Challenges and Opportunities in Procedural Game Design

Maximilian Leschenar
Enschede, Netherlands
maxleschenar@hotmail.com

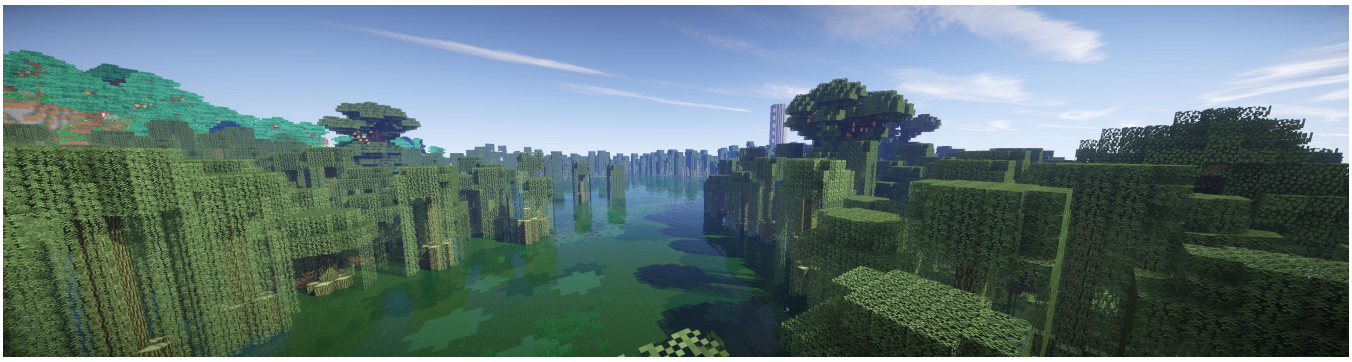

Figure 1: Example of Procedural World Generation in the game *Minecraft*[exotwist 2014]

ABSTRACT

The paper discusses the challenges and opportunities of Procedural generation in video games, specifically with a focus on procedural Game Design. It will discuss different methods of generating procedural video game levels while keeping the design philosophy of the game in mind. Key examples of different generation are analysed for their methods and how they try to create a well working experience, concluding with the result that the possibilities of procedural generation are immense and should be further experimented with.

KEYWORDS

game design, procedural generation, procedural design

ACM Reference Format:
Maximilian Leschenar. 2021. Challenges and Opportunities in Procedural Game Design. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Procedural generation describes the creation of assets based on an algorithm as opposed to it being created by a human. While different procedural generation methods have found different means of applications in varying fields one of its most common usage is in the creation of assets for video games.[Smith 2015], This connection mainly stems from the fact that procedural generation potentially has the capabilities to generate an infinite amount of environments and game states without human interference. Since video games are often intended to be replayable while offering a different experience each time for each player procedural generation has been a point of interest for game developer since the first video games.[Smith . 2011]

The first applications of procedural generation in early video games was in prominent examples such as *Rogue*(1980)", where it was used to create new dungeons out of ASCII signs every time the game was played[Nitsche . 2006]. Other games from that time frame such as *Elite*(1984), a space trading and combat simulator, used procedural generation to generate huge areas of play while still being limited to the small storage devices at the time.[Spufford 2003]

As technology and video games progressed so did procedural generation and the algorithms behind it. An example of a contemporary game based on procedural generation is *No Mans Sky*(2016). No Mans Sky uses more advanced algorithms to generate Billions of planets, all of which have theoretically unique flora and fauna. And while this theoretically offers an universe with unlimited replayability, players still mention that they get bored of the planets and quit. This boredom is caused by the lack of human control and guidance over the results and due to the limitation of the generator as all of the planets will still look and play very similar.[Heaven 2016]

This is not limited to No Mans Sky and can be found in a lot of games which use procedural generation to create potentially infinite asset. The term "Procedural Oatmeal" was created by the writer Kate Compton to describe this phenomenon as one "*can easily generate 10,000 bowls of plain oatmeal, with each oat being in a different position and different orientation, and mathematically speaking they will all be completely unique. But the user will likely just see a lot of oatmeal.* [Compton 2016]"

This problem has permeated big parts of the Game design regarding procedural generation and is still a major issue with games such as No Mans Sky. This paper will talk about the often discussed

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

opportunities offered by Procedural Generation in regards to Game Development and how different generation methods can alleviate problems like the "Procedural Oatmeal Issue".

2 PROCEDURAL CONTENT GENERATION IN LEVEL DESIGN

All of the examples of Procedural Content Generation(PCG) that are listed so far, mostly limit their procedural generation to the space in which the game takes place. Procedural generated maps and Levels have become a big part of the modern gaming landscape. Entire genres of games, such as roguelikes or roguelites are based on at least partially generating most of their game space[Harris 2009]. These games are based around the fact that the game space is a new one each time the game is played which encourages understanding and mastery of the elements that stay persistent in the game world while adapting to the new environment on each run. These elements and rules can be found on most other games using procedural generation for game space creation. Famous examples such as *Minecraft*(2019) are creating the map, with different biomes and landscapes, based on preset rules and conditions by both the player and the developers by either letting the player input a seed or creating the seed based on the internal system clock. The resulting world is then generated based on different repeated usages of Perlin noise, always sticking to the preset rules of realistic World generation which limit the values created by the Perlin noise generator.[Bergenstein 2011] Rules are one of the most important parts of procedural generation. The rules of these games are mostly based in the fact that the game spaces are procedurally generated. This also means that the demands aimed at the worlds, that are generated, are less demanding as the gameplay design already takes the generation into account. Other game genres, where the levels are usually handmade by a designer often struggle more when trying to automate that process using procedural generation. The entertainment and challenge found in game genres such as platformers is heavily dependent on the level layout and structure. Creating a functioning generator for a game like this demands a high understanding of the rules dictating the composition of a "good" level if one wants to create an experience similar to the hand crafted level.[Sorenson . 2011]

2.1 Examples of procedural level generation

Two examples on how to create procedural levels for games that are usually based on hand made levels can be found in the attempts to create a generator that is capable of generating complete levels of the classic platformer *Super Mario Bros*(1985) in the research papers by Steve Dahlskog and Julian Togelius and by Nathan Sorenson, Philippe Pasquier and Steve DiPaola. Both of the generators in these papers have the same final goal but go about this with different approaches.

2.1.1 Pattern-based Generation. The first paper, written by Steve Dahlskog and Julian Togelius, focuses on the importance of emulating existing design patterns taken from hand designed levels. Their approach is to analyse hand crafted levels for the existence of repeating, well working design patterns which they then categorize with associated problems, solutions, how to the pattern and specific comments, all of which would be later used for usage in the generator. In total they listed 23 different design patterns in different categories, separated into Enemies, Gaps, Valleys, Multiple Paths and Stairs.

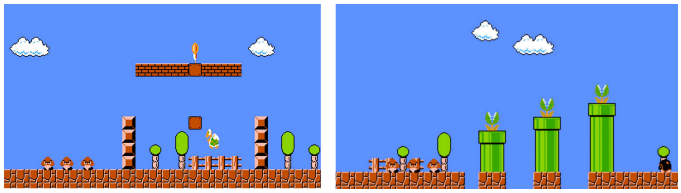

Figure 2: Examples of Design Patterns[Dahlskog Togelius 2018]

Their generator combined these 23 patterns in a semi-random series of patterns with only the start and end point of the game always staying the same. They also recognized that, since some patterns work better in conjunction with other, the generator should take this into account and place these next to each other more often. One issue with their current method was that due to the limited size of patterns, repetition soon became obvious. Their solution to this being, the option for the generator to modify some elements of the patterns based on adjacent patterns or player performance. An example of this would be to increase or decrease the size of a platform if adjacent patterns are deemed as too easy or too hard. [Dahlskog Togelius 2018]

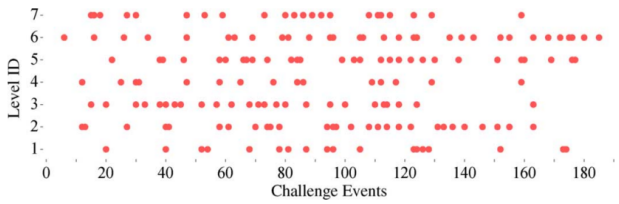

Figure 3: The timelines of seven Levels with the red dots depicting all challenge events. The grouping into different rhythm groups can be clearly seen.[Sorenson . 2011]

2.1.2 Generation based on Rhythm groups. The second paper, written by Nathan Sorenson, Philippe Pasquier, and Steve DiPaola focuses more on the design philosophy behind the entire level in comparison to isolated design patterns. Their approach was to analyse the entire level based on the concept of their *"Model of Fun"*. This model is based on the *"Framework for Analysis of 2D Platformer Levels"* by Gillian Smith, Mee Cha and Jim Whitehead [Smith . 2008] which proposes the importance of "rhythm groups" in 2D Platformer Level Design. Rhythm groups are small sections of levels which encapsulate enemies or other obstacles to the player which are then followed up by a section of relative peace where the player can recover. The difficulty of these groups can then be easily adjusted individually to create the ideal flow for that level. [Smith . 2008]

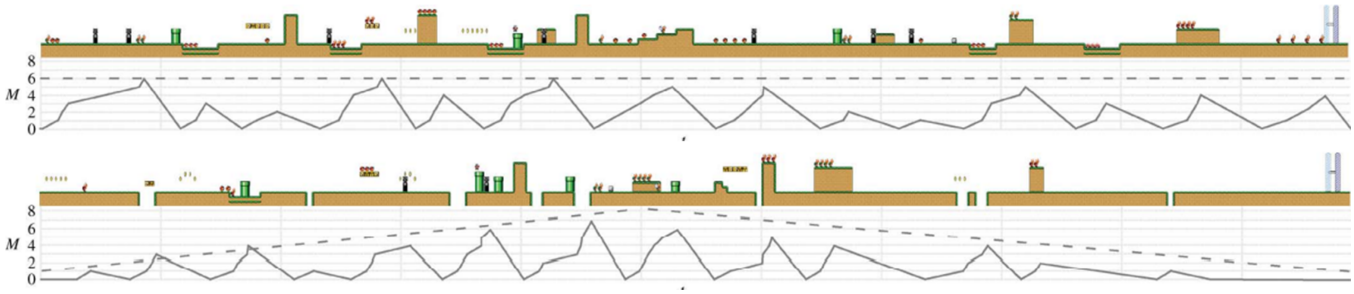

Figure 4: Two Levels generated by the Generator. The first one was generated based upon a static difficulty curve(dashed line) and the second one on a rising and falling one. Note the placement of objects based on rhythm groups.[Sorenson . 2011]

The "Model of Fun" takes this Framework as baseline upon which the entirety of a levels design is deconstructed. They used computational analysis to create this model by looking at 28 Levels from the original game for the placement of elements which introduce difficulty to the playthrough. The placement and distribution of elements such as enemies, gaps, jumps and was then traced upon timelines for each levels. These timelines then showcased the optimal placement for rhythm groups for the typical level setup and also an easy way to identify increasing difficulty by looking for increased density of obstacles.(Fig. 3)[Sorenson . 2011]

Their generator allows the designer to input graphs based on these timelines which depict the rhtyhm groups and their associated difficulty. The generator will then take these graphs and place obstacles in such a way that they line up with the pre-made rhythm groups. While this is technically already procedural generation their generator also allows designer to only input a difficulty curve upon which the generator will generate its own rhythm groups.(Fig 4.) [Sorenson . 2011]

Both of these generators in the end create levels that are of similar if not the same quality as hand designed levels and while both of them have the issue, that a player will notice repeating patterns after a certain amount of replays they imitate existing Level structure well and showcase the possibility of procedural Level design for more then just simple Level layouts. [Dahlskog Togelius 2018] [Sorenson . 2011]

3 PROCEDURAL GAME AND DIFFICULTY DESIGN

While the importance of static level design is great there are some aspects of game design that are rarely explored when talking about procedural generation[Agis . 2015] . One of them is procedural game design and difficulty adjustment during gameplay based on player performance. Changing the difficulty and some elements of how the game is played at runtime is big factor when it comes to player retention on games, especially if you combine it with procedural level generation.[Agis . 2015]

Rules and limitations are one off the most essential parts of game design and are therefore an important part when creating a generator for PCG that is supposed to create an experience on runtime which is still supposed to be enjoyable. [Adams Dormans 2012]. An example of the structure for a generator that is effectively able to generate new content on runtime based on player performance was proposed by Rilla Khaled, Mark J. Nelson and Pippin Barr. [Khaled . 2013].

3.1 Design metaphors for Procedural Game Design

Their structure is made up of 4 different metaphors, all of which work with each other and are essential for the generation of a well working PCG system based on player performance. The first metaphor is the "Tool". A general term to describe the software or program used to create something. This tool does not necessarily need to be used for PCG, but can also be used for other tasks as long as they involve modifying or creating. Their paper defines them as:"Devices or instruments manipulated for the purpose of achieving specific goals, changing the environment and acting as an extension of the user"[Khaled . 2013]. In the context of PCG the tool is used to modify the "material", the second metaphor. The materials are the assets which change based on the input the tool receives. Examples of Assets would be the structure of a level, the stats and visuals of a weapon or even the mechanics of a Game.[Khaled . 2013] The instructions which are received by the tool, in the context of PCG are given by the "designer". A term to describe all algorithms designed for solving game design problems. Their tasks can widely vary between how much PCG is needed in the existing context, with examples ranging from adapting the difficulty of an individual level to even implementing new plot points into the game if it deems them to be right. The limitations of the designer are based on what humans evaluate and the "Expert". The tasks of the expert are defined as "monitoring, analysing, interpreting, and assessing data resulting from gameplay" [Khaled . 2013]. Since this amount of data that can be collected by analysing player behaviour, the metaphor of the expert is split into the player and domain Expert. The player expert is especially focused on player experience. The output it gives is used to specialize the experience based on the current player. For this purpose it analyses player performance and gives the orders to the designer to design the level accordingly. [Khaled . 2013] The domain expert is a concept more often found in serious games. The task of the domain expert is to keep track of the players skill in specific areas upon which it tell the designer to create levels or puzzles specifically tailored to that skill level which differentiates it from the player expert. One example of the usage off a domain expert can be found in *Refractions*([Andersen 2012]. The goal of Refractions is to teach the player about fractions

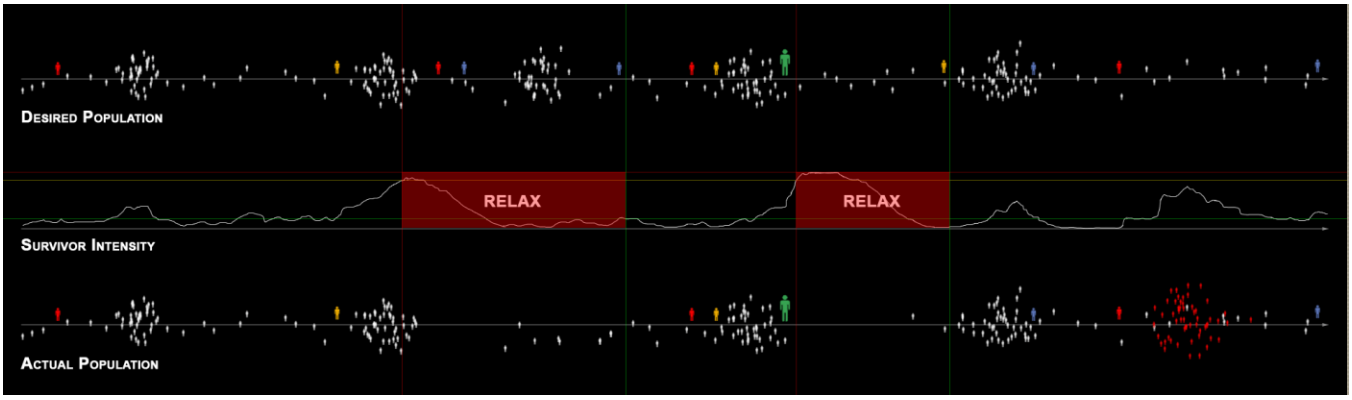

Figure 5: A graph showcasing the pregenerated zombie population at the start of the level, the level of survivor intensity throughout the gameplay and the actual zombie population during gameplay based on the survivor intensity. White figures symbolise normal zombies while coloured ones symbolise special zombies[Booth 2009]

and it does so by the usage of a domain expert which tracks the players progress in levels. The expert will then change values such as Thoroughness, forgiveness and Aggressiveness upon which the designer will construct the next levels.[Khaled . 2013]

3.2 Procedural Game Design in Left4Dead

Great examples of using PCG during runtime were created by *Valve Corporation*, a video game developer based in the United States of America. One example are the games *Left 4 Dead*(L4D)(2008) and *Left 4 Dead 2*(2009), in which the player(s) have to fight hordes of zombies to reach a safe room at the end of a level. During development the developers of L4D noticed that players are good at memorizing static locations of enemies, loot and level triggers, which removes suspense and results in players learning an "optimal strategy"that works on very playthrough. Even multiple sets of manually placed scripts and triggers don't work as player will also learn their location. To prevent this both games make heavy usage of the so called "AI Director". An algorithm which combines both the designer and the Player expert into one. The AI Directors main task is to promote replayability by creating "Structured Unpredictability"[Booth 2009]. It does so by first procedurally creating a flow through the linear level by deciding the amount of differently sized zombie hordes and special infected a specific area will have. During this process it considers all possible paths throughout out the level, placing more enemies on the "ideal" shorter paths to increase difficulty.The director will also keep some areas clear of zombies so that the player has time to relax. While the game is ongoing the Director will remove enemies that the players have moved past and will place new ones outside of the view of the players. The placement of the Enemies is based on different rules to empathize the difference between them and increase the difficulty. Zombie hordes are often spawned behind the players to catch them by surprise while enemies which are more programmed towards creating traps for the players are spawned in front of them. Weapons and loot are likewise placed procedurally throughout the level. [Booth 2009]

The previously generated zombie hordes and special infected can also be overwritten during gameplay by the director. The director is using a value called "Survivor intensity" which build up over time based on certain actions such as: the player taking damage, a player being incapacitated or a player dying. The Survivor intensity follows the same routine numerous times throughout the level starting with a build up where the intensity rises until a certain peak during which the maximum amount of enemies are spawned. this is sustained for three to five second upon which the intensity will rapidly fall during which the current enemy encounter will fade out. Afterwards comes a period of relaxation for 30-45 seconds until the build up starts again.(Fig. 5) [Booth 2009] Note the resemblance to the rhythm groups mentioned earlier.

This creates an experience that is individually tailored to the current player while being generated new every run. The functionality connected to the Survivor Intensity allows for a playthrough that is always adjusted to the player performance, easing of the player is he is under pressure for too long while still maintaining a constant system of difficulty.

4 CONCLUSION

Procedural Level Design has already proven itself to be a powerful tool that has the power to generate expansive worlds with comparatively little effort. These environments can, with a well adjusted generator, be just as good from a gameplay perspective then ones that were created by hand. Procedural generation also has shown itself to be a valuable tool for designers, when trying to create experiences that are easily replayable. The possibilities for procedural game design are therefore only limited to the imagination of designers and will most likely see more widespread adaptation in the future.

REFERENCES

ErnestAdams2012Adams, E. Dormans, J. 2012. Game mechanics: advanced game design Game mechanics: advanced game design. New Riders.

Agis2015Agis, RA., Cohen, A. Mart´ınez, DC. 2015. Argumentative AI Director Using Defeasible Logic Programming Argumentative ai director using defeasible logic programming. Computer Games Computer games (96–111). Springer.

Andersen2012Andersen, E. 2012. Optimizing adaptivity in educational games Optimizing adaptivity in educational games. Proceedings of the International Conference on the Foundations of Digital Games Proceedings of the international conference on the foundations of digital games (279–281).

Bergenstein2011Bergenstein, J. 2011. A Short Demystification of the "Map Seed" A short demystification of the "map seed". Mojang.

MichaelBooth2009Booth, M. 2009. The AI Systems of Left 4 Dead. The ai systems of left 4 dead. Presentation.

compton2016Compton, K. 2016. So you want to build a generator… So you want to build a generator….

SteveDahlskog2018Dahlskog, S. Togelius, J. 2018. Patterns and procedural content generation: revisiting Mario in world 1 level 1 Patterns and procedural content generation: revisiting mario in world 1 level 1. Proceedings of the First Workshop on Design Patterns in Games Proceedings of the first workshop on design patterns in games (1–8).

exotwist2014exotwist. 2014. It's no bridge, but I took this ultra-widescreen picture as a wallpaper for my new computer (more in comments). It's no bridge, but i took this ultra-widescreen picture as a wallpaper for my new computer (more in comments). Reddit. https://www.reddit.com/r/Minecraft/comments/2o7fjr/its_no_bridge_but_i_took_this_ultrawidescreen/ [Online; accessed August 23, 2021]

Harris2009Harris, J. 2009. The Berlin Interpretation The berlin interpretation. game-setwatch.

Heaven2016Heaven, D. 2016. When infinity gets boring: What went wrong with No Man's SkyR When infinity gets boring: What went wrong with no man's skyr. NewScientist.

Khaled2013Khaled, R., Nelson, MJ. Barr, P. 2013. Design metaphors for procedural content generation in games Design metaphors for procedural content generation in games. Proceedings of the SIGCHI conference on human factors in computing systems Proceedings of the sigchi conference on human factors in computing systems (1509–1518).

nitsche2006designingNitsche, M., Ashmore, C., Hankinson, W., Fitzpatrick, R., Kelly, J. Margenau, K. 2006. Designing Procedural Game Spaces: A Case Study Designing procedural game spaces: A case study. Proceedings of FuturePlay2006.

Gillian2015Smith, G. 2015. An Analog History of Procedural Content Generation. An analog history of procedural content generation. FDG. Fdg.

Smith2008Smith, G., Cha, M. Whitehead.Jim. 2008. A framework for analysis of 2D platformer levels A framework for analysis of 2d platformer levels. Proceedings of the 2008 ACM SIGGRAPH symposium on Video games. Proceedings of the 2008 acm siggraph symposium on video games.

Smith2011Smith, G., Gan, E., Othenin-Girard, A. Whitehead, J. 2011. PCG-based game design: enabling new play experiences through procedural content generation Pcg-based game design: enabling new play experiences through procedural content generation. Proceedings of the 2nd International Workshop on Procedural Content Generation in Games Proceedings of the 2nd international workshop on procedural content generation in games (1–4).

Sorenson2011Sorenson, N., Pasquier, P. DiPaola, S. 2011. A generic approach to challenge modeling for the procedural creation of video game levels A generic approach to challenge modeling for the procedural creation of video game levels. IEEE Transactions on Computational Intelligence and AI in Games33229–244.

Spufford2003Spufford, F. 2003. Masters of their universe Masters of their universe. The Guardian.