A Generative Grammar Approach for Action-Adventure Map Generation in *The Legend of Zelda*

Becky Lavender[1] and **Tommy Thompson**[2]

Abstract. In this paper we present an approach towards procedural generation of maps for 2D action-adventure games akin to those found within the classic Nintendo series *The Legend of Zelda*. Maps are generated through courtesy of a two-phase constructive approach, expanding upon existing research in the composition of missions and spaces for the action-adventure genre. Having completed the'mission graph' in the first phase, we are reliant upon a constraint-based approach to build geometry that faithfully represents the original mission structure. We investigate the effectiveness of this approach and the playable levels it can generate by creating dungeons within the open source game *The Legend of Zelda: The Mystery of the Solarus*.

1 Introduction

Procedural content generation (PCG) is a popular problem area in both games research and development given the opportunities present in crafting artefacts for player consumption. However, there are risks when building in-game spaces such as levels or maps that require emphasis to be placed not just on consistency and validity of the content, but the context which dictates players' exploration and sense of progression. The generation of 'quests' - stories or structure that dictate context - alongside maps has been suggested as means to realise the broader goals of PCG research and expand the potential of the field [24].

In this paper we summarise a body of work aimed at using the aforementioned two-tier approach to overcome problems inherent in action-adventure dungeon generation. By adopting the model of mission structure as distinct from the physical interpretation of the in-game world, we can build each component independently of one another. The contributions of this paper are aimed at highlighting the effectiveness of decoupling the challenge and macro-level puzzle structure challenges players face from the topology of the in-game environment.

We begin this paper with a short introduction to the action-adventure genre and the *Zelda* series in particular in section 2. This is followed by an investigation of existing research in the area of action-adventure games and the challenge of procedural generation for connected gameplay spaces such as dungeons in section 3. We provide an overview of our two-phase generative system in section 4 followed by an analysis of its effectiveness against a series of metrics in section 5, concluding with some final thoughts on the current state of the work and future avenues for this research.

[1] University of Derby, Derby, UK, email: becky@beckylavender.co.uk
[2] Anglia Ruskin University, Cambridge, UK, email: tommy@t2thompson.com

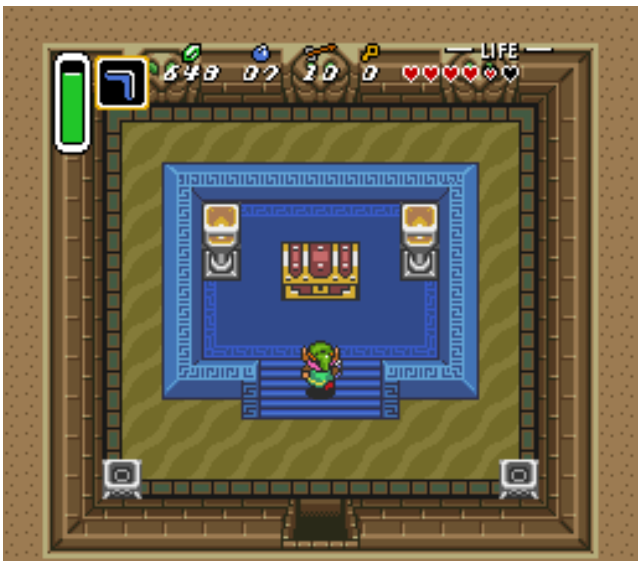

Figure 1: A dungeon room in *The Legend of Zelda: A Link to the Past* [15], where a special item can be found in the treasure chest.

2 The Legend of Zelda & Action-Adventure Games

The action-adventure (AA) genre is distinct in the taxonomy of video game genres given that it is typified by use of mechanics and tropes typically found in action games in addition to adventure titles. The core underpinning of AA games is adopted from adventure games such as *Zork* [9] and *The Secret of Monkey Island* [12] in which players must explore carefully crafted environments, typically requiring the solving of puzzles in order to progress to new areas. However, these mechanics are supplemented by the use of combat mechanics from action games; providing the player with means to attack enemies in the game world as well as finite resources to ensure their own survival. This typically requires dynamic and responsive control of the player's avatar in order to maintain a smooth gameplay experience. Despite the combination of elements from these two subsets, the AA genre is arguably one of the broadest found in all of gaming given the range of possible games that can be achieved by merging these genres. The AA genre is popularised in modern gaming courtesy of titles such as *Tomb Raider* [4], *Resident Evil* [3] and *God of War* [19]. However, as a genre in itself, it has been largely established by tropes and mechanics found within *The Legend of Zelda* [18].

The Legend of Zelda is a series of games developed by Nintendo starting with the first title released in 1986. The series places emphasis on players' progression through several enclosed areas or 'dungeons': constrained groupings of rooms and corridors comprised of

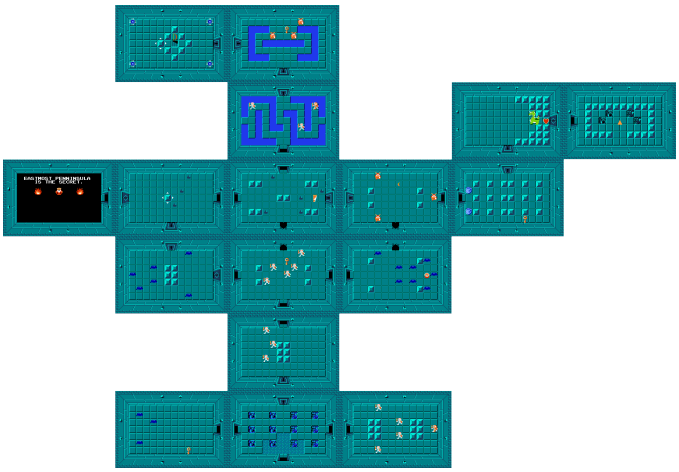

Figure 2: The topology of the first dungeon players visit in *The Legend of Zelda*. The player enters from the central square in the bottom row and must progress in a non-linear fashion to the top-right corner.

monsters and treasures [7]. As shown in figure 2, dungeons in *Zelda* are typically structured into rooms, with specific patterns of gameplay using locked doors, switches, keys and traps. A full list of dungeon-specific tropes that the *Zelda* series adopts can be found in [5]. Each dungeon contains three key elements that are now established series tropes:

A Special Item: A hidden treasure is guaranteed to be found within each dungeon that provides the player with new-found agency.
Item-Specific Puzzles: Challenges hidden throughout the dungeon that can only be solved after the player has found the special item.
Boss Monster: A dungeon can only be completed once the boss monster has been defeated. This monster will typically be found in a locked room requiring a specific 'boss key' to unlock it.

3 Related Work

As research in procedural content generation has continued to grow, the range of topics it aims to address has gradually diversified. While level generation is not a new topic - with arguably the research based upon the 2D platforming game *Super Mario Bros.* [17] being the most widely recognised [20] - research into AA games and specifically dungeons has only recently gained traction. As detailed in section 2, AA games carry their own qualities that would distinguish them from 2D platforming games. As such, we focus our review solely on methods relevant to the task at hand.

3.1 Grammar-Driven Generation

One means by which to achieve procedural content while adhering to structures or constraints can be found in the use of generative grammars. Grammars allow for the use of a structured set of symbols with rules that not only dictate how the grammar can be adopted, but also enforce constraints to ensure consistency. This approach has proven popular in the field of narrative generation, with notable works such as *Facade* [13] and *Prom Week* [14] and parallel research in the creation of quest storylines detailed in [10].

Our focus is on work detailed in [6], in which the author identified that AA levels can be divided into two distinct structures: the 'map'

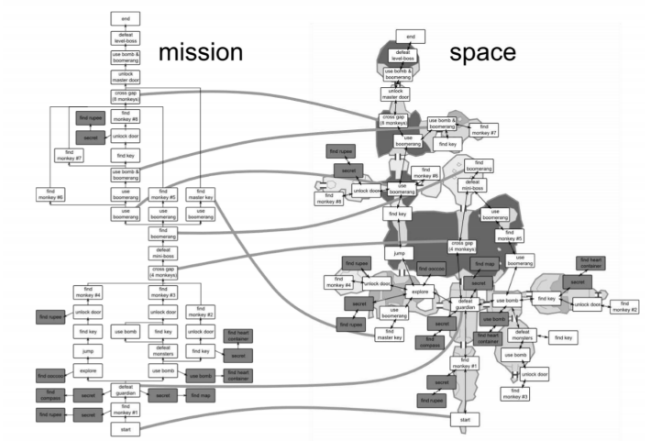

Figure 3: Dormans mission and space diagrams of the 'Forest Temple' in *The Legend of Zelda: Twilight Princess* from [6].

and the 'mission'. The former is the geometric shape and topology of the gameplay space whereas the latter represents the ordered sequence of tasks the player must complete within said space. As shown in figure 3, the mission structure can be modelled as a graph. While this particular mission is derived from assessing a specific location from *The Legend of Zelda: Twilight Princess* [16], it is not dependent on the map from which it was derived; allowing for the creation of linear or non-linear missions to be subsequently transposed onto linear or non-linear maps. An example of how a high-level dungeon structure could be generated is given in [6], with rules (figure 4a) and an alphabet (figure 4a).

The final alphabet for the mission grammar, shown in figure 5, combined with the series of rules for transforming non-terminal (upper-case) literals to terminal (lower-case) in [6] allows for a range of mission graphs to be established. This work follows with a subsequent explanation of a shape grammar for creation of maps, using the grammar in figure 6 as a basic example. Figure 6a shows an alphabet consisting of an unresolved connection, a door, and a wall respectively. Figure 6b describes rules which determine what any unresolved connection can be replaced with, and figure 6c displays a possible map result which could be generated using this grammar. Dormans later applied these maps to a 2D game [1], but not with the typical tiled room layout found in *The Legend of Zelda*.

3.2 Dungeon Generation

The generation of dungeons is one of the first examples of procedural content generation in commercial games with *Rogue* [25]. Since then algorithmic construction of dungeons has continued to be adopted in modern game development such as *Diablo III* [2]. However from an academic perspective, research in dungeons is still in its relative infancy. A notable contribution is the *MiniDungeons* domain, originally designed for modelling decision-making habits of players [8]. It has subsequently been adopted as a test-bed for generative approaches in dungeon creation [11]. However, recent research in establishing core design patterns of dungeon construction is aimed at fostering further research in this area [5].

At present, procedural generation research of action-adventure maps that adhere to the design tropes of *The Legend of Zelda*, is still relatively unexplored academically. To-date, the sole other body of research in this area is focussed on the adoption of bayesian networks aimed at generalising level topology structure having been fed

1. Dungeon → Obstacle + treasure
2. Obstacle → key + Obstacle + lock + Obstacle
3. Obstacle → monster + Obstacle
4. Obstacle → room

(a) Rules.

1. key + monster + room + lock + monster + room + treasure
2. key + monster + key + room + lock + monster + room + lock + room + treasure
3. room + treasure
4. monster + monster + monster + monster + room + treasure

(b) Results.

Figure 4: Dormans' high-level example of how a grammar could be used to generate dungeons. [6]

Alphabet:

bl = boss (level)	G = Gate	l = lock
bm = boss (mini)	g = goal	lf = lock (final)
C = Chain	H = Hook	lm = lock (multi)
CF = Chain (Final)	ib = item (bonus)	n = nothing/exploration
CL = Chain (Linear)	iq = item (quest)	S = Start
CP = Chain (Parallel)	k = key	t = test
e = entrance	kf = key (final)	ti = test (item)
F = Fork	km = key (multi piece)	ts = test (secret)

Figure 5: The mission grammar detailed in [6].

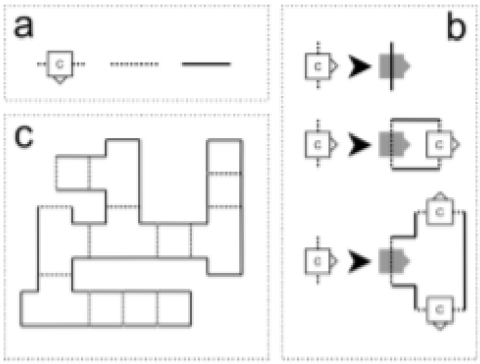

Figure 6: The shape grammar detailed in [6].

existing *Zelda* levels [23].

4 System Design

In this section we give a brief overview of the PCG system adopted for this experiment. We begin by replicating the mission and space grammars denoted in [6] in order to build our generator. This proves ideal given that these grammars are designed to reflect traditional *Zelda* dungeon structures. However, several issues were encountered when attempting to transcribe the space grammar approach detailed in [6] directly to map generation for a 2D tiled game.

4.1 Mission Generation

Here, Dormans' described method of using graph grammars to create dungeon missions was followed exactly. A graph with nodes representing rooms of different types (eg. entrance, key, lock) was iterated on according to the rules prescribed in [6], with sections of the graph being swapped out accordingly. A variety of mission graphs were produced, such as that shown in figure 7.

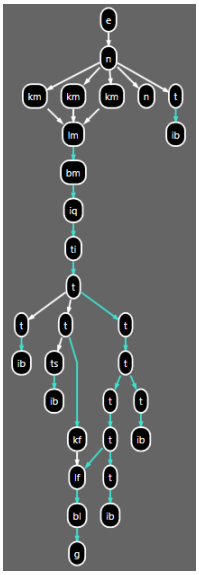

Figure 7: A completed mission graph.

4.2 Map Generation

Here, the general shape grammar approach described in [6] was used as the basis to lay rooms out in a grid, similar to the dungeons in *The Legend of Zelda: A Link to the Past* [15]. Each room type as described from the original mission grammar has one or more corresponding room templates built within the Solarus engine. When generating a

(a) Test Item (ti): Bombs are required to reach certain items as well as use secret doors.

(b) Item Quest (iq): Treasure room with a specific dungeon item inside.

(c) Multi Lock (lm) Room. Four monsters (keys) must be killed in the dungeon for the lamps (locks) to light and the door to open.

Figure 8: Interpretations of specific mission items from figure 5 to room templates that are used as part of the generated maps.

specific room type one of these rooms would be selected. (See figure 8). The maps produced were applied to *Mystery of Solarus* [22] : an open-source game designed to replicate the mechanics and aesthetic of *The Legend of Zelda: A Link to the Past* [15].

Some additional constraints to Dormans' original approach were needed for this particular generation approach to work, and were added to the dungeon generator. These included:

Compromised Structure: Some rooms need to be placed consecutively in a chain for mission structure to make sense. To honour these chains while preserving the layout of keys and locks so the map was completable, it was necessary to use a combination of topological sort and depth-first expansion of the grammar traversal when generating dungeons.

Dead Ends: When rooms were placed in a grid, a consecutive chain of rooms might hit a dead end. Therefore, a process was implemented in which the generator tested formations, reserving spots for every room in a chain before actually placing any down.

However, our implementation still imposes some limitations. For example, all rooms had to be of a fixed size and shape in order to fit the generation grid. Furthermore, it was necessary to design some rooms symmetrically, given our inability to pre-empt which direction the player would come from or be headed to.

5 Analysis

Given the focus of this research to evaluate the feasibility of the map and shape grammars, we elect to assess the expressive range of the generator. This is achieved by adopting metrics discussed in [21], and assessing the range of content that can be produced. For our dungeon generator, we establish four key metrics:

Mission Linearity: A continuum ranging from highly-linear missions to those with a high branching factor.

Map Linearity: A continuum ranging from highly-linear maps to those with a high branching factor.

Leniency: Scale of the danger represented to the player throughout the map: ranging from maps with zero threat to player to maps where each room poses a threat.

Path Redundancy: Scale measuring the number of rooms with 'purpose': either presenting players with reward/information or leading them to one that does.

In accordance with the expressivity studies conducted in [21], we provide representative samples of 1000 generations for each test. Our tests were focussed on a series of predefined rule-sets which help dictate how certain aspects of map design are considered in the generator. These are summarised in Table 1.

Table 1: Rule sets to control specific aspects of the mission and map generators.

Mission Rules	
Rule Set	**Description**
Branching Missions	Favours branching rules with multiple paths of actions.
Linear Missions	Favours linear rules with a single path of actions.
Long Missions	Favours longer rules containing more actions.
Short Missions	Favours shorter rules containing less actions.
Control	Balances all parameters described above.
Map Rules	
Rule Set	**Description**
Few Exits	Generates a more linear map layout: one exit rooms are given the highest weight.
Many Exits	Generates branching map layouts: 3-exit rooms are given the highest weight.
Control	Balances all parameters described above.

Mission linearity was found to be limited to central values, with few results at the high or low end of the spectrum. This can be explained when looking at the generation rules used. For example, the main mission flow rules dictate that some forks are mandatory, so a completely linear mission is impossible to achieve. Leniency of missions tended towards the more difficult but this can also be explained by the ruleset used: there were 16 hazardous room templates and only 9 safe ones.

Meanwhile, one of the most prominent biases found in map generation was high map linearity. This can be explained when analysing the generation algorithm used. For example, even if branching rooms with 3 exits are always chosen, its more likely that these will be closed than successfully link to another room; either because they were next to another wall or because they remained unconnected at the end of the generation process. In general, our expessivity results were successful in that appropriate patterns were seen: more linear rulesets produced more linear results, and so on. There were some gaps in the expressive range but this was to be expected as only one base ruleset and alphabet were tested. Our mission generation analysis suggests that gaps were due to limitations of the sample grammar, not the generation algorithms themselves.

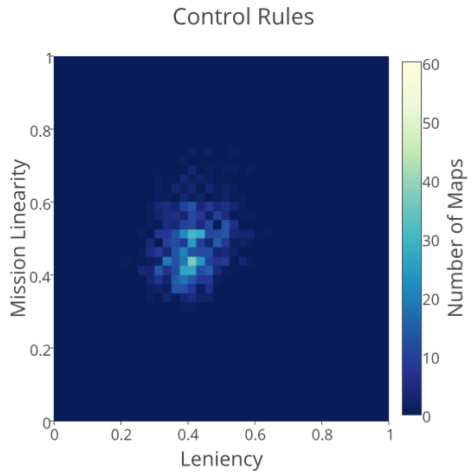

Figure 9: One of the 2D Histograms used to analyse the expressivity of 1000 dungeons. This graph compares mission linearity and leniency with a normal control set of rules.

6 Conclusion

In this paper we have highlighted the use of a two-phase generative system that enables dungeon generation for action-adventure games. This is achieved by adopting existing research detailed in [6] where we distinguish the overall gameplay experience or mission from the actual environment that this experience takes place within. Ultimately, this is aimed at providing context for the players actions. As detailed in section 5, the system maintains a reasonable expressive range as modelled through metrics defined in [21] albeit constrained by the rules of the grammar itself. The final maps built by the generative system are completely playable within the open-source *Mystery of the Solarus* game. The given method of ensuring that a core mission or structure is retained to the gameplay experience could yield interesting effects in other problem domains.

Given our use of the expressivity metrics detailed in [21], it would be interesting to compare our maps against those from actual *Legend of Zelda* games. While this would be relatively easy to achieve when considering the maps themselves, this would require more effort to develop means to reverse-engineer the missions within them: easily a project in itself. Furthermore, at present one major limitation of our generator is that it is reliant upon hand-crafted rooms within the Solarus engine. It would be of interest to adopt principles from recent analysis of design patterns in dungeons [5] and approach pattern-driven generation to create unique permutations of rooms that adhere to the constraints imposed by the mission grammar. Lastly, the current implementation of our generator is reliant upon iteration of our own grammar/rule-driven system. This is certainly scope for the adoption of search-based procedural content generation that adopts a subset of these rules - not to mention the metrics employed in section 5 to act as fitness criteria.

REFERENCES

[1] Sander Bakkes and Joris Dormans, 'Involving player experience in dynamically generated missions and game spaces', in *Eleventh International Conference on Intelligent Games and Simulation (GameOn2010)*, pp. 72–79, (2010).
[2] Blizzard Entertainment. Diablo III. Blizzard Entertainment, 2012.
[3] Capcom. Resident Evil. Capcom, 1996.
[4] Core Design. Tomb Raider. Eidos Interactive, 1996.
[5] Steve Dahlskog, Staffan Björk, and Julian Togelius, 'Patterns, dungeons and generators', in *Proceedings of the 10th International Conference on the Foundations of Digital Games*, (2015).
[6] Joris Dormans, 'Adventures in level design: generating missions and spaces for action adventure games', in *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*, p. 1. ACM, (2010).
[7] Gygax, Gary and Arnseon, Dave and Mentzer, Frank. Dungeons and Dragons Set 1: Basic Rules [Role-playing Game], 1983.
[8] Christoffer Holmgård, Antonios Liapis, Julian Togelius, and Georgios N. Yannakakis, 'Generative agents for player decision modeling in games', in *Poster Proceedings of the 9th Conference on the Foundations of Digital Games*, (2014).
[9] Infocom. Zork. Infocom, 1977.
[10] Boyang Li and Mark O Riedl, 'An offline planning approach to game plotline adaptation.', in *AIIDE*, (2010).
[11] Antonios Liapis, Christoffer Holmgård, Georgios N. Yannakakis, and Julian Togelius, 'Procedural personas as critics for dungeon generation', in *Applications of Evolutionary Computation*, volume 9028, LNCS, Springer, (2015).
[12] Lucasfilm Games. The Secret of Monkey Island. LucasArts, 1990.
[13] Michael Mateas and Andrew Stern, 'Façade: An experiment in building a fully-realized interactive drama', in *Game Developers Conference*, volume 2, (2003).
[14] Joshua McCoy, Mike Treanor, Ben Samuel, Aaron A Reed, Michael Mateas, and Noah Wardrip-Fruin, 'Prom week: Designing past the game/story dilemma.', in *Proceedings of the 2013 Foundation of Digital Games (FDG)*, pp. 94–101, (2013).
[15] Nintendo EAD. The Legend of Zelda: A Link to the Past. Nintendo, 1991.
[16] Nintendo EAD Group No. 3[a]. The Legend of Zelda: Twilight Princess. Nintendo, 1991.
[17] Nintendo R&D 4. Super Mario Bros. Nintendo, 1985.
[18] Nintendo R&D 4. The Legend of Zelda. Nintendo, 1986.
[19] SCE Santa Monica Studio. God of War. Sony Computer Entertainment, 2005.
[20] Noor Shaker, Julian Togelius, Georgios N Yannakakis, Ben Weber, Tomoyuki Shimizu, Tomonori Hashiyama, Nathan Sorenson, Philippe Pasquier, Peter Mawhorter, Glen Takahashi, et al., 'The 2010 mario ai championship: Level generation track', *Computational Intelligence and AI in Games, IEEE Transactions on*, **3**(4), 332–347, (2011).
[21] Gillian Smith and Jim Whitehead, 'Analyzing the expressive range of a level generator', in *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*, p. 4. ACM, (2010).
[22] Solarus. Zelda mystery of solarus dx, 2008.
[23] Adam J Summerville, Morteza Behrooz, Michael Mateas, and Arnav Jhala, 'The learning of zelda: Data-driven learning of level topology'.
[24] Julian Togelius, Alex J Champandard, Pier Luca Lanzi, Michael Mateas, Ana Paiva, Mike Preuss, Kenneth O Stanley, Simon M Lucas, Michael Mateas, and Mike Preuss, 'Procedural content generation: Goals, challenges and actionable steps.', *Artificial and Computational Intelligence in Games*, **6**, 61–75, (2013).
[25] Toy, Michael and Wichman, Glenn. Rogue, 1980.