Computer Aided Content Generation – A Gloomhaven Case Study

Kristian Tijben
kristianwillem@gmail.com
University of Twente, Creative Technology
Enschede, The Netherlands

Marcus Gerhold
m.gerhold@utwente.nl
University of Twente, Formal Methods and Tools
Enschede, The Netherlands

ABSTRACT

We present how an evolutionary algorithm can be used to generate scenarios for the board game Gloomhaven. The scenarios are evaluated according to size, difficulty, thematic coherence, complexity and layout. We encode the game's default scenarios into textual descriptions and use them as initial population for the algorithm. Our dungeon generation works within the confines given by the physical board game, i.e., special attention is given to availability of game pieces and map tiles. The generated dungeons can be constructed without overlapping tiles.

CCS CONCEPTS

• **Human-centered computing**; • **Computing methodologies** → **Search methodologies**;

KEYWORDS

Evolutionary algorithm, dungeon generation, board game, Gloomhaven

ACM Reference Format:
Kristian Tijben and Marcus Gerhold. 2023. Computer Aided Content Generation – A Gloomhaven Case Study. In *Foundations of Digital Games 2023 (FDG 2023), April 12–14, 2023, Lisbon, Portugal.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3582437.3587196

1 INTRODUCTION

"We are in the golden age of board games. It might be here to stay." writes Washington Post's author Jaclyn Preiser [34]. The COVID-19 pandemic was only a catalyst for the surge in popularity of board games. With more and more of our time being reliant on computer screens, many people flock to analogue board games as their preferred pastime [6]. It brings a welcome alternative to digital content, offers satisfying haptics of interacting with physical game pieces, and perhaps most importantly: it brings together people.

Nowhere can this better be seen than in table-top role playing games (TTRPGs) like *Dungeons & Dragons* [31], *Pathfinder* [25], or *Starfinder* [24]. Next to being tactical, turn-based role playing games in their own right, these games excel in collaborative story-telling. Certainly, big television series productions like *Netflix*'s *Stranger Things* [20], or weekly shows on *Twitch* like *Critical Role* [38] have brought these games to the consciousness of mainstream consumers. Many such games require a game master (GM) to function. While players engage with their surroundings, the GM creates, steers and voices the surroundings. Preparing games as a GM can be fun, but also a daunting task [26], and many people start out being a player first. Because of that, some role playing games incorporate the functionality of a GM to the game itself. By providing story booklets and scenario guides players can branch the narratives whichever way they choose without a GM having to prepare outcomes.

A prime example among those role playing games is the award winning Gloomhaven [14]. The self-contained board game weighs roughly 10kg and comes with a plethora of game pieces: detailed miniatures, monster cards and standees, dungeon map tiles that can be combined in a jigsaw-puzzle fashion, a well-defined rule book and 95 hand-crafted scenarios by the games creator Isaac Childres and other various guest-creators.

At the time of writing, the game enjoys a 8.4 out of 10 star ranking on the internet forum BoardGameGeek [22] making it one of the highest ranking board games there. Due to its popularity it is decorated with numerous prizes, and spawned related games: (1) Gloomhaven – *Jaws of the Lion* [15] offers a light-weight introduction to the rules and acts as an entry-point to the full game, (2) Gloomhaven – Forgotten Circles [16] is an add-on that brings more official content to players, and (3) Gloomhaven – the video game adaptation [41] brings the AI-like behaviour of enemies in the board game to screens. Gloomhaven launched on Kickstarter. Its campaign finished in March 2017 with 4,904 backers who pledged roughly 386, 000 USD [28]. The success of the game cannot better be described than by the follow-up Kickstarter campaign for the second printing which had roughly 40,000 backers pledging roughly 4, 000, 000 USD [29]. The game's success continues in Frosthaven, Gloomhaven's continuation, which increased the numbers yet again to 83,193 backers pledging a total of 13, 969, 608 USD [27].

Gloomhaven's scenarios are carefully handcrafted in terms of difficulty, layout and thematic coherence. There is a vibrant online community that takes Gloomhaven creator Isaac Childres' offer [13] and extends the game with their own scenarios and settings [5]. With the plethora of work on procedural content generation for digital games [40], this then raises the question whether existing methods can be leveraged to generate content for the physical board game: Can scenarios and combat encounters be automatically generated? Are they as balanced and immersive as handcrafted ones? Gloomhaven has finitely many game pieces, and yet they can combine to countlessly many different scenarios.

In this paper we present computer aided scenario generation of Gloomhaven dungeons by means of an evolutionary algorithm. Evolutionary algorithms rely on the idea of *survival of the fittest* that produces an elite into a population over many *evolutionary cycles*. The 95 default scenarios form the ground population from which an evolutionary cycle begins to cross-breed dungeons and mutate their properties. We pay particular attention to the constraints of Gloomhaven's physical game pieces: There is a limitation on the

FDG 2023, April 12–14, 2023, Lisbon, Portugal

ACM ISBN 978-1-4503-9855-8/23/04.
https://doi.org/10.1145/3582437.3587196

maximum number of enemies per dungeon and map tiles cannot be generated, adjusted and combined at will.

Paper overview. In Section 2 we present closely related work. Section 3 briefly introduces the game Gloomhaven and the concept of evolutionary algorithms. In Section 4 we present in detail the design choices of our evolutionary algorithm and its fitness function. We showcase two generated dungeons in Section 5. Lastly, Section 6 concludes the paper with a short discussion and future work.

2 RELATED WORK

Procedural content generation is a modern and popular research field [40, 43] focusing on the algorithmic generation of media content. In particular, a plethora of work exists in procedural map generation [35]. While the main focus often lies with digital content, computer aided support for physical board game generation is not unheard of [1, 8]. For instance, the authors of [39] use an ant colony optimisation algorithm to generate maps and game balance in Terra Mystica [33] and Settlers of Catan [42].

The work by [45] discusses the fitness function for map generation in dungeon crawler games. The discussion of mutation operator weights directly translates to our setting. However, unlike our presented work, their approach is not limited by physical game pieces. A taxonomy of analog procedural content generation methods is presented in [7] with a focus on war games such as Warhammer 40k [47] or Axis and Allies [23].

A taxonomy of search-based PCG is provided in [44], classifying the content of generation and its evaluation. Our focus of generating physical dungeon layouts clearly falls into the category of offline generation. The author of [10] shows how floor maps consisting of rectangles can be generated given certain constraints. Another example of offline generation can be found in the seminal work of Browne & Maire [8]. The authors apply the evolutionary cycle to develop board game rule systems offline that are to be automatically evaluated. While we focus on content for board game whose rule set is known, the evolutionary approach remains similar.

The authors of [3] present an interactive evolutionary dungeon designer and follow-up their initial study with a focus on dungeon aesthetics [2]. This closely reflects our focus on dungeon *thematic coherence*: We believe that Gloomhaven dungeons should not introduce too many different thematic components such as enemy types or terrain. Notably this is something that the game's inherent random dungeon generator does not always provide, cf. Figure 1.

A taxonomy of data structures in games is provided in [19]. This perfectly includes the translation of hexgrids to graphs we apply to Gloomhaven dungeons to guarantee solvability of the generated scenarios. We employ similar graph-based techniques like [30] to ensure that dungeons are solvable by base game actions and do not require specific classes or items.

3 BACKGROUND

We briefly introduce the game Gloomhaven, the concept of evolutionary algorithms, and we show how we apply it to Gloomhaven scenarios. We believe that the detailed rules of Gloomhaven are not crucial to the understanding of our work. Thus, we declare them beyond the intended scope of this paper and refer the interested reader the official rules book [14] or the brief introduction video [9]. Instead, we give a high-level overview of Gloomhaven's game-loop, and which part thereof we focus on in the presented work.

3.1 The Role-Playing Game Gloomhaven

Gloomhaven is a board game owned by Cephalofair which was developed by Isaac Childres. At its core the game-loop is a combination of turn-based tactical scenarios that are embedded in an evolving campaign. Players take on the role of a band of mercenaries that travels through the surroundings of the city Gloomhaven to solve mysteries, fight monsters and collect treasure. The first campaign scenario, for instance, tasks players to collaboratively fight bandits and skeletons in the dungeon "Black Barrow" before facing a boss type enemy in the follow-up scenario "Barrow Lair". From here, the story forks off into many possible paths which further diverge based on player characters and their personal quests. After each dungeon players progress their characters by acquiring stronger skills or buying new equipment with money that was collected or rewarded, and they progress the story by engaging with events in and around the city of Gloomhaven.

The game is played cooperatively by two to four players, and each scenario approximately takes 30 minutes per player. The game's difficulty is adjustable to the players' liking ranging from zero to seven. The properties of enemies on difficulty setting seven are vastly superior to that of setting zero. Players can choose which setting they would like to engage with to tailor their experience, but the game provides a recommendation based on the player characters' own level. The game includes a scenario booklet that narratively introduces the dungeons and defines rules and objectives without the need of a game master (GM). There are a total of 95 scenarios differing in difficulty and settings. In the game, players explore dungeons infested by skeletons and the cultists who summoned them, travel to distant planes of existence to fight elemental spirits and demons, defy the harsh surroundings of Gloomhaven's outdoors or battle the very city guard that is protecting the city's walls.

The game offers high replayability due to its character progression and customization options. In addition, Gloomhaven contains a random dungeon deck, i.e., a method to generate random dungeons on-the-fly via a system of random dungeon and monster cards, cf. Figure 1. While the 95 scenarios offer high thematic coherence in narrative, terrain and enemy types, a random scenario is literally that: random. To illustrate, a random scenario always consists of three rooms with unique sets of enemies. Thus, one room could contain cultists summoning living bones in a cavern, the next could swarm with flying drakes on icy terrain, and the last could contain city guards on a tile depicting wooden surroundings. Isaac Childres encourages players to create new content for the game [13] and the internet forum BoardGameGeek [22] has a large collection of fan-made scenarios using the game's game pieces and rule system [5].

In this paper we present a new way to generate Gloomhaven scenarios[1] by means of an evolutionary algorithm. Our algorithm addresses the lack of *thematic coherence* exhibited by the random

[1]We use the words "dungeon" and "scenario" interchangeably to refer to Gloomhaven game scenarios that including physical map tiles, enemies, treasure, and scenario rules.

Figure 1: Cards of the random dungeon deck. The card on the left specifies the room tile, while the card on the right populates it with monsters, traps and treasures. The images were taken from the Gloomhaven manual [14].

dungeon deck, cf. Figure 1. Special attention is paid to the limitations posed by physical game pieces that are sold with a copy of the game.

3.2 Evolutionary Algorithms

Evolutionary algorithms [36] have a plethora of applications ranging from engineering [18] over management [4] to game rules [8]. While the applications vary, the crucial components remain similar:

Population. An initial set of individuals comprising a population. Each member possesses unique properties called the *genotypes* specific to this individual. The fitness of this population is measured according to some *fitness function*.

Crossover. An evolutionary algorithm chooses two distinct parent individuals, takes into account their fitness, and produces a new member based on their crossed-over genotype.

Mutation. The *offspring* of two parents undergoes mutation of its genotype to avoid inbreeding and to allow for evolution.

Fitness. The newly crossed-over and mutated individual gets measured according to a fitness function. This fitness function is at the core of an evolutionary algorithm. It describes what constitutes a *good* individual. After the evaluation the newly generated individual gets added to the population and the cycle begins anew.

The core concept of an evolutionary algorithm is that many cycles of reproduction eventually lead to an elite surfacing in the population that is highly optimised with respect to the fitness function. Thus, changing the fitness function steers the optimisation by giving concrete numeric incentives.

We apply the cyclic algorithm to Gloomhaven dungeons. The initial population is a subset of Isaac Childres carefully crafted 95 scenarios. Some scenarios, such as boss scenarios, were left out of the ground population for the sake of simplicity. Those scenarios use mechanics that are unique to the boss levels e.g., the "Bandit Commander" opens doors, or the "Dark Rider" appears on marked placement tokens on the map. During the crossover phase, two scenarios are chosen. The algorithm then randomly selects the genotype of either parent for (1) rules, (2) map tiles, (3) monsters, (4) environment, and (5) treasures. For instance, given scenarios A and B, the rules and map tiles of A may host the monsters, environment and treasures of scenario B in the offspring.

During the mutation phase each single element of a genotype can be mutated with a chance p. Therefore, mutation of multiple features at once is possible, albeit with lower probability. For instance, it is possible that both the monsters and the rules mutate during one cycle. Lastly, the fitness function is applied to the newly generated dungeon, and it is added to the population.

4 IMPLEMENTATION

We explain our algorithm[2] and design choices alongside Figure 2. The algorithm is written in Python [46]. A starting population is the prerequisite for the evolutionary algorithm. We chose to encode a subset of the default scenarios designed by the game's creator Isaac Childres. Special scenarios like boss scenarios were excluded due to their reliance on special rules. They can be added later.

Gloomhaven's first introduction scenario is called *#1 Black Barrow*, cf. Figure 3. It contains the three enemy types "Bandit Guard", "Bandit Archer" and "Living Bones" spread out on three different room tiles "L1a", "G1b" and "I1b". The goal of the scenario is to "kill all enemies". We encoded this information in a Python dictionary, cf. Figure 4.

Example 4.1. The encoding of room tiles in Line 5 includes the physical tile, its side (either "a" or "b") and its orientation in multiples of 60 degrees. This ensures that the mutation algorithm will not use sides "a" and "b" for one map. The connections of the physical map pieces are defined in Line 6. The first three numbers are cubic hex-coordinates that give the unique location in the hexgrid, cf. Figure 5. The last number indicates its rotation angle: Physical tiles are connected like jigsaw-pieces with an entry and an exit. Naturally, both only fit together if they are aligned correctly, e.g., if an exit is positioned with 0 degree with respect to the hexgrid, then its connecting piece must be aligned with 180 degree, lest the jigsaw connectors will not fit. The last number represents this in increments of 30 degrees, e.g., the exit of map tile "L1a" is positioned in a 90 degree angle that connects to the entry of "G1b" at a 270 degree angle.

Line 7 defines the enemy types and the number of normal and elite monsters. The latter are typically stronger and thus increase the dungeon's overall difficulty. The number of both varies depending on the number of player characters. I.e., a dungeon contains more enemies for four players than for two to scale the difficulty.

Lines 8 to 16 define the number and location of coin and treasure tiles, and additional overlay tiles such as traps and obstacles. We define a nested dictionary of placement locations with hex-coordinates of all these items. For the sake of readability it is not included here, since it only consists of lists containing cubic hexgrid coordinates. For instance, the unique treasure chest has the coordinates $[7, -10, 3]$ and both of the dungeon's damage traps are located at $[6, -4, -2]$, $[7, -4, -3]$. Lastly, Line 16 defines the main theme of the map "Dungeon", comprising all brown/orange map tiles with stone floor tiles.

[2]The implementation can be found in the GitHub repository: https://github.com/kristianwillem/GloomDungeons.

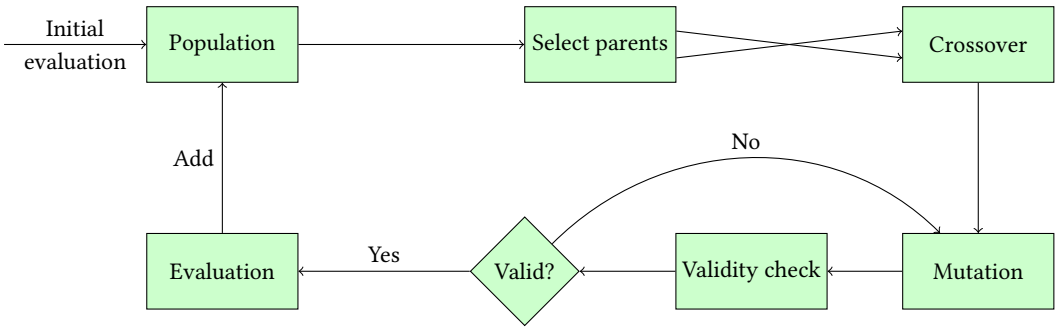


Figure 2: The evolutionary cycle applied to Gloomhaven dungeons. In an initial evaluation a subset of the game's default 95 scenarios is rated according to a fitness function. Two parents are chosen according to a uniform distribution over their rating, which makes it more likely to choose *fit* parents. The genotypes of parents are crossed-over and properties of either are chosen with a ratio of 50:50. The mutation step changes random elements of the genotype before a validity check happens. If the check is passed, the newly created dungeon enters the population and the cycle begins anew.

Figure 3: Gloomhaven Scenario #1 - Black Barrow [14].

```
black_burrow = {
  "name": "Black Barrow",
  "goal": "kill all enemies",
  "rules": [],
  "rooms": [["L1","a",3], ["G1","b",0], ["I1","a",3]],
  "connections": [["L1", [2, -3, 1, 3, "exit"],
                   "G1", [-1, 1, 0, 9, "entry"]],
                  ["G1", [4, -1, -3, 0, "exit"],
                   "I1", [-3, 1, 2, 6, "entry"]]],
  "dungeon_monsters": [["Bandit Guard", 4, 1],
                       ["Bandit Archer", 2, 1],
                       ["Living Bones", 2, 0]],
  "obstacles": 4,
  "traps": 2,
  "hazardous_terrain": 0,
  "difficult_terrain": 0,
  "chests": ["item"],
  "coins": 5,
  "placements": black_barrow_placement,
  "start": 7
  "main_theme": "Dungeon",
}
```

Figure 4: Encoding of the default Gloomhaven scenario #1 – Black Burrow [14].

4.1 Select Parents

Every dungeon in the population is initially evaluated according to the fitness function, cf. Section 4.5. To ensure a general tendency towards elitism in the population, we give a preference to those dungeons with a high fitness value. This is done by a uniform distribution over a dungeon's own contribution to the sum of all fitness values. E.g. if dungeons A, B and C have fitness values of 1, 2 and 3 respectively, then the chance to select A is 1/6, the chance to select B is 1/3 and the chance to select C is 1/2.

4.2 Crossover

We define the genotype of a dungeon to consist of (1) rules, (2) map, (3) monsters, (4) overlay tiles, and (5) treasure. We showcase their encoding in Example 4.1 and Figure 4, respectively. A new offspring dungeon is created by randomly choosing each property of parent dungeon A or parent dungeon B with equal probability.

4.3 Mutation

Where the crossover was a simple selection between properties of A or B, the mutation step of each gene differs. Each gene in the genotype of the offspring has probability p to mutate. E.g., the probability that all genes mutate in one step is p^5, and the probability that exactly two genes mutate is $p^2 \cdot (1-p)^3$. An easy extension is to give each genotype its own mutation probability $p_1, \ldots, p_5$. For simplicity we did not include it yet in our evolutionary algorithm.

Rule mutation. The pool of possible scenario rules is predefined by the scenario book. For the sake of simplicity, our implementation contains the rules "All characters start the scenario with the poison/disarm/wound/immobilize effect" and "Add three curse cards/-1 attack modifier cards to each character's attack modified deck". If the rule mutation triggers then one of the predefined rules

is chosen with some probability. The probability to choose another rule decreases by half with every new addition. Thus, it might also be possible that no rule is chosen at all, cf. Figure 4.

Monster mutation. Each monster is assigned a theme. The bandit guards and archers of Example 4.1 are of theme "Outlaw", whereas the living bones are of theme "Undead". This is done to maintain thematic coherence during the dungeon evolution. To do so, we introduce a bias towards a theme. Hence, it is possible to introduce new monster themes, e.g., "Elementals" and "Undead", but the probability of doing so is set. We point out that the initial definition of each theme was done by us and can be changed on demand. As an extension, it would be interesting to *infer* the different themes from the original 95 scenarios. For now we declare this beyond the scope of our work and focus on the usage of the theme instead.

If the monster mutation triggers, new monster are chosen depending on theme, difficulty and maximal game piece availability. To help creators balance combat encounters Isaac Childres has given each monster an inherent difficulty rating ranging from 0.5 for easy to 2.0 for difficult [11]. Additionally, Isaac Childres introduces a factor of $\times 2$ for elite monsters [12] in his difficulty rating. We emphasize that this difficulty rating is not the one chosen by the players ranging from zero to seven, but scenario inherent, cf. Section 3.1.

Our algorithm calculates the total difficulty and the number of monster types of the original offspring. It then chooses new values for both of them with a normal distribution, i.e. the number of different monster types and absolute number of monsters of this type may change. It then populates the dungeon until the initial difficulty rating threshold is reached by either adding new monsters or promoting them to elite status.

Example 4.2. The total enemy difficulty rating of Example 4.1 is twelve. It comprises: four regular "Bandit Guards" (1), one elite "Bandit Guards" (2), two regular "Bandit Archers" (1), one elite "Bandit Archer" (2), and two regular "Living Bones" (1). Since all enemies have inherent difficulty rating one, the total difficulty of the dungeon yields twelve. During the mutation step, all of these enemies could, for instance, be replaced by three regular "Stone Golems" (2), one elite "Stone Golem" (4), and lastly four "Ancient Artillery" (1). All newly chosen enemies are of type "machine" such that the thematic consistency persists. Lastly, the chosen enemies physically occur six times in the base game, thus making this setting possible.

Overlay tile mutation. If the overlay tile mutation is triggered, all overlay tiles of the offspring dungeon are counted. This includes obstacles, difficult terrain, hazardous terrain and traps. The actual mutation uses a Gaussian distribution to ensure that the new number of overlay tiles will be between 0.5 to 1.5 times that of the original with a probability of 95%. Hence, the number of overlay tiles may increase or decrease in every evolution cycle.

Treasure mutation. Coin tokens mutate by using the same mechanic as overlay tiles, i.e. there is a 95% change that the new number is 0.5 to 1.5 times the original. Treasure tiles use the same mutation mechanic like rules, i.e., the chance that new treasure is added decreases. The first one has a probability p, the second has $p/2$ etc. The treasure chest content is randomized uniformly over a handpicked selection of possible treasures.

Map mutation. The physical board tiles and puzzle connector pieces prevent the map mutation step from being straightforward. Intuitively, one would want to swap one map tile for a random other one. However, the connecting pieces must fit while simultaneously not causing an overlap of the physical game tiles, cf. Example 4.1.

Instead, we chose the following method to generate a map from ground up: First the algorithm takes the minimum and maximum total hexes of dungeons in the population to establish a distribution over desirable numbers. E.g., if the smallest dungeon had two hexes, and the largest one 100, the map mutation creates a map that has a normally distributed number of hexes around the mean of 51 hexes.

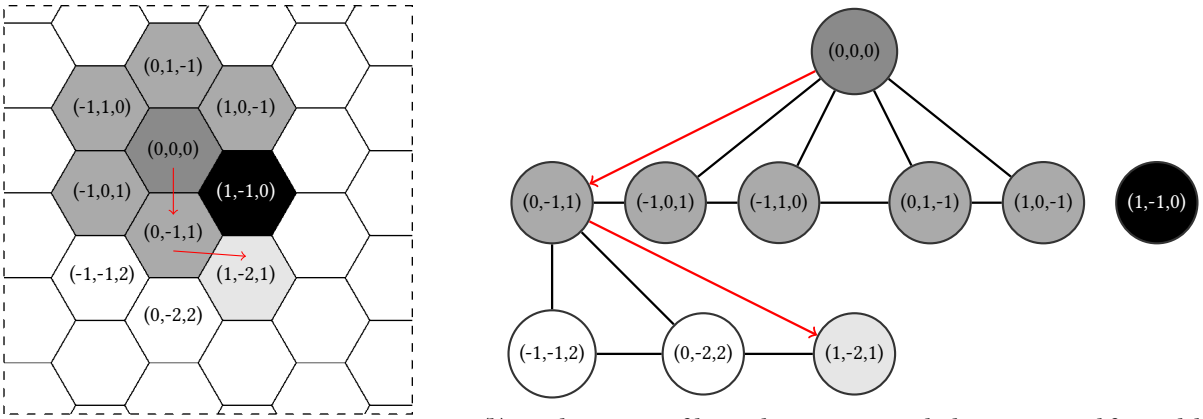
To adhere to the restrictions of the physical game pieces the new map is constructed step-by-step. First, a map tile is chosen at random. Then a random connecting piece is chosen. This new piece has to adhere to two conditions: (1) it cannot be a tile that was already added to the map, neither the "a" nor the "b" side, and (2) the rotation of the piece must be chosen, such that it aligns the jigsaw-puzzle connectors. The latter also involves overlapping of the card-board pieces beyond just the connecting hex. To guarantee no overlapping, we run a validity check that is further explained in Section 4.4. It is based on the underlying graph structure of the hexes, cf. Figure 5.

Map tiles are added with some probability until the number of hexes lies between the predefined minimum and maximum tiles. For simplicity, our implementation always places an unopened door between two map tiles. There are scenarios in the Gloomhaven booklet that connect two map tiles by covering the wall with empty floor tiles. This creates one big room instead of two smaller ones.

The last step of the map mutation mutates the *placements* of tiles such as overlay tiles, treasures or monsters. Since the map is known, this phase of the algorithm distributes each piece over unoccupied hexes of the map in a uniform distribution. For the sake of readability we grouped these hex-coordinates together in a separate dictionary, cf. Line 14 in Figure 4. One exception is the placement of the starting locations; they are clustered together in almost all default scenarios, cf. Example 4.1. Hence, our implementation does the same.

4.4 Validity Check

Gloomhaven offers the players a variety of classes to play as, e.g., the melee class "Brute" or the ranged class of "Doomstalker". To guarantee solvability of the generated scenarios no matter the choice of classes, we apply a validation step at the end of the cycle. This prevents the generation of unsolvable *deadlock* scenarios. This could be a dungeon that has the completion rule "kill all enemies", and an enemy that is locked behind a wall of obstacles – an impossible ask for a melee-only player. Next to guaranteeing the ability to solve the game with just base walk and base attack actions, this checks the limitations of the physical game pieces, i.e. overlapping map pieces. Deadlock scenarios and physical limitations are checked with a graph transformation of the hexgrid and resulting graph search algorithms, cf. Figure 5.



(a) Hexgrid with cubic coordinates.

(b) Graph structure of hexgrid. Many states and edges are omitted for readability.

Figure 5: Converstion from a hexgrid (left) to a graph (right). The hexgrid origin is labeled $(0, 0, 0)$. The red line indicates a path through the grid and graph, respectively. Notably, the node $(1, -1, 0)$ is an obstacle. Thus, it can neither be reached nor traversed. To ensure solvability of a Gloomhaven dungeon a depth-first search is performed from the starting location(s) to each enemy coordinate. If there is an enemy that cannot be reached, the scenario cannot be solved with only the base attack and base move action. Note that many coordinates, nodes and connecting edges were omitted for readability.

Graph construction. We construct a graph data structure from the hexgrid of the dungeon map. This process can be seen in Figure 5. One hex is selected as the origin of the coordinate system and number triplets give each other hex a unique location with respect to the origin. The graph on the right hand side in Figure 5 consists of vertices and edges. Two connected vertices thus indicate that movement from one hex to the other is possible. Searching a path inside the hexgrid then turns into a standard search for paths on mathematical graphs with algorithms such as depth-first search (DFS) or breadth-first search (BFS) [17].

Reachability analysis. To facilitate that each scenario is solvable irrespective of player class or equipment, we require that scenario goals are achievable by only the base attack action and base move action available to all classes. Our algorithm checks via graph search algorithms such as DFS whether there exists a path on the graph from the players to each enemy. Should there be an enemy that is not reachable via DFS, then the player character cannot eliminate it with the standard melee attack. As a result, the generated scenario is discarded and the mutation step is repeated, cf. Figure 2. This state reachability is a well-established procedure and has been done for other games like Sokoban [30]. Currently our implementation only supports the scenario goal "kill all enemies". However, the same validity check could be done for goals like "loot all treasure tiles" and the basic loot action, or reaching specific hex coordinates on a map and the basic move action.

Physical restrictions. The dungeon layouts are limited by physically existing game pieces. Our algorithm ensures that the total number of available game pieces is not exceeded, no single map tile is used twice via its "a" or "b" side, and that the created dungeon causes no overlap of physical map pieces. This highlights the key difference between working within the constraints of a physical board game versus a purely digital one. In a digital version of Gloomhaven, an overlap could be avoided by, e.g., stretching a single hexagon, or introducing elevation levels.

The map is either directly given from the crossed-over parents and correct by transitivity, or newly created in the mutation phase, cf. Section 4.3. In the latter, we can use the hexgrid to determine its validity. Should overlapping game pieces exist, then two nodes in the hexgrid will have the same coordinates. Due to the uniqueness of coordinates with respect to the origin $(0, 0, 0)$, this is not allowed. Consequently, the map is discarded and the mutation begins anew, cf. Figure 2. Since a mutation is triggered with a non-zero probability p, there is a non-zero chance that no mutation will happen which is $(1 - p)^5$. This ensures that the algorithm terminates almost surely.

4.5 Evaluation

After a dungeon was crossed-over, mutated and validated it will be added to the population. For that, its fitness needs to be determined. The fitness function is the core of every evolutionary algorithm. Our approach includes the following components: (1) difficulty, (2) size, (3) complexity, (4) theme, and (5) layout. This results in our proposed fitness function which we briefly explain below:

$$\begin{aligned} fitness \quad = \quad & w_1 \cdot difficulty + w_2 \cdot size + w_3 \cdot complexity + \\ & w_4 \cdot theme + w_5 \cdot layout. \end{aligned} \tag{1}$$

Difficulty is the sum of difficulty ratings of all monsters in a dungeon. The rating was introduced by the game's creator Isaac Childres to help players create dungeons. Note that this does not refer to the difficulty levels zero to seven that can be chosen by players at the start of a scenario, cf. Section 3.1.

Size is the absolute number of all hexes on the map, including connection tiles on jigsaw-pieces.

Complexity is the number of rules in a scenario, e.g., "all characters start poisoned".

(a) Generated dungeon "Overgrown Ruin". (b) Generated dungeon "Drake Kennels".

Figure 6: Two dungeons that were generated using the evolutionary algorithm described in Section 4. Both dungeons are the top ranked individuals in their batch with respect to their fitness value after 100 cycle iterations. "Overgrown Ruin" had a total score of 4.83 out of a possible 5, and "Drake Kennel" had 4.85.

Layout is the ratio of all occupied versus empty hexes. A hex can be occupied by at most one of monsters, obstacles, overlay tiles, treasures, coins or starting locations.

Theme consists of the number of used monster themes, e.g., "Undead" or "Machine", and the used dungeon theme, e.g., "Cave" or "Building".

Defining a fitness function steers the evolutionary cycle and defines what is considered a *good* individual. For the properties *difficulty*, *size*, *complexity* and *layout* that fitness function is defined by the default scenario book's average, i.e., for each property p:

$$p = 1 - \frac{|fitness_{ideal}(p) - fitness_{new\_dungeon}(p)|}{fitness_{ideal}(p)} \tag{2}$$

For instance, according to Isaac Childres the ideal dungeon difficulty lies between 30 and 36 for four players. Thus, choosing the average in $fitness_{ideal}(difficulty) = 33$ is a reasonable choice[3]. Moreover, the average number of hexes of all scenarios we used in our prototype was 90. Hence, we initially set $fitness_{ideal}(size) = 90$.

Only the fitness of *theme* is calculated differently. Under the assumption that a highly thematically coherent dungeon is desirable, each additionally used *theme* is penalized. Thus,

$$theme = 1 - thematic\_difference \tag{3}$$

where $thematic\_difference \in [0, 1]$.

[3]Creating scenarios for two players requires us to halve this number to a range of 15 to 18 with an average of 16.5. For numerical reasons we doubled this number again and adjusted all other difficulty weights accordingly.

Combining all in the fitness equation (1) means that an "ideal" dungeon has the overall fitness of sum of all its weights. While it is possible to construct such a dungeon manually, our approach repeatedly creates new ones whose score will be close to ideal.

5 EXPERIMENTS

We present two scenarios that were generated with the evolutionary algorithm described in Section 4. Both dungeons are the best-rated dungeon according to the fitness function (1) in their respective batches after 200 iterations of the evolutionary cycle. The execution time for 200 iterations of the cycle is close to instantaneous (approximately one second) on a Lenovo Carbon X1 10th generation with Intel® Core™ i7-1260P processor and 32GB RAM.

To test the feasibility of the generation we set $w_i = 1$ for all $i = 1, \ldots, 5$, i.e., all components of a dungeon's genotype were ranked as equally important. Given the allocation of the weights the highest score possible is 5.

5.1 Generated Dungeon 1 – "Overgrown Ruin"

The generated dungeon can be seen in Figure 6a. It was the best dungeon of its population after 200 cycles with the scores:

Total	Difficulty	Size	Complexity	Layout	Theme
4.83	0.92	0.96	1.0	0.95	1.0

Evidently, the fitness of all sub-components came close to the ideal value specified in (2) and (3), respectively. The generated dungeon has the two enemy types "Ooze" and "Giant Viper". Under

our enemy type categorization both fall under "Poison". The used dungeon tiles are of category "Dungeon". Given this thematic composure we baptised it "Overgrown Ruins".

We would like to highlight two things: (1) The selection and layout of the map clearly shows that tiles do not overlap, even though the space is rather tight. This is based on the underlying graph transformation, checking whether every cubic hexcoordinate in the graph is unique, cf. Figure 5. (2) The "layout" value, i.e., the value accounting for hexes occupied with treasure chests, coins, traps and obstacles is 0.95. According to our specification, this is close to the ideal value. Notably, the generation produced an obstacle closely edged into the corner of a room, making the hex behind it entirely inaccessible. This hex, however, does not contain an enemy. As such, the algorithm deems the dungeon solvable by basic move and basic attack options, and it is not discarded.

5.2 Generated Dungeon 2 – "Drake Kennels"

The generated dungeon can be seen in Figure 6b. It was the best dungeon of its population after 200 cycles with the following scores

Total	Difficulty	Size	Complexity	Layout	Theme
4.85	1.0	0.97	1.0	0.93	0.95

Again, the fitness of all sub-components came close to the optimally specified values of (2) and (3). The dungeon contains "City Guard" and "Rending Drake" enemies. The first belongs to "City" type and the latter counted towards the "Drake" type according to our classification. Given that all the map tiles belonged to the "Dungeon" category we baptised the scenario "Drake Kennels".

Compared to "Overgrown Ruin" the dungeon is rather compact. Two interesting things stand out: (1) Many dungeons of the default 95 scenarios connect two map tiles of equal size to one big one by covering the wall with floor tiles. "Drake Kennel" connects them via one door, instead. Our algorithm currently only checks for the connecting jigsaw-pieces and places door tiles on top of them. Hence, covering a wall between two rooms entirely with floor tiles is currently not possible. (2) The separate room containing four coin tokens is rather unusual given the default scenarios, but all within the boundaries of our algorithm. Adding some narrative to the dungeon, this could be a treasury or treasure hoard.

6 CONCLUDING REMARKS

We presented how new Gloomhaven scenarios can be generated with an evolutionary algorithm. A subset of the 95 existing default scenarios is encoded in data structures that can be used for crossbreeding and mutation. Due to a validity check we guarantee that our generated scenarios can be build by using the game's default game pieces. In addition, they are solvable by all player classes, and incentivised by our fitness function to be thematically coherent and close to existing scenarios in terms of size and difficulty.

We conclude the paper with a short discussion of our work, and indicate directions for future work and extensions to our algorithm.

6.1 Discussion

The application of computer aided game content design via an evolutionary algorithm is not limited to Gloomhaven. Any game of sufficient complexity could benefit from this approach. Additionally, the same approach can be used on rule mechanics to create entirely new games [8]. The linchpin in all applications of evolutionary algorithms is the fitness function. We began this research due to the perceived lack of thematic coherence in the game's own random dungeon generator and consequently made it a factor in our fitness function. In addition to *theme* our fitness function includes: *difficulty*, *size*, *complexity* and *layout*. However, a well-designed Gloomhaven scenario is much more than the sum of its parts; There is elegance in storytelling by strategic object placement or enemy placement, symmetry, or special scenario rules and game tiles. While these are currently not covered by our fitness function, the modularity of our implementation makes it such that adaptations can easily be done.

Ultimately, we are interested in creating scenarios that are to be enjoyed by human players. Currently it is unclear to what extent individual weights and the fitness value as a whole predicts enjoyment. Our interest therefore is twofold: (1) On one hand future research should focus on playtesting and measuring the contribution of individual weights in the fitness function. This can be done either by humans, or automating this process by using AI-agents to some extent [8, 21]. (2) On the other hand it is important to find important metrics to include in the fitness function and to strengthen the links between the currently existing ones. For instance, hazardous terrain and rules that impose a disadvantage on players should be intertwined with a scenario's difficulty rating.

We also point out that the evolutionary algorithm creates standalone scenarios that are supposed to be thematically coherent. To further automate the content generation and to easily create a narrative surrounding a generated scenario we could utilize large language models like ChatGPT [32]: Given the outlines of a scenario like enemy types, map tiles, scenario goal and rules, ChatGPT could craft a story that accompanies the scenario.

6.2 Future Work

There is some work to be done to refine our tool for Gloomhaven to facilitate its usefulness in real life game sessions as an on-the-fly way to generate dungeons. In its current state our tool outputs a textual description of a Gloomhaven dungeon using cubic hexcoordinates. This can be quite challenging for human players to parse on the spot. Since all game pieces are known *a priori*, an easy follow-up is the visualisation by means of images and map depictions. This aligns our algorithmic output with the depiction of scenarios in the game manual. Integration into existing tools is rather easy. For instance, [37] is a service that allows dungeon imports via files in the JSON format. To ensure that generated dungeons fit on a physical table, the validation step could include a check for maximal physical size: Each hex is roughly 32mm from side to side and 38mm from corner to corner. The cubic hex-coordinates can then easily be utilized such that the maximal length/width threshold is not exceeded.

Beyond that, the customization possibilities of the algorithm are near endless: An obvious next step is to facilitate the generation of scenarios for more than two players. With some effort we could also generate new boss scenarios by further linking scenario goals to the existence of certain game pieces, e.g., the "Bandit Commander"

requires the existence of more than one door in its room, such that it can use its special ability to open them. We already outlined that the existing validity check could be utilized for that and other scenario goals: In addition to requiring reachability of all enemies by players in the hexgrid, the same could be done for treasure chests, special obstacles etc. We could incentivise the algorithm to cluster together overlay tiles such as the water in "Drake Kennels". We could also use other distributions to calculate the contribution of individual properties, i.e. formulas (2) and (3).

We could, however, also embrace some quirkiness of the generated dungeons, take on some agency as Game Masters and let them inspire hand-crafted scenario descriptions.

REFERENCES

[1] Ingo Althöfer. 2003. Computer-aided game inventing. *Friedrich Schiller Universität, Jena, Germany, Tech. Rep* (2003).

[2] Alberto Alvarez, Steve Dahlskog, Jose Font, Johan Holmberg, and Simon Johansson. 2018. Assessing Aesthetic Criteria in the Evolutionary Dungeon Designer. In *Proceedings of the 13th International Conference on the Foundations of Digital Games* (Malmö, Sweden) *(FDG '18)*. Association for Computing Machinery, New York, NY, USA, Article 44, 4 pages. https://doi.org/10.1145/3235765.3235810

[3] Alberto Alvarez, Steve Dahlskog, Jose Font, Johan Holmberg, Chelsi Nolasco, and Axel Österman. 2018. Fostering Creativity in the Mixed-Initiative Evolutionary Dungeon Designer. In *Proceedings of the 13th International Conference on the Foundations of Digital Games* (Malmö, Sweden) *(FDG '18)*. Association for Computing Machinery, New York, NY, USA, Article 50, 8 pages. https://doi.org/10.1145/3235765.3235815

[4] Jörg Biethahn and Volker Nissen. 2012. *Evolutionary algorithms in management applications*. Springer Science & Business Media.

[5] BoardGameGeek. 2000. BoardGameGeek: Gloomhaven: Scenarios. https://boardgamegeek.com/wiki/page/thing:174430:Scenarios. Accessed: 2023-01-27.

[6] Paul Booth. 2021. *Board Games as Media*. Bloomsbury Publishing USA.

[7] Joseph Alexander Brown and Marco Scirea. 2018. Procedural Generation for Tabletop Games: User Driven Approaches with Restrictions on Computational Resources. In *Proceedings of 6th International Conference in Software Engineering for Defence Applications, SEDA 2018, Rome, Italy, June 7-8, 2018 (Advances in Intelligent Systems and Computing, Vol. 925)*, Paolo Ciancarini, Manuel Mazzara, Angelo Messina, Alberto Sillitti, and Giancarlo Succi (Eds.). Springer, 44–54. https://doi.org/10.1007/978-3-030-14687-0_5

[8] Cameron Browne and Frédéric Maire. 2010. Evolutionary Game Design. *IEEE Trans. Comput. Intell. AI Games* 2, 1 (2010), 1–16. https://doi.org/10.1109/TCIAIG.2010.2041928

[9] Gaming Rules! YouTube Channel. 2016. How to Play Gloomhaven - Official Tutorial. https://youtu.be/XUx7riwsrwY. Accessed: 2023-01-27.

[10] Philippe Charman. 1994. A Constraint-based Approach for the Generation of Floor Plans. In *Sixth International Conference on Tools with Artificial Intelligence, ICTAI '94, New Orleans, Louisiana, USA, November 6-9, 1994*. IEEE Computer Society, 555–561. https://doi.org/10.1109/TAI.1994.346443

[11] Isaac Childres. 2015. Gloomhaven Monster Spoilers. https://cephalofair.com/pages/gloomhaven-monster-spoilers. Accessed: 2023-01-27.

[12] Isaac Childres. 2015. Isaac Childres Forum Post. https://boardgamegeek.com/thread/1430857/submissions-fan-made-scenarios-are-open. Accessed: 2023-01-27.

[13] Isaac Childres. 2016. The Endless Adventure. https://cephalofair.com/blogs/blog/endless-adventure. Accessed: 2023-01-27.

[14] Isaac Childres. 2017. Gloomhaven. Board Game.

[15] Isaac Childres. 2020. Gloomhaven: Jaws of the Lion. Board Game.

[16] Isaac Childres and Marcel Cwertetschka-Mattasits. 2020. Gloomhaven: Forgotten Circles. Board Game.

[17] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2022. *Introduction to algorithms*. MIT press.

[18] Dipankar Dasgupta and Zbigniew Michalewicz. 2013. *Evolutionary algorithms in engineering applications*. Springer Science & Business Media.

[19] Luiz Jonata Pires de Araujo, Mariia Charikova, Juliano Efson Sales, Vladislav Smirnov, and Ananga Thapaliya. 2019. Towards a Game-Independent Model and Data-Structures in Digital Board Games: An Overview of the State-of-the-Art. In *Proceedings of the 14th International Conference on the Foundations of Digital Games* (San Luis Obispo, California, USA) *(FDG '19)*. Association for Computing Machinery, New York, NY, USA, Article 93, 8 pages. https://doi.org/10.1145/3337722.3342238

[20] Matt Duffer and Ross Duffer. 2016. Stranger Things. TV series.

[21] Raluca D. Gaina, Martin Balla, Alexander Dockhorn, Raúl Montoliu, and Diego Perez Liebana. 2020. TAG: A Tabletop Games Framework. In *Joint Proceedings of the AIIDE 2020 Workshops co-located with 16th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE 2020), Worcester, MA, USA, October 19-23, 2020 (online) (CEUR Workshop Proceedings, Vol. 2862)*, Joseph C. Osborn (Ed.). CEUR-WS.org. https://ceur-ws.org/Vol-2862/paper9.pdf

[22] Board Game Geek. 2000. Board Game Geek. https://boardgamegeek.com/. Accessed: 2023-01-27.

[23] Larry Harris. 1984. Axis & Allies: Classic. Board Game.

[24] Paizo Inc. 2017. Starfinder. Core Rulebook.

[25] Paizo Inc. 2019. Pathfinder. Core Rulebook Second Edition.

[26] Akrivi Katifori, Dimitra Petousi, Pantelis Sakellariadis, Maria Roussou, and Yannis Ioannidis. 2022. Tabletop Role Playing Games and Creativity: The Game Master Perspective. In *Proceedings of the 17th International Conference on the Foundations of Digital Games* (Athens, Greece) *(FDG '22)*. Association for Computing Machinery, New York, NY, USA, Article 62, 7 pages. https://doi.org/10.1145/3555858.3555918

[27] Kickstarter. 2015. Kickstarter Campaign Frosthaven. https://www.kickstarter.com/projects/frosthaven/frosthaven. Accessed: 2023-01-27.

[28] Kickstarter. 2015. Kickstarter Campaign Gloomhaven. https://www.kickstarter.com/projects/frosthaven/gloomhaven. Accessed: 2023-01-27.

[29] Kickstarter. 2017. Kickstarter Campaign Gloomhaven – 2nd printing. https://www.kickstarter.com/projects/frosthaven/gloomhaven-second-printing. Accessed: 2023-03-23.

[30] Yoshio Murase, Hitoshi Matsubara, and Yuzuru Hiraga. 1996. Automatic Making of Sokoban Problems. In *PRICAI'96: Topics in Artificial Intelligence, 4th Pacific Rim International Conference on Artificial Intelligence, Cairns, Australia, August 26-30, 1996, Proceedings (Lecture Notes in Computer Science, Vol. 1114)*, Norman Y. Foo and Randy Goebel (Eds.). Springer, 592–600. https://doi.org/10.1007/3-540-61532-6_50

[31] Wizards of the Coast. 2014. Dungeons and Dragons fifth Edition. Player's Handbook.

[32] OpenAI. 2022. ChatGPT. https://openai.com. Accessed : 2023-01-27.

[33] Helge Ostertag and Jens Drögemüller. 2012. Terra Mystica. Board Game.

[34] Jaclyn Peiser. 2022. https://www.washingtonpost.com/business/2022/12/24/board-game-popularity/. *The Washington Post* (Dec 2022). https://www.washingtonpost.com/business/2022/12/24/board-game-popularity/

[35] Leonardo Tortoro Pereira, Paulo Victor de Souza Prado, Rafael Miranda Lopes, and Claudio Fabiano Motta Toledo. 2021. Procedural generation of dungeons' maps and locked-door missions through an evolutionary algorithm validated with players. *Expert Systems with Applications* 180 (2021), 115009. https://doi.org/10.1016/j.eswa.2021.115009

[36] Alain Pétrowski and Sana Ben-Hamida. 2017. *Evolutionary algorithms*. John Wiley & Sons.

[37] PurpleKingdomGames. 2020. Virtual Gloomhaven Board. https://vgb.purplekingdomgames.com/Creator. Accessed: 2023-01-27.

[38] Critical Role. 2015. Critical Role. https://critrole.com/. Accessed: 2023-01-27.

[39] Rommel Dias Saraiva, Alexandr Grichshenko, Luiz Jonatã Pires de Araújo, Bonfim Amaro Junior, and Guilherme Nepomuceno de Carvalho. 2020. Using Ant Colony Optimisation for Map Generation and Improving Game Balance in the Terra Mystica and Settlers of Catan Board Games. In *Proceedings of the 15th International Conference on the Foundations of Digital Games* (Bugibba, Malta) *(FDG '20)*. Association for Computing Machinery, New York, NY, USA, Article 112, 7 pages. https://doi.org/10.1145/3402942.3409778

[40] Noor Shaker, Julian Togelius, and Mark J. Nelson. 2016. *Procedural Content Generation in Games*. Springer. https://doi.org/10.1007/978-3-319-42716-4

[41] Flaming Fowl Studios. 2021. Gloomhaven: Forgotten Circles. Video Game.

[42] Klaus Teuber. 1995. Settlers of Catan. Board Game.

[43] Julian Togelius, Alex J. Champandard, Pier Luca Lanzi, Michael Mateas, Ana Paiva, Mike Preuss, and Kenneth O. Stanley. 2013. Procedural Content Generation: Goals, Challenges and Actionable Steps. In *Artificial and Computational Intelligence in Games*, Simon M. Lucas, Michael Mateas, Mike Preuss, Pieter Spronck, and Julian Togelius (Eds.). Dagstuhl Follow-Ups, Vol. 6. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 61–75. https://doi.org/10.4230/DFU.Vol6.12191.61

[44] Julian Togelius, Georgios N. Yannakakis, Kenneth O. Stanley, and Cameron Browne. 2010. Search-Based Procedural Content Generation. In *Applications of Evolutionary Computation, EvoApplicatons 2010: EvoCOMPLEX, EvoGAMES, EvoIASP, EvoINTELLIGENCE, EvoNUM, and EvoSTOC, Istanbul, Turkey, April 7-9, 2010, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 6024)*, Cecilia Di Chio, Stefano Cagnoni, Carlos Cotta, Marc Ebner, Anikó Ekárt, Anna Esparcia-Alcázar, Chi Keong Goh, Juan Julián Merelo Guervós, Ferrante Neri, Mike Preuss, Julian Togelius, and Georgios N. Yannakakis (Eds.). Springer, 141–150. https://doi.org/10.1007/978-3-642-12239-2_15

[45] Valtchan Valtchanov and Joseph Alexander Brown. 2012. Evolving dungeon crawler levels with relative placement. In *Fifth International C* Conference on Computer Science & Software Engineering, C3S2E '12, Montreal, QC, Canada, June 27-29, 2012*, Bipin C. Desai, Emil Vassev, and Sudhir P. Mudur (Eds.). ACM, 27–35. https://doi.org/10.1145/2347583.2347587

[46] Guido van Rossum. 2007. Python Programming Language. In *Proceedings of the 2007 USENIX Annual Technical Conference, Santa Clara, CA, USA, June 17-22, 2007*, Jeff Chase and Srinivasan Seshan (Eds.), Vol. 41. USENIX, 1–36.
[47] Games Workshop. 2020. Warhammer 40,000. Miniature Wargame ninth Edition.