# A Comparison of which Procedural Content Generator Provides More Variety and Reliability for an Indie Development Team

**COMP110 - Computer Architecture Essay**

1507516

December 6, 2015

**WIP - recommendation and conclusion not finished**

This paper will evaluate and compare three articles; A Rhythm-Based Level Generator for 2-D platformers [1], Procedural Content Generation Using Occupancy-Regulated Extension [2] and Procedural Content Generation Using Patterns as Objectives [3]. These articles all have their own techniques to procedurally generate content for a 2-D platformer. This article recommends that Launchpad: A Rhythm-Based Level Generator is the most appropriate Procedural Content Generator (PCG) for an indie developer that would like to implement a 2-D platformer, based on variety and extensibility.

## 1 Introduction

The way in which we use Procedural Content Generators (PCG) to develop content for games is a vital part of the video game design process. With the recent release of popular procedurally generated 2-D games, such as *Terraria*, *Starbound* and *Spelunky* [4], it is important to choose a procedural content generator that suits your needs, as each is designed for a particular situation. For the context of this paper, the indie developers would like to implement a PCG onto the PC platform that can be integrated into the engine, then compile the level at runtime.

The work is motivated by the complexity and variety of all the different approaches to Procedural Content Generation, and this paper aims to provide a comparison of three different papers and make a recommendation based on the extensibility and variety of each PCG.

The reason why these three PCG articles were chosen was because they are all designed to generate levels in the style of Super Mario Bros (SMB). The three articles are:

Launchpad: A Rhythm-Based Level Generator for 2-D platformers [1]

Procedural Level Generation Using Occupancy-Regulated Extension [2]

Procedural Content Generation Using Patterns as Objectives [3]

The metrics being used to recommend the most appropriate PCG are variety and reliability. Variety can be measured by *Linearity* and *leniency* [1] which are metrics proposed by Smith and Whitehead [5]. Extensibility is being measured by how much control the developer has over the PCG and how the addition of content to the game will affect the performance of the PCG.

[1]leniency is how forgiving the level is to the player, e.g. levels with less obstacles or enemy's are more lenient. Linearity is a measure of the varying geometry of a level

# 2 Review of Procedural Content Generation Articles

## 2.1 Launchpad: A Rhythm-Based Level Generator

Launchpad generates levels out of small segments called "*rhythm groups*" which are short, non-overlapping areas that contain a challenge for the player. I.e. a series of quick jumps or jumping back and forth between platforms. The rhythm groups are generated using a 2 tiered grammar [2] based approach.

The first stage of this creates a set of beats that correspond to a player action. This is good because it means the game can be improved by the addition of varying player actions. Furthermore each beat can be The second stage takes the player action and creates geometry based on its parameters, this is then added to the level as a "rhythm group".

To achieve guaranteed playability of levels, Launchpad takes into account a set of parameters that determine how the player moves, e.g. The maximum speed the player can move, how high they can jump etc [7]. This means that each level that is generated does not create sections that the player cannot pass. Furthermore designers can have a lot more control over the design of player movement without having to be concerned with level geometry, thus also increasing the extensibility of the PCG.

However, this approach is designed for a "speedrun" type of gameplay, where player dexterity is key [1]. This means that it does not currently work well with different types of gameplay, i.e. exploration.

[2]Grammar in this context meaning a set of rules for rewriting strings, which forms the basics of how procedural content generation works [6].

## 2.2 Procedural Level Generation Using Occupancy-Regulated Extension

This procedural level generator is able to generate levels without knowing anything about how the game works. This method works by using a library of chunks and assembling them based on a complex set of parameters.

This algorithm generates the content in 3 stages, firstly it selects the location at which to start generating geometry. It then selects chunks from the library that is then tested with its filtering and selection algorithms to see if it is appropriate to place at that location. Lastly, that chunk is integrated within the existing content.

Compared to Launchpad, this method requires a large amount of human authored content to be available in order to get a varied level. Furthermore, this method is not very extensible as the more items are added to the library the longer it takes to generate a level. Even after the level is generated, it still requires a large amount of post-processing to make the game playable.

However when done correctly with the right settings, this approach can generate some very creative and complex levels as seen in figure 1 .

This approach does not guarantee playable levels, which is a very undesirable property when creating a level.

## 2.3 Procedural Content Generation Using Patterns as Objectives

Dahlskog and Togelius take an interesting approach to PCG, by looking for patterns in the way levels are structured in an original game such as SMB, then analyzing that level design to generate levels that use the same level design techniques. This means that the levels can maintain a high quality, but still be varied at the same time.

This method works by identifying "micro-patterns" and "meso-patterns" in an existing game, such as SMB. Micro-patterns are a vertical 1-block wide slice of the level, and meso-patterns are the way in which micro-patterns are arranged. The generator then uses the micro-patterns as building blocks to generate a new level.

However this PCG is not ideal for an indie developer as the implementation requires a version of an existing game to work, this means that the developer must create a game and a well designed level in order to generate more levels of the same type. Furthermore it does not allow for much linearity if it is being based of a level without a varied level.

## 3 Recommendation

Figure 1 shows a set of images displaying the different outputs of each PCG.

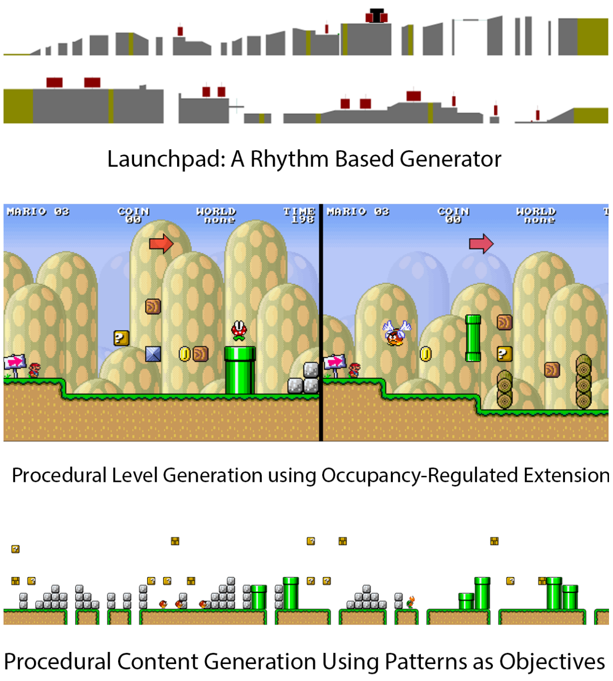
Launchpad: A Rhythm Based Generator
Procedural Level Generation using Occupancy-Regulated Extension
Procedural Content Generation Using Patterns as Objectives

Figure 1: PCG level comparison

## 4 Conclusion

Write your conclusion here. The conclusion should do more than summarise the essay, making clear the contribution of the work and highlighting key points, limitations, and outstanding questions. It should not introduce any new content or information.

## References

[1] G. Smith, M. Treanor, J. Whitehead, and M. Mateas, "Rhythm-based level generation for 2d platformers," in *Proceedings of the 4th International Conference on Foundations of Digital Games*. ACM, 2009, pp. 175–182.

[2] P. Mawhorter and M. Mateas, "Procedural level generation using occupancy-regulated extension," in *Computational Intelligence and Games (CIG), 2010 IEEE Symposium on*. IEEE, 2010, pp. 351–358.

[3] S. Dahlskog and J. Togelius, "Procedural content generation using patterns as objectives," in *Applications of Evolutionary Computation*. Springer, 2014, pp. 325–336.

[4] Steam. (2015, Dec. 05) Browsing procedural generation. Most popular Steam PCG games. [Online]. Available: http://store.steampowered.com/tag/en/Procedural%20Generation/#p=0&tab=TopSellers[Accessed:Dec.05,2015]

[5] G. Smith and J. Whitehead, "Analyzing the expressive range of a level generator," in *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*. ACM, 2010, p. 4.

[6] N. Shaker, J. Togelius, and M. J. Nelson, *Procedural Content Generation*

*in Games: A Textbook and an Overview of Current Research.* Springer, 2015.

[7] G. Smith, A. Othenin-Girard, J. Whitehead, and N. Wardrip-Fruin, “Pcg-based game design: creating endless web,” in *Proceedings of the International Conference on the Foundations of Digital Games.* ACM, 2012, pp. 188–195.