A Comparison of Procedural Content Generators

COMP110 - Computer Architecture Essay

AR185160

December 1, 2015

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 aims to provide a recommendation for the most appropriate technique if a Indie developer is wanting to make a procedurally generated 2-D platformer.

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 games, such as Minecraft, Elite: Dangerous and The Binding of Isaac, it is important to choose a procedural content generator that suits your needs, as there are a lot of different implementations of PCG to choose from, each designed for a particular

enviroment. The work is motivated by the complexity and variety of all the different approaches to Procedural Content Generation, and aims to provide a comparison of three different papers to make a recommendation based on qualities that would suit the needs for an Indie developer that would like to implement a PCG into their 2-D platformer game.

There is always going to be advantages and disadvantages of one PCG over another depending on the context. However for the context of a 2-D platformer, these are three PCG articles that are designed to generate levels in the style of Super Mario Bros(SMB).

The First Paper is:

Launchpad: A Rhythm-Based Level Generator for 2-D platformers[1], which is designed modularly into groups called "rhythm groups" which contain a set of challenges for the player, which are then pieced together to form a complete level. The objective of this PCG is to ensure every level it generates is playable and each level requires very little hand-authoring, all the designer has to do is modify the parameters in the PCG settings.

The next paper is:

Procedural Level Generation Using Occupancy-Regulated Extension [2], which is a more general purpose algorithm that can be applied to other types of games. However this aproach prefers variety over playability, so not all the levels it generates will be playable, but can be very complex.

The last paper is:

Procedural Content Generation Using Patterns as Objectives[3] They take an interesting approach to PCG, by looking at the way levels are structed in an original game such as SMB, then analysing that level design to generate levels that use the same design techniques.

2 Review of the Articles

There are many different approaches to implementing PCG in games, the reason I chose these three articles is because they were specified towards 2-D platformers. In particular, they were all loosely based of SMB, and generate content in that same style.

The recommendation at the end of this paper will be based off the desirable characteristics in each algorithm, such as *expressive range* proposed by Smith and Whitehead[4]. Expressive range contains a set of metrics such as *leniency* and *Linearity* which are appropriate for analyzing the difficulty and height variance of platform levels.

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. These challenges can be things such as a series of quick jumps or jumping back and forth between platforms, the rhythm groups are generated using a 2 tiered grammar [1] based approach.

2.2 Procedural Level Generation Using Occupancy-Regulated Extension

2.3 Procedural Content Generation Using Patterns as Objectives

3 Recommendation

Your essay must make a clear recommendation, in terms of which of the three techniques you have reviewed is the best according to whichever metric or

[1] Grammar in this context meaning a set of rules for rewriting strings, which forms the basics of how procedural content generation[5]

metrics you feel is most appropriate. You must justify your choice, backing it up with empirical evidence. However remember that an academic essay is not a murder mystery: you should already have briefly discussed your recommendation in the introduction and in other parts of the essay. Do not save it for a grand reveal at the end.

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*, pp. 175–182, ACM, 2009.

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

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

[4] 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*, p. 4, ACM, 2010.

[5] N. Shaker, J. Togelius, and M. J. Nelson, *Procedural Content Generation in Games: A Textbook and an Overview of Current Research.* Springer, 2015.