DemiGen: Procedurally Generating Content for Dungeons & Dragons

Project Specification

By Ryan Murray

Supervised by Michael Gale

1 Problem Statement

With the advent of popular livestream shows such as *Critical Role* and *The Adventure Zone*, Dungeons and Dragons and its derivatives are experiencing a surge in popularity. An element of the game that has proven daunting to new groups is the generation of structures and locations (hereby referred to as "levels") for use in games.

Tabletop Roleplaying Games (TTRPGs) such as Dungeons & Dragons are often defined by three core elements; exploration, interaction, and strategy. All of these elements require the Game Master (GM) to create, describe, and narrate the events of a physical world in which these elements can take place. While its acceptable to loosely define a location as a number of rooms/substructures of vague sizing[1], many find it preferable to specifically define the dimensions and contents of locations instead. However, creating more intricate and interesting locations increases the workload on the GM, who is already also responsible for creating and controlling all non-player characters and the rest of the game world. This workload is generally not shared with the players, as many players find it preferable to play content they are not familiar with and can be surprised by. Aggravating this issue further, TTRPGs allow players a large degree of choice, meaning a GM cannot always predict where a party goes, nor what locations they choose to explore in detail.

This project aims to lessen the workload imposed on a GM without spoiling future developments for future players, and allowing for improvised locations that are both extensive and interesting. It aims to do so by providing software that randomly generates quality maps[2] of locations to explore via the 5th edition of the *Dungeons & Dragons* ruleset.

As *Dungeons & Dragons* has been popular in online spaces for decades, there exist a number of tools that try to fill this niche, and some video games that attempt to capture the experience of a classic dungeon crawl. However, the most popular examples have flaws that prevent them from being on par with user-generated content. Donjon's Dungeon Generator implementation[3] generates a series of rectangular rooms connected by uniform corridors. The

[1] an exercise referred to as "theatre of the mind"

[2] A "quality" map is defined in section 2.1.1.

[3] https://donjon.bin.sh/fantasy/dungeon/

only way in which themes are expressed in the architecture is the outer boundary within which those rooms are generated being marginally different. The rooms themselves are indistinct and uninteresting. No effort has been made to define any interesting progression blockers. *Rogue*[4] was an innovative cornerstone for procedurally generated dungeon crawlers, but neither it nor its derivatives necessarily translate well to good map generators for Tabletop Games. Similarly to Donjon, the rooms can best be described as rectangles connected by uniform corridors. However, it is notable for its use of constraints to create a minimum progression path. Dizzy Dragon's implementation[5] results in the most dynamic substructures. Maps are layed out in uniformly sized grid squares, each of which has at most two regularly spaced connections to each neighbour. While this implementation allows for arbitrarily large spaces, it also means that levels become predictable. Furthermore, while grid cells have their own theme, these themes appear to be placed arbitrarily, and as such the overall map is a mess of different shapes with no cohesive structure. As such, a user cannot specify a progression path for the map to create.

2 Objectives

2.1 Functional Objectives

- Create software that produces randomly generated quality maps of locations that can be used in *Dungeons & Dragons 5th Edition*
- Implement themes, specifying constraints and rules that generate maps that are identifiable as matching that theme
- Allow the user to impose constraints on the shape and content of the maps generated by the software
- Compare created software with freely available competing tools

2.1.1 Generating Maps for 5e

As the description of space in most TTRPGs can be left vague, the prerequisites for a map to be usable in 5e are very simple. At a minimum, a map in 5e clarifies the distance between characters, and indicates features such as walls or doors. A map can be anything from a vague diagram representing distances between creatures to fully realised locations, with elaborate traps, secret passages and varied terrain that may impact combat strategies. This project aims to provide software that generates maps on the more intricate end of the scale, as a number of free tools already provide generation of sparser, simpler locations (see section 2.1.4). For 5e, this generally means the inclusion of a number of standardised features including traps, secret entrances, and differing

[4] https://britzl.github.io/roguearchive/
[5] http://www.dizzydragon.net/adventuregenerator/gen

degrees of strategic cover. The first two elements are aspects of the exploration part of 5e, which generally follows the philosophy of punishing players that do not tread carefully, and rewards players that invest resources and time into improving their character's skills related to discovery and intuition. The use of strategic cover allows combat encounters to be more varied, making a fight more challenging if the enemy is better fortified, and rewarding careful positioning.

The map should also adhere to the conventions of good level design. As a minimum, a substantial critical path[6], opportunities for players to test their abilities, and a coherent structure that rewards intuitive play. Specifying a "substantial" critical path means trying to prevent situations in which the specified goal(s) of a level are trivial to reach; a GM that expects a dungeon to take at least a few hours to complete will not be pleased if the meaningful content in it is cleared within minutes. The range of skills a character in 5e can be made to possess is wide; from varying forms of combat to investigation to languages to craftsmanship. While expecting a level to allow all of these skills to come into play would be contrived, effort should still be made to ensure there is variation between all combat or all navigation. This allows players to build more diverse characters without worrying about not being an effective team member. A "coherent structure" in this case refers to a layout that is analogous enough to real world logic that the players can believe it can exist (suspension of disbelief). For example, if a level had a "treasure hoard" guarded by a dragon, it'd make sense that all the traps and guards would be between the hoard and the entrance, rather than being positioned to guard dead ends.

The creation of randomly generated levels that are constrained by a set of rules is a well-explored topic in video games. Such as the genre of Rogue-likes (inspired by the 1980 PC Game *Rogue*). Many of these games often make use of modular rooms designed by the developer, resulting in encounters that are generally more balanced at the cost of a decrease in variety. The patterns defined by these games are likely to be explored in the course of this project. Particularly, the pattern of having pre-defined important locations (entrances, goals, ect.) with the in-between being generated with some degree of randomisation.

2.1.2 Thematic Maps

The "Theme" of a map is defined as the set of conventions that map draws upon. Common themes in fantasy TTRPGs include dungeons, castles, and enchanted forests. These locations can have major variations, such as the presence of man-made structures in towns vs. the natural landscape of a forest. Or they can have very subtle structural differences, such as the barracks of a military base and the prison cells of a dungeon.

By allowing for the creation of distinct thematic locations, the software would provide more freedom for a GM to more readily adapt the content to their needs.

[6]Defined as the required actions that take a group from the start of the level to the "goal" of the level

2.1.3 User Constraints

In addition to allowing for different themes, the software may grant the user a degree of control over the generated content in a number of key areas; number of encounters (both critical and optional), number of progression locks, and the physical locations of key points of interest. These additions may allow the software to serve not only as a tool for random ad-hoc content, but also as an assisting tool for GMs designing more extensive and personalised levels.

2.1.4 Comparison to Existing Tools

The popularity of TTRPGs means that a number of tools have been created to try and answer this need for randomly generated content. This project aims to implement a quality alternative to fill a number of niches. Additionally, it may prove useful to compare against randomly generated dungeons as implemented in video games. Particular points of comparison include:

- donjon's Dungeon Generator
- *Rogue 1980*
- Dizzy Dragon's Geomorphic Dungeon Adventure Generator

2.2 Extensions

The aim of this project is to produce software that generates visually interesting and thematically identifiable grid-based maps. Many tools and games that aim to do the same thing also provide additional elements to enhance the atmosphere and character of their maps. The following elements could be added after the core functionality has been implemented;

- Thematic descriptions of substructures, either the result of a markov chain or chained together from manually written sentence fragments
- Enhancing the visuals of the map with appropriate colour and decoration
- Thematically described and structured objectives and non-player characters

3 Methodology

3.1 Software Development

The proposed software will be broken down into a number of components, where the aim is to develop each component sequentially within a given time frame. While it is likely that some backtracking to debug and refactor earlier components will be required, the benefits of focusing on one component at a time, with a set of tests for each, will be valuable.

As such, the most appropriate development cycle for this project would be the Iterative Model.

The initial plan for this software's individual components is as such: a substructure generator, a system for specifying how substructures are generated, a system for connecting substructures, and User Interface elements allowing for easy generation of maps. These components are described in more detail throughout Section 4.

3.2 Testing & Assessment

While the software will need to be automatically tested for functionality, there are a number of features that require outside assessment.

3.2.1 Automated Evaluation

Some features of the software can be assessed objectively; such as sufficient handling of errors, prevention of constraint-breaking results and adherence to good coding practices. Requiring a sufficient pass rate for a test suite before software is considered finished is referred to as test-driven development. For each component, a suite of tests shall be constructed, which the software will have to pass before development of that component can be considered minimally complete. This series of tests is to be decided and created as one of the first steps of a component's development cycle.

3.2.2 User Assessment

The most effective way to measure the effective implementation of ideas such as thematic adherence and structural coherency would be to assess their effectiveness using feedback with the target audience. Such feedback questions would test the software's ability to perform the following;

- Produce maps that are visually identifiable as adhering to a certain theme
- Produce substructures in which a diverse range of encounters can be designed
- Produce maps of comparable quality to hand-crafted maps of comparable size

This feedback will be collected from anonymous volunteers from online Tabletop Game communities. They will be asked to perform some simple tasks; attempting to discern DemiGen maps from hand-created maps[7], and assigning a perceived theme to a number of DemiGen maps that have been generated using theme settings[8]. This will require seeking approval from the Department of Computer Science to ensure the tests are ethical.

[7] Both of which shall be rendered using the same tileset

[8] Once again these maps will use a neutral tileset to eliminate guesswork based on factors beyond the software's generation.

3.2.3 Supervision

The student and supervisor will meet at a minimum of once per week for reflection on current progress. Feedback will also be provided on future development plans to alleviate issues associated with iterative development such as feature creep.

4 Timetable

The timeline for this project is for the most part arranged into two week blocks. The first of which will be a general research block, followed by a number of "development" blocks, and concluded with an overall evaluation as of to the effectiveness of the final software. The general structure of a development block is as such;

- Refinement of the goals of that iteration
- Development of a test suite for that iteration
- Development of that iteration, towards meeting the minimum criteria
- Evaluation and refinement of that iteration

Week 1 - Research
Some areas of research include the tools DemiGen will be competing with, and procedural generation as implemented in the video game industry. Of particular note is a novel form of generating 2d spaces from constraints is an algorithm based on the quantum phenomena Wave Function Collapse[2].

ID	Objective
1.1	Make extensive notes on the features provided by competing tools/game implementations[3]
1.2	Research procedural generation patterns in video games development, including popular patterns in the current industry[1]
1.3	Experiment with implementation of procedural generation patterns
1.4	Decide upon the exact tools and libraries to be used in development

Week 2 & 3 - Substructure Generation
Preliminary investigation into methods of procedural generation have lead to the conclusion that likely best course of action would be to treat the generation of each substructure as a separate entity. For this reason, the first iteration of DemiGen will produce structures from a set of rules and a tileset. The idea is for these substructures

to be seeded across a map, after which connections between them will be generated procedurally.

ID	Objective
2.1	Define data structures to store maps and intermediate data
2.2	Make final decision on structure of overall map generation pipeline
2.3	Implement first prototype of substructure generation algorithm

Week 4 & 5 - Biome Generation Substructures will be generated by a set of constraints based on the "biome" they occupy. Biomes are a pattern popular in procedural generation for video games. A biomes are regions of space between which the constraints dictating procedural generation differ. A prominent example would be their implementation in *Minecraft*. For example, desert biomes are covered almost entirely in flat sandy planes, whereas forest biomes are characterised as having thick vegetation including trees and flowers.

An example of biomes between which the rules for procedural generation should differ in DemiGen could be the dungeon cells of a fortress and its treasure vault.

ID	Objective
3.1	Define a set of example biomes
3.2	Decide on an algorithm for biome generation
3.3	Implement first prototype of biome generation algorithm
3.4	Integrate with Substructure generation layer
3.5	Conduct extensive testing with consideration to edge cases

Week 6 & 7 - Substructure Connection
The final stage of the procedural generation pipeline is ensuring the generated substructures have a reasonable level of connectivity. This stage may also introduce the possibility of hidden paths, traps and dead ends.

ID	Objective
4.1	Decide on an implementation of substructure connection
4.2	Implement substructure connection
4.3	Assess data structure for implementing constraints on biomes
4.4	Conduct extensive testing and tweak based on results

Weeks 8, 9 10 & Winter Break - Implementation of Controls & Content Creation

The final stage of the software development will be the implementation of a User Interface that allows for control over aspects of the generated map such as room placement, theme, and size.

With the structures for defining rules for biomes, it should be possible to implement with reasonable speed a number of example map themes, such as dungeons, cave systems, ect.

ID	Objective
5.1	Implement a rudimentary UI allowing for user constraints
5.2	Generate a minimum of 3 example theme options
5.3	Perform extensive testing on (hopefully) finalised generation pipeline
5.4	Conduct testing on UI

Week 11 & 12 - User Feedback

With the aim of having a stable prototype by the start of second term, the effectiveness of DemiGen will be tested based on user feedback from a sample of the target audience. By reaching out to online TTRPG communities, anonymous users will be asked to identify the "theme" associated with a sample of generated DemiGen maps. They will also be asked to discern generated maps from hand-crafted maps. The effectiveness of DemiGen will be measured based on how easy it was to tell what a map is supposed to look like and how hard it was to discern pre-generated maps from hand-crafted ones.

The UI element of DemiGen will be tested based on feedback from a number of university students, who will be asked to provide qualitative feedback on the ease of use and clarity of options.

ID	Objective
6.1	Select a number of hand-crafted maps and recreate with whatever tileset DemiGen is generated from.
6.2	Devise an online test to assess the effectiveness of DemiGen
6.3	Attain an acceptable number of responses to the aforementioned test
6.4	Collect feedback on UI elements

Weeks 13 & 14 - Feedback analysis and Report Draft

With the collection of user feedback, the effectiveness of DemiGen can be gauged, allowing for conclusions about the overall project to be drawn. With development finished, this means that work can

begin towards drafting the final report. This report will be broken down into chapters roughly correlating to the sections outlined in this timetable;

- Preliminary Research
- Substructure Generation
- Biome Generation
- Substructure Interconnection
- Thematic Presets
- Feedback & Conclusion
- Further Work

ID	Objective
7.1	Analyse feedback and conclude on software's effectiveness
7.2	Compare DemiGen's capabilities to those of its competitors
7.3	Outline final report
7.4	Draft first chapters of report

Weeks 15 & 16 - Report Draft Continuation
With the presentation at the end of Term 2, it would be beneficial to have completed a draft of the report by this point.

ID	Objective
8.1	Draft rest of report
8.2	Seek feedback on drafted chapters
8.3	Begin second draft based on feedback

Weeks 17& 18 - Presentation Preparation Time will be set aside for preparing the project presentation. This will be done late in the term and after the first draft to ensure ample analysis has been done allowing for an informative and effective overview of the project.

ID	Objective
8.1	Draft and finalise presentation
8.2	Seek feedback on second draft of report
8.3	Alter report based on feedback

Weeks 19 & Beyond The remaining time will be used to ensure the report is to a good standard. Ideally, the final draft will be complete with a few weeks remaining before the deadline. With some last minute feedback, the report will be complete.

5 Resources

The plan is for this software to be written in Haskell. It should not require any specialised hardware, nor any software that is not freely available. While an amount of data collection shall occur, this will pertain only to comparisons between DemiGen generated content and freely available resources for tabletop games. As such there will be minimum need to consider legal questions such as issues of copyright and data privacy laws.

As for project management, liaison with the supervisor will occur both in person and over Slack, progress will be tracked with a simple spreadsheet. Version control for the software will be handled with Git, with regular uploads to GitHub. Additionally, major versions will be backed up to external memory storage mediums.

The Bibliography section contains a number of texts that have been considered as starting points for the research for this project.

References

[1] M. Hendrikx, S. Meijer, J. Van Der Velden, and A. Iosup. Procedural content generation for games: A survey. -, 2011.

[2] I. Karth and A. M. Smith. Wavefunctioncollapse is constraint solving in the wild. *Proceedings of FDG'17*, 2017.

[3] R. van der Linden, R. Lopes, and R. Bidarra. Procedural generation of dungeons. *IEEE Transactions on Computational Intelligence and AI in Games*, 6, 2013.