM (2017). https://doi.org/10.1145/3102071.3106344, https://doi.org/10.1145/3102071.3106344
19. Szarowicz, A., Francik, J., Lach, E.: Reinforcement learning and genetic programming for motion acquisition. Int. J. Intell. Games Simul. **4**(1), 55–66 (2006), http://www.scit.wlv.ac.uk/OJS_IJIGS/index.php/IJIGS/article/view/5
20. Villegas, E., Labrador, E., Fonseca, D., Fernández-Guinea, S.: Validating game mechanics and gamification parameters with card sorting methods. In: Rocha, Á., Adeli, H., Reis, L.P., Costanzo, S. (eds.) New Knowledge in Information Systems and Technologies - Volume 3. Advances in Intelligent Systems and Computing, vol. 932, pp. 392–401. Springer (2019). https://doi.org/10.1007/978-3-030-16187-3_38, https://doi.org/10.1007/978-3-030-16187-3_38
21. Volkovas, R., Fairbank, M., Woodward, J.R., Lucas, S.M.: Mek: Mechanics prototyping tool for 2d tile-based turn-based deterministic games. In: IEEE Conference on Games, CoG 2019, London, United Kingdom, August 20-23, 2019. pp. 1–8.

IEEE (2019). https://doi.org/10.1109/CIG.2019.8848016, https://doi.org/10.1109/CIG.2019.8848016