

Lindenwood University

Digital Commons@Lindenwood University

Theses

Theses & Dissertations

7-8-2022

Hexgen Roguevania

Conrad Garcia

Follow this and additional works at: <https://digitalcommons.lindenwood.edu/theses>



Part of the [Game Design Commons](#)

HEXGEN ROGUEVANIA

By

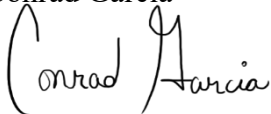
Conrad Garcia

Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Arts in Game Design
at
Lindenwood University

© July 8, 2022, Conrad Garcia

The author hereby grants Lindenwood University permission to reproduce and to distribute publicly paper and electronic thesis copies of document in whole or in part in any medium now known or hereafter created.

Author	Date
Conrad Garcia	7/8/2022



Committee Chair	Date
Ben Fulcher	7/8/2022



Committee Member	Date
Jerimiah Ratican	7/8/2022



Committee Member	Date
Michael Fetters	7/8/2022



HEXGEN ROGUEVANIA

A Thesis Submitted to the Faculty of the Art and Design Department
in Partial Fulfillment of the Requirements for the
Degree of Master of Arts in Game Design
at
Lindenwood University

By

Conrad Garcia

July 8, 2022

Abstract

Title of Project: Hexgen Roguevania (Working title)

Conrad Garcia, Master of Arts Game Design, 2022

Project Directed by: Ben Fulcher, Instructor, Game Design

This project is inspired by procedural generation methods, *Metroid*, *Castlevania*, *Rogue*, and modern roguelikes. This project is structured around the notion of how procedural game content can be experienced when it is tied directly to player progression. The focus of this project incorporates procedurally generated content so that it can be altered during gameplay. In a sense, this idea brings level content to the player, instead of the player travelling to new zones. The resulting game world starts visually coherent, changing into a world that resembles a patchwork of different biomes crafted by the player's decisions to progress the game's difficulty.

Table of Contents

Abstract.....	iii
Introduction.....	1
Literature Review.....	2
Methodology.....	7
Results.....	9
Analysis.....	12
Conclusion	13
Bibliography	16

Introduction

This project aims to design a procedural tool for a 3D video game level generator that ties with player progression. Content of each level generates via pseudo-random distribution or set manually by a designer before runtime and then it is determined by conditions met by the player at each checkpoint in game. The implementation of 3D art content is pulled from a library of designer-set assets; allowing short iteration time and establishes tools for end-user modding features. Gameplay consists of platforming puzzles, where exploration is the main contributor to progression. Additional gameplay involves player vs. environment as well as item acquisition, which are all integrated within the procedural generation system.

Based on my professional experience in game design, along with peer-reviewed research in topics of procedural generation, gameplay design, AI design, puzzle design, and other similar areas, the final deliverable produced will demonstrate the core behavior of level generation and player influence on the level as they progress, serving as a proof of concept that this project focuses on. Throughout early prototyping, the gameplay and game space create a unique and distinct experience while simply navigating platforms. Additional gameplay elements such as enemy encounters, item discovery, and obstacles are incorporated within the procedural generation framework, serving as a demonstration on how any feature can be integrated into the system, and that any gameplay element can be turned on or off, allowing for further experimentation for designer and modders.

Literature Review

Procedural Content Generation

There are a significant number of publications from scholars and experts of the field offering insight into the different methods on approaching procedural content generation (PCG) for video games. These approaches focus on world creation, art creation, and gameplay while others posit on the planning of systems to identify production issues, such as algorithm complexity, runtime performance, and how each system mixes. The design decision regarding PCG for *Hexgen Roguevania* (working title) is in combining several PCG methods to allow a clear representation of player progression throughout the game world and gameplay experience as well as “to relieve the designer of the scutwork in ways that create more opportunities for creativity.” (Parberry 2018).

Procedural world generation for game spaces use algorithms to plot a two- or three-dimensional collection of data (usually vector data) for visual plotting. An investigation by Mawed (2019) categorizes at least seven approaches, they are: (1) random numbers, (2) perlin noise, (3) filling space, (4) sequence generation, (5) partitioning space, (6) maze generation, and (7) procedural flowers, and these can be intermixed with one another. This investigation is 140 pages in length and admits to barely addressing the myriad other ways to develop PCG systems. While these plentiful methods can create content alone, many do so with a degree of predictability or recognizable pattern. This can be adequate, even preferred, but by mixing algorithms, a system can be achieved which is random enough to create a fresh, playable game space every time, while also supporting the designer’s goals for gameplay. A presentation from Georgia Tech (2019) and the research by Parborg and Holm (2018) demonstrate how just a couple of algorithms can generate an exponential number of new outcomes. This expansive

growth makes it harder to predict and control, as it is now each algorithm working towards a complex outcome. In other words, not all parameters will work because certain value combinations produce unplayable game spaces. This is unavoidable; therefore, limitations must be defined either by rejecting generated results or restricting algorithm possibilities, which is what Vegricht (2018) proposes by using evolutionary algorithms. In a case study by Nitsche, et. al. (n.d), the use of procedural generation creates a game space that utilizes each algorithm as much as possible and attempts to synergize the design goals with their outcomes. Their result is a tile-based world with gameplay inherent to each tile, but also from the proximity of different elemental tiles (metal, fire, toxic, etc.), generating unique enemies and complex strategy. Consequently, defining the primary gameplay needs *first* identifies the idyllic algorithms. Furthermore, deconstructing the right algorithms and modifying them to address the specific needs for the PCG system narrows generic algorithms into a realized heuristic.

The popularity of procedural generation techniques has also found its way into art creation throughout various stages of production and mediums. This application differs from the generation of architectural spaces and level layouts, as it concerns over the specific creation of a given art asset, either offline or during game runtime (online). Some engine tools and standalone software can operate completely by procedural definition from its user. Houdini and Substance Designer are two such software, both utilizing node-based construction for producing visuals. For example, in Houdini, a staircase, comprised of steps, railings, lips, and undersides can be defined using nodes and then further adjusted to change the number of steps, curvature or spiraling, width, dimension of individual steps, and so on. This is all based on the needs of the asset, and it is worth defining such a system for production (a game with a lot of stairs would benefit immensely). In a workflow study by Cornelisse (2018), procedural concepts are utilized

to define a texture that can be adjusted for the same character model, yielding an assortment of variants useful for signifying different role-playing game (RPG) enemy types or offered as alternate character skins in a massively multiplayer online (MMO) game. The same technique can be applied to environment and prop assets, even visual effects. In the academic project *Spareware* (Helin, 2016) and in the commercially successful *XCom* series (Hess, 2018), the use of procedural art creation is abundant through all facets, ranging from texture reskinning to rulesets defining the placement of assets, enemies, and items. In all cases, art quantity is needed for the sake of gameplay, and all are achieved in their own way by first developing the tools necessary for production. Although procedural software like Houdini and Substance Designer can generate countless variations, there is need for a human factor which can recognize aesthetic significance to merit inclusion. As with Cornelisse (2018), each texture variant is defined by the artist for aesthetic sake. This human factor maintains a cohesive art direction and theme, filters out unpleasant or undesired generation results, and minimizes memory overhead.

Visual and spatial construction use procedural rules in a predictable manner so they can be understood by the player or user; however, gameplay varies and for good reasons – it can get boring doing the same type of task based on similar rules and, most games have a story where gameplay needs to complement each act. Designing a procedural system for gameplay requires a different approach, but still utilizes algorithms. In the scenario where “the level is the gameplay,” one approach, by Thompson (2021), is to make the levels generate with cyclic paths, so the player returns from where they started once they’ve completed the objective. Thompson’s game, *Unexplored*, implements a lock-and-key system where the player finds the key to unlock a door leading them back to where they started. Zafar (2019) also utilizes a lock-and-key system by investigating pattern-based objectives to determine successful world generation including

objectives and all other gameplay elements. On paper, this sounds repetitive and to an extent it operates under the same ruleset each time, but it works for the RPG and dungeon-crawling styles of gameplay since puzzles are kept simple and the focus is on exploring new zones in search of enemies and items. More and advanced puzzles could be implemented, which is what Kegal and Haahr (2020) analyze in their survey of 32 puzzle methods that they converted into a procedural system. Their survey offers a compendium of ideas to pull from and apply to different genres of games such as a sliding puzzle for a mystery game, riddles for an RPG, assembly puzzles for a sim game, or physics puzzles for an action game. To a further degree, Zook (2019) proposes the use of data analysis during the play session to adapt the difficulty of challenge- and story-based gameplay to reflect the player's "perceived" skill level. These ideas for creating gameplay through procedural means allows for endless variations, but as with art or level creation, the gameplay should also consider the game as a whole. Each method has its place and has the potential to be combined or opened up to innovative means such as tracking player data to inform offline design, adjusting difficulty during runtime, biasing random number generation, or predicting player choice. However, none of this can be quantified without a gameplay prototype (a chicken or egg situation). Limits need to be measured and determined for any application to integrate the gameplay with the procedural system naturally.

Roguelike Game Design

At the core of every roguelike game is an assortment of procedurally generated content, decided on by the designer, such as level layout, enemy placement, and item effects. As Harris (2021) addresses, "instead of just random dungeons... items generated during the game are also randomly selected, and their appearance is scrambled each game." With the understanding of the varied PCG methods, items, in this explanation, are not just randomly spawned in the game

world, but their appearance is also random, making the effect of any item unpredictable. “Even in a winning game it is unlikely that the player will see all the items that can be generated,” (Harris, 2021). Again, by leveraging PCG methods, the core gameplay emerges as a “need to discover the game world anew every time,” (Harris, 2021). However, as Aron (2019) puts it, “roguelikes are not truly random – if they were, the game would be impossible to complete.” So, the implementation of PCG methods allows for discovery and randomness to occur each playthrough, but there must be guidelines or checks in place to generate content based on designer intentions, such as evolutionary algorithms as mentioned earlier. With each playthrough different from the last, due to pseudo-random generation logic, each player experiences the game uniquely.

Roguelikes offer a unique enough experience for each playthrough, where unpredictable generation challenges player skill, creating conflict, a core element to storytelling and game progression. Like RPGs, roguelikes focus on a story being told which benefits from the balanced randomness of encountered enemies and items. While the “player character’s back story is not a high priority,” (Brown, 2018) it is essential to involve the player within an active story that keeps players engaged in building their character throughout gameplay. In this manner, “think of roguelikes as storytelling machines,” (Aron, 2019) where completing the game is a legendary tale and dying causes “the destruction of the player character completely from the hard drive,” (Brown, 2018). “Spelunky, the king of modern roguelikes, is a near-perfect balance of order of chaos,” (Aron, 2019) and keeps deep lore aspects to a minimum, keeping the story simple and communicated through discovery and world art, rather than cutscenes or dialogue sequences.

The concerns for *Hexgen Roguevania* rely on procedural generation of the world, gameplay elements, enemy attributes, items, and upgrades are essential. Additional assets such as

animation and visuals effect could benefit from procedural use but are minor in quantity compared to the former. The world of *Hexgen* is one large hexagon, made up of individual hexagonal tiles varying in elevation. The formulas and concepts from Patel (2020) in working with hexagon generation has provided enough information to solve the generation of six, data-logged slices so they can dynamically change during gameplay, one of the core pillars behind my thesis. Games like *Minecraft* or *No Man's Sky* allow for editing of the procedurally generated world, but at a local level: by the player directly focused on it. *Hexgen* aims to replace large portions of the world with new content to denote player progression, with the goal of the game to progress as far as possible. This respects *Rogue* and the roguelike formula of which *Hexgen* is directly inspired by. The other core pillar is in player growth, taking the form of enemy combat and item acquisition, a gameplay model inspired by *Metroid* and *Castlevania*. This pillar boils down to a focus on simple combat encounters, platforming, and item progression. The former two are of particular concern since enemy AI must be smart enough to navigate a game space of varying elevations via jumping. According to Silva (2019), there are two methods for AI jump navigation in Unreal Engine, one utilizes the A* search algorithm, generating a navigation field for unpredictable navigation. However, this process can be computationally expensive, especially when performed multiple times and during the play session. The second method Silva suggests is the use of AI ray tracing to determine jump behavior. By considering the design needs for *Hexgen*, based on these resources, a viable prototype is developed, serving as the proof of concept for the core gameplay systems.

Methodology

By using Unreal Engine 4, the procedural generation system for *Hexgen Roguevania* can take advantage of its iterative prototyping tools and produce a playable proof of concept. Visual

scripting (Blueprints) will aid in the creation of procedural systems and management of assets such as meshes, materials, and data. Art assets will be imported and organized within the engine, primarily referenced through code and data assets. The look and quantity of assets are balanced for the initial workload required to achieve the core features of the procedural generation system. This is determined by macro and micro design concepts, generative aesthetics, and classifying elements of the system.

Hexagonal tiles are chosen as the “3D building blocks” for *Hexgen*’s exterior world space for its complement to platforming gameplay, broader navigation paths, generative aesthetic form and locus, and its uncommon use as a generated shape for game worlds. This comes from a mix of design and art needs, to which a technical system is meant to support. With the game world consisting of different exterior themes, the use of hexagonally shaped tiles emphasizes an organic, open-ended layout when compared to grid-based layouts. Since platforming is a core gameplay mechanic, the use of hexagon tiles offers more directional choices with similar distances apart, keeping the flow in player motion. One hexagon tile should be big enough to allow for comfortable movement and allow enough space to build up speed to jump across to another tile. This requires a thematically fitting tile much larger than the player, such as a large chunk of land, to be reused hundreds of times in populating entire areas of the game world. Karth (2019) defines how this *individual* generative aesthetic form can be achieved to also contribute towards the *gestalt* of varying elevations and the *repetition* of the same visual, providing enough variation that it can be quickly processed while performing in-game actions and keeps art tasks to a minimum. Additionally, generating with a balance between *gestalt*, *structure*, and *surface*, (Karth, 2019) allows the look, layout, and content of each biome in *Hexgen Roguevania* to be fully adjustable by the designer.

To address the visual and mechanical concerns of art and design, this procedural system must be approached with both in mind. Based on research from Hollstrand (2020), interfacing with such a system must be flexible enough to allow for expanded controllability to make changes manually, design sections, isolate changes, visualize metrics, and precisely target generated elements. For most of these findings, data tables suffice, but require a degree of design planning to address the production concerns and remain intuitive. Also, the use of data tables organizes the art workload by listing it in a table that can be referenced for generation usage. Separate from data tables is the logic which creates the tiles, and this is split into three categories: world layout, elevation differences, and unique tile data, derived from Hollstrand's (2020) reflection of macro and micro design. World layout concerns over tile size and shape, tile quantity, player tracking, collision, and runtime performance of generated elements around the player. Elevation differences highlight a common trait of all tiles and are performed as the second part after world layout by performing finer adjustments such as neighbor checking, data pooling, and serves as a granular entity allowing specific design control. Finally, unique tile data is split between groups of tiles and individual tiles. Player tiles fall into this category because they require specific rules different from the rest of the game tiles. Other tiles in this category are landmark tiles which occur out in the world, and boundary tiles, occurring along the edges of the world. Since enemies can generate in the world, spawners must generate pseudo-randomly, by quantity, difficulty, distance from player start, and elevation, and will be controlled using design tools.

Results

By using Unreal Engine as the primary creation software, along with a thorough understanding of roguelike game mechanics, procedural generation methods, and scholarly

research, *Hexgen Roguevania* accomplishes the goals set out in this investigation on varying degrees of success. The idea of player progression influencing PCG is achieved, but alternative design decisions make the possibilities endless; the abstract notion of progression and generation works at a fundamental level in game design, able to be applied to any genre.

The purpose of the PCG system for *Hexgen Roguevania* is to generate the game world, enemies and items, checkpoints, and obstacles. The world generates in tiles that are roughly hexagon shaped, fitting together like puzzle pieces, creating a stylized landscape with additional visuals such as trees, rocks, grass, and ruins, and at varying elevations. Each tile has logic for spawning itself with all associated content and additional functionality, for instance, checkpoint tiles and obstacle tiles. The spawn layout for tile generation is hand-authored through data tables, referencing spawn codes which have differing possibilities for generation decision-making. The last step for spawning world tiles adds obstacles and hazards to make navigation difficult. Once all world tiles have finished generating, enemies and breakable containers are generated based on table-defined probabilities and presets. These layers of the procedural world generation happen before the game starts, and when the player activates checkpoints the same functionality that generated the initial world is performed again to replace that portion of the game world with the next biome, raising in difficulty.

Platforming and roguelike mechanics were of great concern to incorporate into this project, requiring specific rules to how the world generates. One such rule concerns over landscape layout, where elevation differences and the space of a tile's playable area must naturally create platforming puzzles – aligning with roguelike design where the generate level must be explored in some fashion. Roguelikes typically involve a different mechanic than just jump, such as combat, item discovery, and inventory management, but *Hexgen Roguevania*

merges roguelike mechanics with platforming mechanics and keeps additional mechanics simple, focusing on progression and generation, as well methods of engaging gameplay, in this case, combat, item discovery, crafting, and discovery. These proposed features would inform the PCG system, allowing for parameters to control elevation and size, as well as spawn amounts, rarity, and obstacles.

The start of the project involved preparation and planning: optimizing Unreal Engine's build settings and plugins, setting up folder structures, naming conventions, reference libraries, and creation/management of documentation for technical and artistic purposes. A functioning prototype was the next step, where a mockup was first constructed to understand the kind of customization that would be needed for design purposes. For instance, the early mockup of the hexagon tile world was hand-authored with different platforming scenarios to test player movement, jumping, and air control situations. This showed how different scale settings changed navigation tactics and observations like these informed what to incorporate into the PCG system to allowing further designer adjustments as navigational rules such as movement speeds and jump height need to be fine-tuned throughout development. Another added feature to the PCG system is caching tiles into groups, which was also informed by the early mockup, which allows access to specific sets of tiles based on a grouping preference – in this case a hexagon-shaped world split in six slices and a center honeycomb section.

From testing and experimenting with the game's world size and tile size, as well as different platforming scenarios, the findings during early playtests gave insight to knowing each piece of the PCG system needed. Regarding world design, the generation of hexagon tiles had to make mathematic sense while appearing as a less fabricated landscape. In terms of gameplay, platforming and navigation depends on generation rules and designer-set parameters, allowing

enough customization to solve player and AI navigation throughout a dynamic landscape. And on the note of AI design, platforming is a much-studied area for 2D, and to an extent 3D, but 3D presents further complications, therefore it was decided to use a light, performance-friendly raycast check for several conditions while still benefiting from the navigation solving flexibility of Unreal Engine's navmesh. Finally, the management of all aspects of game implementation occurred with thoroughly updated design documentation, but just as importantly, the various design tables served to outline production work and allow for interfacing with the procedural system hooks.

The manner by which procedural content is influenced takes form of checkpoint-style interfaces that the player must find. This method was chosen to debug progression moments and adjust difficulty from measurable numbers. Checkpoints have requirements and if the player can meet all criteria, then it is activated and changes that section of the game into a new one. The experience is novel each time, and a different experience due to player choice and pseudorandom generation. In addition, the gameplay cycle is meant to be a short, but quick progression through biomes as dying would prompt starting over in a newly generated world.

Analysis

The core mechanic explores how content generation can be influenced by player progression and is done so in a unique way that has not been explored before. And while this core mechanic is successfully achieved, other approaches can be taken as this concept is open-ended to a broader concern. With procedural generation hooked up to the core pillars that allow for player influence, activation occurs based on the designer's preference, though in the case of *Hexgen Roguevania*, by progress, where difficulty increases when replacing previous content, decided on by the player. And due to the number of choices available throughout gameplay, the player could

be solving a greater puzzle as well – the order or combination of choices, for example. This core mechanic informs the PCG system of all gameplay needs and where they extend from; it's the same framework, but with additional features that can be replaced or removed.

The development of art assets for *Hexgen Roguevania* is a paradigm shift from the typical method of production, requiring iterative testing on several levels of art and gameplay quality. Due to the procedural placement of world meshes, rules for the procedural system as well as art creation had to be explored and documented – model size limits, collision complexity, AI navigation, and art needs, for starters. For models, the size limit was rather small, mainly due to AI navigation needing as many valid paths as possible, and for the most part, model height is usually short enough that it can be jumped on or over. With collision, simple shapes worked best to make quick adjustments throughout production and the quantity of collision throughout the world benefits from quick to process colliders. As mentioned, AI navigation has needs, and in fact has influenced many of the project's level design and technical needs, such as collision, which is why the world tile collider is the only custom mesh collider as it is precise to the model shape, allowing the navmesh to generate flawlessly. The other big factor with AI navigation was jumping and looking no dumb when doing so, which took several versions to get right.

Conclusion

The implementation of tying player progression with procedurally generated content to the extend proposed in this paper has been successfully achieved and shows feasible application a wide spectrum of game design mechanics. The PCG works to produce naturally engaging gameplay while navigating or jumping, and it was evident since the prototype. The final deliverable preserves this while additional production work has been done to enhance its framework. The only area of this project that had repeated issues was in designing the AI to

navigate and jump, not look dumb doing it, while also smart enough to know how to react to the player. Several iterations on the AI design were done until behaviors and basic functions worked properly together and could be altered simply.

Exploring this connection between player progression influencing level content isn't commonly explored, especially in a way where the level changes as a result. The system created to generate the game world gives ideas for other approaches to its application - such as through enemies or items instead of the environment - its open enough to try any kind of game mechanics. The implementation of procedural content being affected by player progression occurs for six stages (although the system can scale up as needed) and there are multiple checkpoints to choose from considering these changes occur for a portion of the overall game world.

The uniqueness of *Hexgen Roguevania* was approached by maintaining the design goals of player progression, platforming, simple combat, and exploration with a PCG system that can allow adjustability through development. This benefits greatly from Object-Oriented Programming methods to easily activate game and system features, refine, and then ultimately integrate, or else deactivate. Creating this game to investigate how player progression can affect generated content serves as an example of how these two aspects can be looked at to engage the player throughout the game space with their progress throughout the game. The use of PCG in this project also serves as an example of alternate approaches with PCG concepts, that it can be used in conjunction with game fundamentals.

The abundance of material covering PCG methods and examples provided excellent reference material on deciding how to connect them smoothly with the design goals for gameplay – including movement, jumping, and combat. Generating content is not random, it's

governed by some rules, therefore pseudorandom, and each game mechanic has some influence over the limitations on how or what can generate.

To take this project further is one option, as most features are functionally complete, however there is a great opportunity to theorize and experiment with different directions involving other ludic elements. Prototyping the idea will show right away how gameplay will feel like. The direction for this project is to keep adding more ways for the player to impact their game world as they progress.

The goal of this project constructs a PCG system that creates levels automatically and will change in reaction to the player progressing further in the game. The player reaches checkpoints which changes parts of their game world to harder zones, where the goal is to reveal and explore the final zone. This project has many directions because it is considering a stronger connection between player progress and how it is shown to the player; the outcome of this project is one of many routes explored, where mechanics of Metroidvania-style games are also investigated.

Bibliography

- Aron, J. (2019, September 21). Who wants a predictable life? Rad by Double Fine PC, Playstation 4, Xbox One and Nintendo Switch. *ICGA Journal*, 243(3248), 32-32. New Scientist. [https://doi.org/10.1016/S0262-4079\(19\)31773-7](https://doi.org/10.1016/S0262-4079(19)31773-7)
- Brown, J. A. (2018). *Pitching Diablo: On the development of marketing, procedural content, and narrative*. 40(4), 417-424. IOS Press. <https://doi.org/10.3233/ICG-180066>
- Bycer, J. (2019, Jan 16). *The conflicting design of the Zelda-Rogue*. Gamasutra. https://www.gamasutra.com/blogs/JoshBycer/20190118/334458/The_Conflicting_Design_of_the_ZeldaRogue.php
- Cornelisse, M. (2018, Dec 18). *Procedural texturing for hand-painted stylized character pipelines*. Gamasutra. https://www.gamasutra.com/blogs/MaritCornelisse/20191218/355911/Procedural_texturing_for_handpainted_stylized_character_pipelines.php
- Georgia Tech College of Computing. (2019, Nov 06). *Procedural Content Generation, continued*. https://www.cc.gatech.edu/~surban6/2019fa-gameAI/lectures/2019_11_06-ProceduralContentGeneration_PlayerModels-pt2.pdf
- Guzdial, M., & Riedl, M. (2018). Automated Game Design via Conceptual Expansion. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 14(1). Retrieved from <https://ojs.aaai.org/index.php/AIIDE/article/view/13022>
- Harris, J. (2021). *Exploring roguelike games*. CRC Press. <https://doi.org/10.1201/9781003053576>

Helin, T. (2016). *Defining environment art style in a procedural game world*. Kajaanin Ammattikorkeakoulu University of Applied Sciences.

https://www.theseus.fi/bitstream/handle/10024/114262/Helin_Tommi.pdf?sequence=1

Hess, B. (2018). *Plot and parcel: Procedural level design in 'XCom 2.'* GDC Vault.

<https://www.gdcvault.com/play/1025213/Plot-and-Parcel-Procedural-Level>

Hollstrand, P. (2020). *Supporting pre-production in game development: Process mapping and principles for a procedural prototyping tool* (Publication No. 1433725) [Master's Thesis, KTH]. Digitala Vetenskapliga Arkivet (DiVA). Retrieved from

<http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-273926>

Izgi, E. (2018). *Framework for roguelike video games development*. [Master's Thesis, Charles University]. CU Digital Repository. <http://hdl.handle.net/20.500.11956/94724>

Karth, I., (2019). Preliminary Poetics of Procedural Generation in Games. *Transactions of the Digital Games Research Association*, 4(3), 245-285.

<https://doi.org/10.26503/todigra.v4i3.106>

Kegel, D. B., & Haahr, M. (2020). Procedural Puzzle Generation: A Survey. *IEEE Transactions on Games*, 12(1), 21-40. <https://doi.org/10.1109/TG.2019.2917792>

Mawed, M. A., (2019). *Procedural content generation techniques and approaches*. (Publication No. 1921597) [Master's Thesis, NTNU]. Norwegian University of Science and Technology (NTNU). Retrieved from <http://hdl.handle.net/11250/2602851>

Nitsche, M., et. al. (n.d.) *Designing procedural game spaces: A case study*. Georgia Tech College of Computing.

<http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.72.589>

Parborg, S., & Holm, R. (2018). *Generative design of game graphics and levels*. (Publication No. 1228100) [Master's Thesis, Linköping University]. Digitala Vetenskapliga Arkivet (DiVa). Retrieved from <http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-149036>

Parberry, I. (2018). *Procedural Content Generation*. Ian Parberry.
<https://ianparberry.com/research/content/>

Patel, A. (2020, May). *Hexagonal grids*. Red Blob Games.
<https://www.redblobgames.com/grids/hexagons/>

Sapio, F. (2019). *Hands-on artificial intelligence with Unreal Engine: Everything you want to know about game AI using blueprints or C++*. (L. Pinto, Akhil N., Ralph R., & Safis E. Ed.). Packt Publishing.

Schanzer, E., Krishnamurthi, S., Fisler, K. D. (2018). Creativity, Customization, and Ownership: Game Design in Bootstrap: Algebra. *SIGCSE '18: Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, 161-166.
<http://cs.brown.edu/~sk/Publications/Papers/Published/skf-creativity-bootstrap/paper.pdf>

Silva, G. Q. M. da. (2019, November 22). *Jumping AI for Unreal Engine*. Página principal.
<http://hdl.handle.net/10400.8/4545>

Smith, A. J., & Bryson, J. J. (n.d.). *A logical approach to building dungeons: Answer set programming for hierarchal procedural content generation in roguelike games*. University of Bath, UK. <http://doc.gold.ac.uk/aisb50/AISB50-S02/AISB50-S2-Smith-paper.pdf>

Thompson, T. (2021, Jan 28). *Unexplored's secret: 'Cyclic dungeon generation.'* Gamasutra.

[https://www.gamasutra.com/blogs/TommyThompson/20210128/376783/Unexplored's](https://www.gamasutra.com/blogs/TommyThompson/20210128/376783/Unexplored's_Secret_Cyclic_Dungeon_Generation.php)

[Secret Cyclic Dungeon Generation.php](https://www.gamasutra.com/blogs/TommyThompson/20210128/376783/Unexplored's_Secret_Cyclic_Dungeon_Generation.php)

Vegricht, J. (2018). Evolutionary algorithm-based procedural level generator for a rogue-like game. [Bachelor's Thesis, Charles University]. CU Digital Repository.

<http://hdl.handle.net/20.500.11956/101249>

Zafar, A., Mujtaba, H., & Beg, O. (2021). Procedural content generation for general video game level generation. *Inteligencia Artificial*, 24(68), 33-36.

<https://doi.org/10.4114/intartif.vol24iss68pp33-36>

Zafar, A., Mujtaba, H., Biag, M. T., Beg, M. O. (2019). Using patterns as objectives for general video game level generation. *ICGA Journal* 41(2), 66-77. **[https://doi.org/10.3233/ICG-](https://doi.org/10.3233/ICG-190102)**

[190102](https://doi.org/10.3233/ICG-190102)

Zook, A., et al. (2019, Nov 06). *Skill-based mission generation: A data-driven temporal player modeling approach*. Georgia Tech College of Computing.

<https://www.cc.gatech.edu/~riedl/pubs/fdg-pcg12.pdf>