NodeCrafter: Tool for stochastic modeling board games and RPG

João Josué Arroyo Silva[1], **Lucas Santos Policarpo**[1], **Ronan dos Santos Rosa**[1], **Igor de Oliveira Knop**[1]

[1]Departamento de Ciência da Computação – Universidade Federal de Juiz de Fora
36001-970 – Juiz de Fora – MG – Brasil

{joaojosue.silva,lucas.policarpo,ronan.santos}@estudante.ufjf.br,

igor.knop@ufjf.br

Abstract. Introduction*: Good game balancing provides fair and engaging gameplay experiences, ensuring equal conditions for players and memorable progression. However, many game creators lack the mathematical knowledge to analyze models based on stochastic mechanics, often relying on intuition and, at best, a large number of practical tests.* ***Objective:*** *This work presents NodeCrafter, a visual tool for modeling game subsystems, focusing on Monte Carlo simulations of stochastic processes for balancing design and evaluation.* ***Materials and Methods****: The proposed solution offers an online graphical modeling environment based on node flow that allows for the practical modeling, simulation, and sharing of rules. In addition to comparison with other available tools, the power of representing existing game systems is presented.* ***Results:*** *The results indicate that NodeCrafter can represent various mechanics common in tabletop and RPG games and presents itself as a promising tool for enthusiasts and studios.*

Keywords *Game design, game balancing, stochastic systems, Monte Carlo simulations.*

1. Introduction

In game development, whether digital or analog, the balancing phase is the pillar that sustains commercial viability and the depth of the user experience. More than a technical adjustment of rules, balancing is essential to ensure that no single strategy dominates the system and that all available options are perceived as viable and interesting, directly impacting value perception, long-term engagement, and the success of the final product [Schreiber e Romero 2021]. A well-balanced system transforms complex mechanics into immersive experiences, ensuring that victory or defeat is perceived as a consequence of player choices rather than design flaws.

Among the components with the highest risk to system balance, those based on randomness stand out. When implemented correctly, they enrich gameplay; however, they can also result in frustration or the perception of a loss of player agency [Paparistodemou et al. 2008, Zamith et al. 2020]. The impact of poor balancing is immediate: player abandonment and damage to the title's reputation, especially in highly competitive markets where fairness and pacing are rigorously evaluated.

Games can be modeled and balanced using a variety of tools, such as Markov chains [Zamith et al. 2020, Mocanu 2023], discrete simulation [Ourique et al. 2024], or

even analytical models. Considering that the game market consists of interdisciplinary teams that do not always have expertise in these areas, it becomes necessary to design tools that facilitate the observation and experimentation of stochastic scenarios. Visual modeling thus emerges as an alternative to democratize access to complex simulations, allowing designers to validate their hypotheses without the need for coding or dense formulas.

This motivation inspired the development of NodeCrafter [dos Santos Rosa 2023, Policarpo 2025], a web environment created for the modeling and simulation of tabletop game systems. However, the initial version of the tool focused on numerical and logical flows, presenting limitations in scenarios that require the manipulation of qualitative data, treated here as symbols. This article presents an expansion of NodeCrafter, developed to enable the representation of features that rely on symbolic abstractions, making the modeling of complex systems more accessible and allowing a wider variety of games to be represented.

2. Related Work

AnyDice[1] is a textual tool with its own syntax for modeling and calculating precise probabilities of dice rolls and interactions between game elements. Its interface allows for the creation of configurable and shareable scripts, enabling the analysis of the impact of different statistical variables in complex scenarios by presenting the system's behavior via Monte Carlo simulations.

The Monte Carlo method [Metropolis e Ulam 1949] uses simulations based on massive random sampling to find approximate solutions to complex problems where direct mathematical calculations would be unfeasible. By generating thousands of possible scenarios through statistical draws, it allows for predicting behaviors and probabilities based on the frequency of observed results.

The application of modeling for early balancing is explored by [Silva et al. 2023], who uses Machinations[2] to model the internal economy of the game ESG+P. Machinations focuses on modeling dynamic flows of discrete resources (*tokens*), and the case study by [Silva et al. 2023] demonstrates that using visual simulations before building physical prototypes allows for the identification of *deadlocks* and the efficient adjustment of costs. However, the authors report that representing multiple random possibilities can clutter the diagram, requiring the model to be simplified to maintain visual clarity. This limitation is exacerbated by the fact that the tool does not natively support traditional stochastic systems, such as dice rolls, which forces the designer to create complex resource flows to simulate simple random events.

On the other hand, the need for robust modeling tools is reinforced by the difficulty of maintaining balance in commercial games post-launch. A case study of the game Clash Royale is presented in [Filho et al. 2021], using data mining and logistic regression to analyze the *a posteriori* balance. The results indicate that, even with rigorous planning by the designers, only a small portion of the elements (22% of the cards) remained within ideal usage metrics, highlighting that minor rule changes can generate unpredictable systemic impacts if not validated by prior simulations.

[1]Available at `https://anydice.com`
[2]Available at `https://machinations.io`

Node-based visual modeling simplifies the representation of complex systems by encapsulating functionalities into blocks (nodes) connected by information flows (edges), eliminating the need for traditional programming and promoting accessibility and visual clarity. This approach is widely effective in shader development within Blender, Unity, and Unreal Engine. The Unity node editor for building shaders served as the inspiration for McDie [Engelstein 2021].

McDie[3] adopts a graph node architecture to process random events like dice rolls, result manipulation, and triggering new rolls. This allows the user to build models by interconnecting visual components to calculate the probabilities of complex systems frequently seen in RPG and board game settings. This makes system analysis more accessible to individuals unfamiliar with mathematical equations or programming, presenting system behavior results in a graphical and intuitive way to visualize certain influences and feedback loops. It also utilizes Monte Carlo simulation to present results and serves as a direct inspiration for NodeCrafter [dos Santos Rosa 2023, Policarpo 2025].

Thus, unlike Machinations, which focuses on internal economy and the flow of numerical tokens, NodeCrafter projects its scope toward stochastic simulation and the analysis of statistical distributions. Furthermore, although it inherits concepts from McDie, the tool distinguishes itself by being a fully web-based, open-source application that expands support for symbolic systems and hybrid operations. Finally, while AnyDice relies on a textual scripting language to calculate dice probabilities, NodeCrafter addresses similar purposes visually, utilizing a user-friendly interface based on node diagrams.

NodeCrafter[4] is an open-source web application (Figure 1) that can be expanded in the future. Its initial versions supported dice roll manipulations such as “explosions”, selection, and value addition [dos Santos Rosa 2023]. It has been used to model parts of popular RPG systems, such as Dungeons and Dragons and Tormenta 3rd edition, as well as some mechanisms of the board game Zombicide [Policarpo 2025]. This work presents an iteration that adds symbol pools and draws with and without replacement, allowing for the modeling of mechanisms such as enemy draws, skills, conditions, and rewards, or even increasingly popular games known as “bag-building”.

3. Methods and Materials

This work consists of applied research with a qualitative approach and an exploratory objective, based on direct observation. The modeling capability of the proposed tool is evaluated by its semantic accuracy in representing game systems available on the market. To this end, the game *Automobiles* (David Short, 2016) was used as a case study, selected because it operates under a *bag-building* logic that requires the complex manipulation of symbolic inventories.

New nodes are added to NodeCrafter and are explored herein. The models are also made available in the repository alongside the open-source code. The current version can be used directly from its *online* deployment[5].

[3]Available at `https://gengelstein.itch.io/mcdie`

[4]Available at https://github.com/ufjf-gamelab/node-crafter

[5]Available at: https://ufjf-gamelab.github.io/node-crafter/

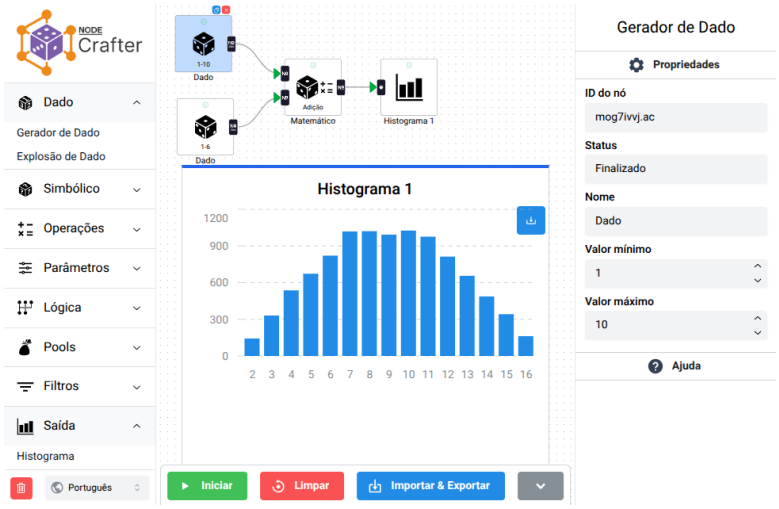


Figure 1. Nodecrafter Editor: nodes representing dice rolls and operations are visualized in a histogram as the result of a Monte Carlo simulation.

By default, all models and experiments executed in NodeCrafter utilize Monte Carlo simulations parameterized at 10,000 runs. However, users can adjust this total number of iterations according to their preferences.

4. Development

This work presents an addition of symbolic nodes to NodeCrafter. The following sections detail their applications.

4.1. The symbolic and numerical nature of games

In the *Dungeons & Dragons (5th Edition) Player's Handbook* system, there is a clear distinction between numerical variables, such as hit points and damage, and "Conditions", which define character states. Spells such as "Hold Person" require the selection of a humanoid target within a specific numerical range and attempt to impose the "Paralyzed" condition. This condition cannot be simply defined by arithmetic operations; it introduces rules that alter the game state, such as the inability to act, granting advantages to attackers, and the automatic application of critical hits.

In the game *Divinity: Original Sin*, for example, interacting with ice surfaces is a recurring challenge. If a character walks on ice, the system imposes a high chance of falling and losing a turn; however, avoiding movement to prevent an accident can be costly in terms of tactical positioning in combat. The system, nevertheless, does not only check if the character is walking on that surface but also verifies the presence of a "Slip Immunity" tag. The existence of this tag makes the character invulnerable to such falls, acting as a symbolic value that affects the character's interaction with the environment without directly manipulating numerical variables.

However, the distinction between these values does not imply that they operate in isolation; symbolic values often act as constraints on the final result of a numerical operation. The elemental type system in *Pokémon* exemplifies this hybrid interaction: when performing an attack, the system simultaneously processes type tags (Fire, Ice, Flying, etc.) and the numerical Attack and Defense attributes of those involved. The

resulting arithmetic calculation is governed by these symbolic variables, which function as contextual multipliers capable of amplifying, reducing, or nullifying the final damage. This motivates the implementation of symbols in NodeCrafter models.

4.2. *Pool/Deck Building*

In deck-building or pool/bag-building systems, symbolic elements are added to a deck or bag, and a specific number of elements are drawn to perform a game action. Modeling the frequency of these draws and observing probable and improbable scenarios is of paramount importance to the designer. To demonstrate the viability of this approach, *Automobiles*[6] will be used as a case study in the following sections, detailing how *NodeCrafter* maps the flow of these symbolic cubes through new nodes to model game mechanisms.

4.3. Automobiles Modeling

Automobiles is a board game where the central mechanic is the bag-building of a pouch that stores colored cubes. Each color serves as a symbolic representation of the different actions available to the player during a *Stock Car* race.

The foundation of symbolic operations is the "Symbolic Pool" node, which represents the receptacle where markers associated with a symbol (identifier) and their quantity are stored. With this node, it is possible to easily represent deck-building or bag-building systems, among many other symbolic data structures.

In the case of *Automobiles*, Figure 2a displays the node's properties, where a player's initial cubes are set: two light-gray cubes (4th Gear), five white cubes (3rd Gear), and five yellow cubes (Garage). It is also possible to simulate, as shown in Figure 2b, the outcome if the player chooses to purchase three different cubes: one dark-gray (5th Gear), one purple (Pit Captain), and one blue (Fuel Injector).

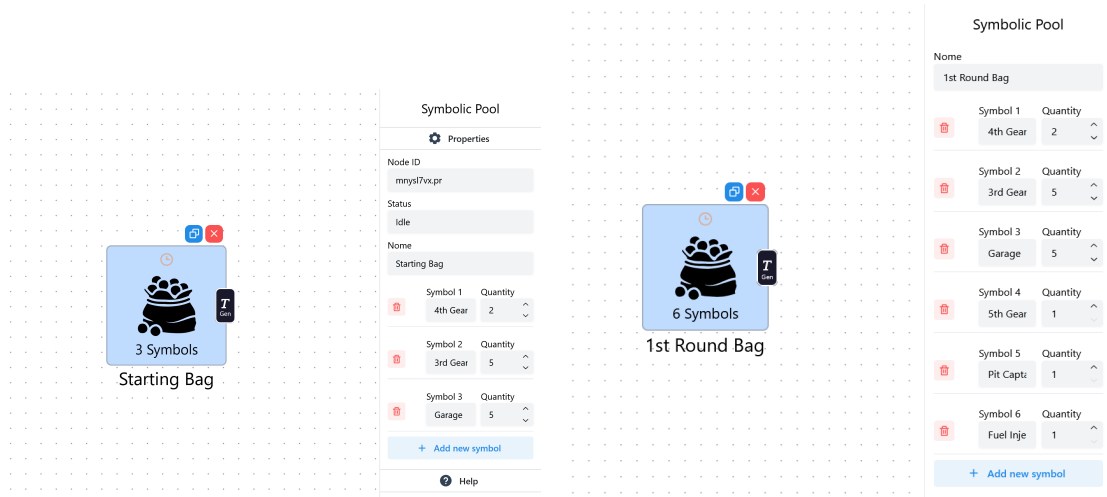
The draw is performed by connecting a "Draw Symbols" node to the "Symbolic Pool" node, as illustrated in Figure 3a, for a draw of seven cubes from the bag. To observe the distribution of the draws, a "Histogram" node is added at the end, where the Monte Carlo simulation will present the frequency of all possible sets of symbols (Figure 3b). By hovering over the bars, the designer can evaluate which sets are more or less frequent.

4.4. Converting symbols into values

For scenarios requiring the linking of arithmetic operations to symbolic variables, the "Numeric Value" node was added. Since each node features a symbol-to-value conversion table, this component enables the transformation of the same symbol across various parts of the model.

As an example, the movement potential of a player in a first turn was modeled for *Automobiles*. Considering gear cubes exclusively, a value representing the number of board spaces they allow the player to advance was assigned to each type, simplified by disregarding track layout or blocking, as illustrated in Figure 4a. This configuration allows for observing the frequency of reachable distances resulting from the draw (Figure 4b), offering insight into potential performance based on the composition of the player's bag.

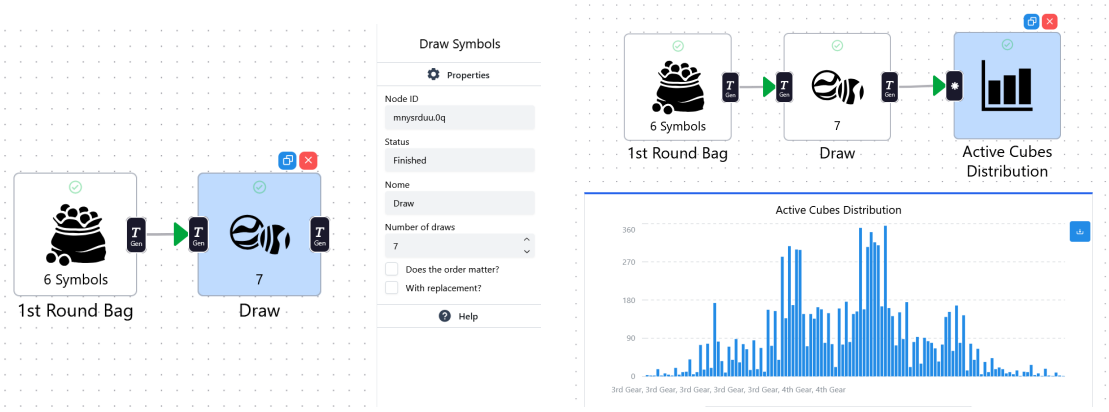
[6]Available at: https://boardgamegeek.com/boardgame/180680/automobiles



(a) Modeling the initial state of the bag in the game *Automobiles* using the Symbolic Pool node.

(b) Modeling the state of the bag after the first round of purchases.

Figure 2. Modeling the game *Automobiles*: initial bag state and subsequent draw distribution.



(a) Modeling the cube drawn from the bag.

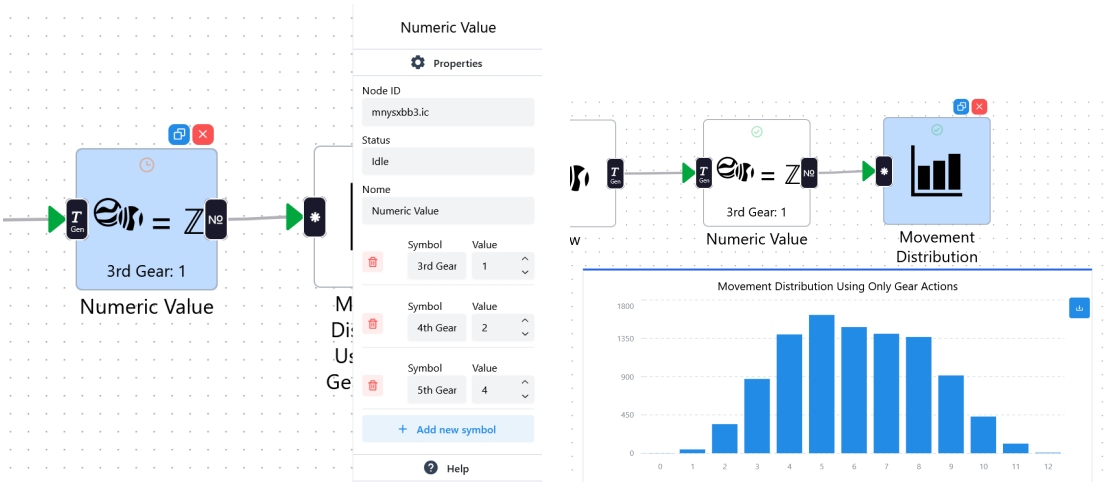
(b) Distribution of active cubes drawn using the Histogram node.

Figure 3. Distribution of active cube draws using the Histogram node.

4.5. Symbolic Set Verification

The verification of specific conditions is performed by the "Symbolic Conditions Check" node, which analyzes the set of symbols from a drawing and validates whether they meet the configured conditions, returning a Success or Failure value.

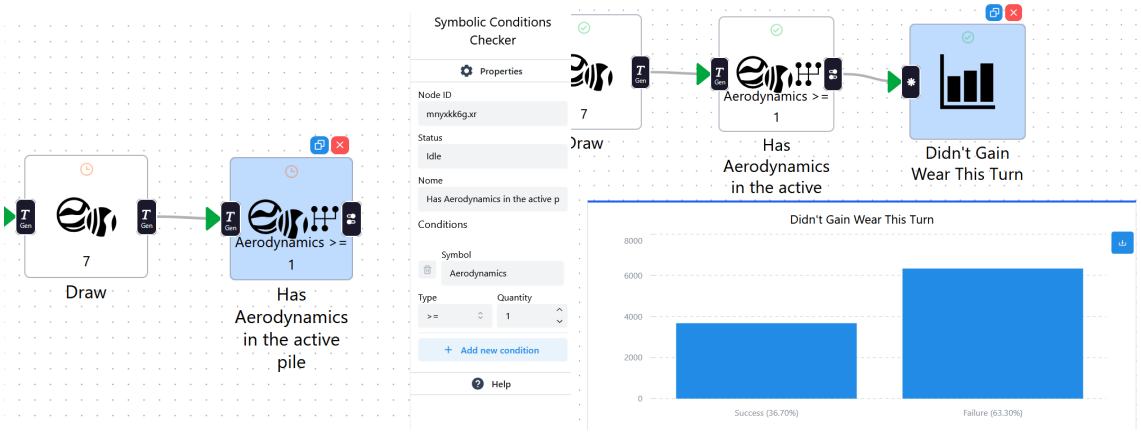
In the game *Automobiles*, after a player ends their turn, they receive a specific amount of wear cubes ("Wear") based on the color of the space at their current position and the presence of an opponent ahead. Since wear accumulation directly impacts the performance of future turns, protecting oneself with "Aerodynamics" completely prevents this wear gain at the end of the turn. The probability of being protected can be modeled by checking for the existence of this cube in the draw (Figure 5a) and observing the percentage of Successes in the Histogram (Figure 5b).



(a) The Numeric Value node allows for the creation of a value lookup table for symbols.

(b) Frequency of the resulting sets from a seven-symbol draw.

Figure 4. Assignment of numerical values to symbolic variables.



(a) Symbolic Condition Verifier configuration.

(b) Success or Failure distribution for avoiding wear gain.

Figure 5. Operation of the symbolic condition verifier for wear control.

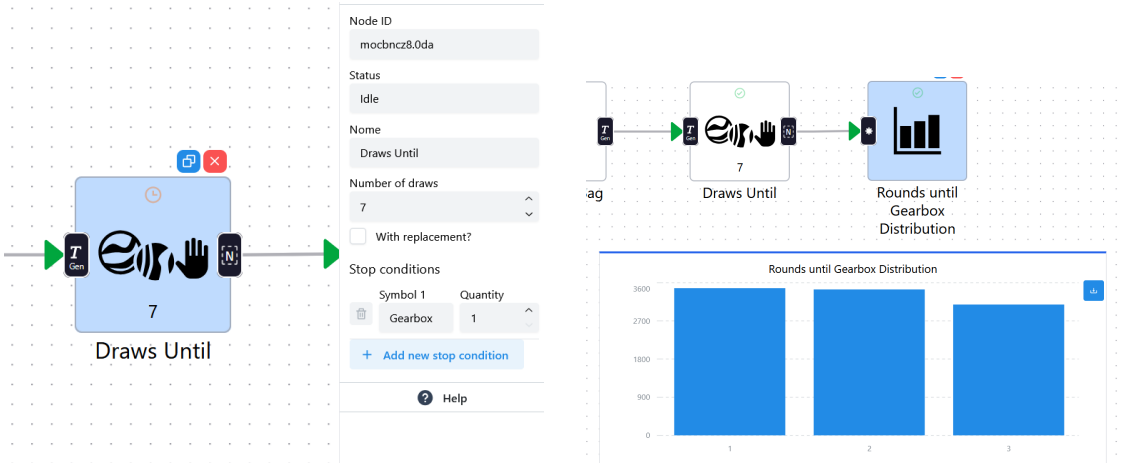
4.6. Draws with stopping conditions

Another very common type of modeling in games, especially those featuring "press-your-luck" mechanisms [Engelstein e Shalev 2022] such as *The Quacks of Quedlinburg*[7] and *Wonderland's War*[8], is a draw limited by a specific condition. This condition may be finding a particular symbol or reaching a certain number of symbols. NodeCrafter utilizes the "Draws Until" node, which defines a stopping condition and returns the number of draws required to reach it.

In matches of *Automobiles*, when a player is far behind the others, they can recover using specific actions like the Gearbox, which allows the player to move many light-gray spaces equal to their current position in the race. This is a very powerful action that serves

[7] Available at https://boardgamegeek.com/boardgame/244521/quacks
[8] Available at https://boardgamegeek.com/boardgame/227935/wonderlands-war

as a catch-up mechanism. We can model this using the "Draws Until" node to estimate the number of rounds needed to obtain the specific cube (Figure 6b).



(a) "Draws Until" node configuration.

(b) Distribution of rounds required to obtain the Gearbox action cube.

Figure 6. Simulation of rounds required to obtain the cube for the Gearbox action.

5. Final Considerations

This work presents an addition to NodeCrafter, an open-source online modeling environment for tabletop games. This update introduced the capacity to model systems using isolated symbols, a departure from the numerical data manipulations previously available. With isolated symbols, it is possible to model effects such as conditions, token draws (with and without replacement), and to map symbols to values across multiple simultaneous tables.

NodeCrafter can be used by both enthusiasts and professionals to test game subsystems or specific scenarios, aiding in design decision-making. It is expected that its visual language will allow more individuals to utilize modeling and simulation as support during the creative process.

Despite advances in symbolic representations, NodeCrafter is currently restricted to a predominantly linear flow. Lacking mechanisms that allow for flow alteration with more complex conditionals (such as if/else structures), it is not possible to create true branches in the simulated systems, a characteristic that becomes visible when attempting to simulate high-complexity situations. This necessitates the division of such mechanics into isolated models, which compromises prototyping potential and fidelity.

As future work, we intend to provide a mapping of existing game systems in a public model database. This will allow for further exploration of NodeCrafter's representational capacity while increasing the outreach of modeling and simulation within the creative game process. Additionally, we believe the environment has educational potential for teaching both probability concepts and game design, which demands the creation of supplementary documentation and materials.

References

dos Santos Rosa, R. (2023). Ambiente online para modelagem de jogos. Bachelor's thesis, Universidade Federal de Juiz de Fora.

Engelstein, G. (2021). A visual probability analysis tool for board game designers. *Patterns*, 2(5).

Engelstein, G. e Shalev, I. (2022). *Building Blocks of tabletop game design: An encyclopedia of mechanisms*. Crc Press.

Filho, F. F., Gomes, G., Júnior, A. L., Junior, N. C., e Carmo, R. (2021). Balanceamento em jogos para dispositivos móveis: estudo de caso do jogo clash royale. In *Anais Estendidos do XX Simpósio Brasileiro de Jogos e Entretenimento Digital*, pages 48–57, Porto Alegre, RS, Brasil. SBC.

Metropolis, N. e Ulam, S. (1949). The monte carlo method. *Journal of the American Statistical Association*, 44(247):335–341.

Mocanu, D. (2023). Modeling and analyzing board games through markov decision processes.

Ourique, L., Silva, F., Parreiras, M., Magalhães, M., e Xexéo, G. (2024). Balancing and analyzing player interaction in the esg+p game with machinations. *Journal on Interactive Systems*, 15(1):461–477.

Paparistodemou, E., Noss, R., e Pratt, D. (2008). The interplay between fairness and randomness in a spatial computer game. *International Journal of Computers for Mathematical Learning*, 13(2):89–110.

Policarpo, L. S. (2025). Uma ferramenta visual para modelagem e simulação de regras de jogos de mesa. Bachelor's thesis, Universidade Federal de Juiz de Fora.

Schreiber, I. e Romero, B. (2021). *Game balance*. CRC Press.

Silva, F., Ouriques, L., Parreiras, M., Magalhães, M., e Xexéo, G. (2023). Balanceamento do jogo esg+p utilizando o machinations: um estudo de caso. In *Anais Estendidos do XXII Simpósio Brasileiro de Jogos e Entretenimento Digital*, pages 157–168, Porto Alegre, RS, Brasil. SBC.

Zamith, M., da Silva Junior, J. R., Clua, E. W., e Joselli, M. (2020). Applying hidden markov model for dynamic game balancing. In *2020 19th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*, pages 38–46. IEEE.