Implementation of PCG: Three Examples But Which Is The Easiest To Implement In A 2D Platformer?

COMP110 - Computer Architecture Essay

1506530

December 1, 2015

Procedural content generation (PCG) has been used in games for decades. It is an ever evolving method for creating immersive new content for the levels in games, that players can enjoy. This paper will describe three distinct algorithms for implementing PCG in a 2D platform game. Based on an analasis of relevant papers a recommendation will be made as to which algorithm would be the easiest to implement for a small indie company.

1 Introduction

Procedural content generation (PCG) is one of many methods used today to create and design levels, particularity 2D platformers. PCG not only creates a new experience for the player with each play through, but also when done correctly can keep development costs down, by removing the need for extra level designers [1]. The only real issue with PCG is which algorithm to choose,

with the variety of algorithms that are available it can be daunting to decide on which direction to go. So with PCG its important to have a clear idea of the task you want to accomplish as this will clearly define which algorithm would be the best for that game. With this in mind this paper describes three different algorithms that could be used in a 2D Platform game.

A brief description as possible for each algorithm and how they work are presented in the following sections. A comparison of ease of implementation into games will be drawn. Although looking at this, the recommended choice would be the ICA due to its simplicity and ease of use on more than one game. The three examples used are to be discussed on which algorithm would be the easiest to implement while still being the most efficient: An Iterative Content Adaptation Algorithm (ICA) used to enhance difficulty of a game [2]; Genetic Algorithm (GA) used alongside an Agent-based Digger Algorithm to use and add assets to an established game [1]; as well as an algorithm used alongside an input device to create a whole new game[3]. More detailed information can be found in academic literature for example [2, 1, 3].

2 Iterative Content Algorithm (ICA)

The first algorithm that is looked at is an ICA. It is considered interactive as it uses a graph overlay of a generated map to populate the map with objects and obstacles. Unlike other PCG levels where the level structure is produced from graphs, this algorithm requires each level to already be present to create something similar to the graph in Fig.1 Prince of Persia was used for these examples [2].

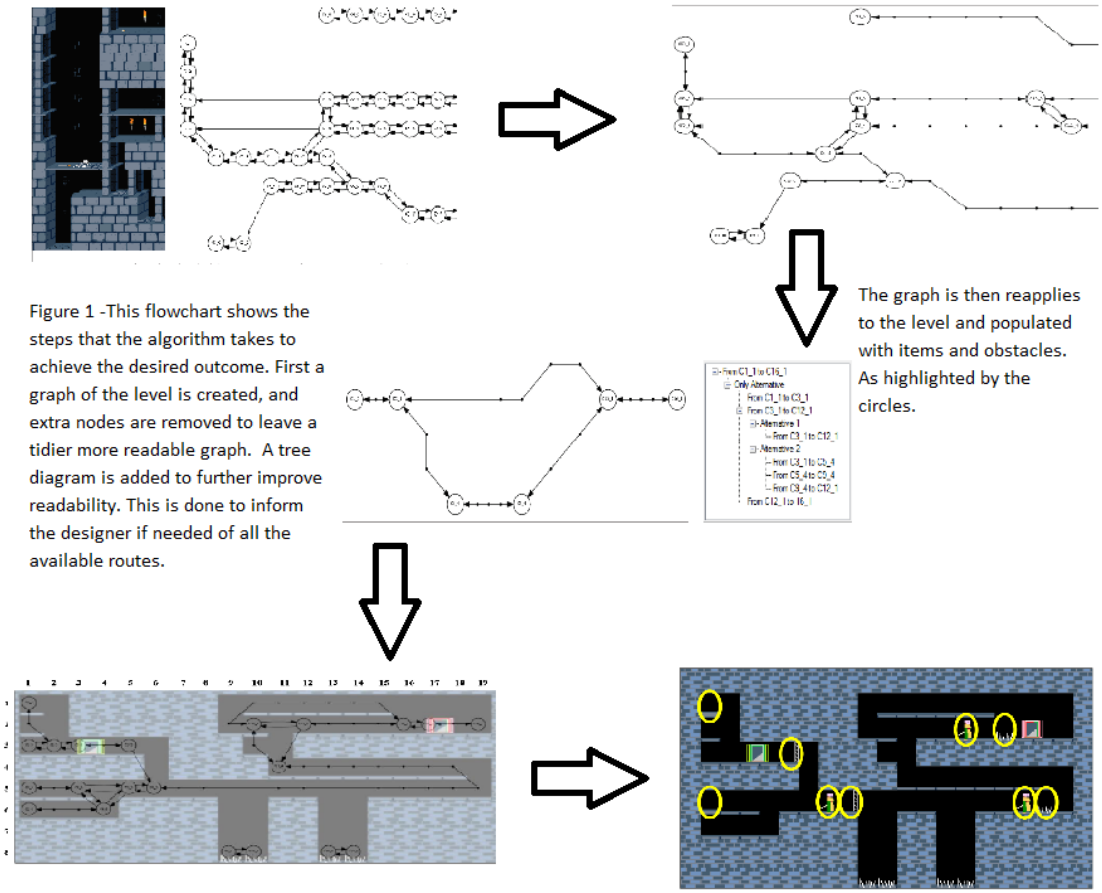

Figure 1: A summarised flowchart for the process of the ICA [2]

As can be seen from Fig.1 this particular method for PCG can be simple to implement. It does have limitations, one being that a level must already be created before implementation. Although it can also be put into multiple games with little to no modification see Fig.2,3. This makes it quite viable for a small indie company that can not afford the time to implement something with a greater complexity. Unlike the next method which takes it a step further by using a GA algorithm and created the entire level.

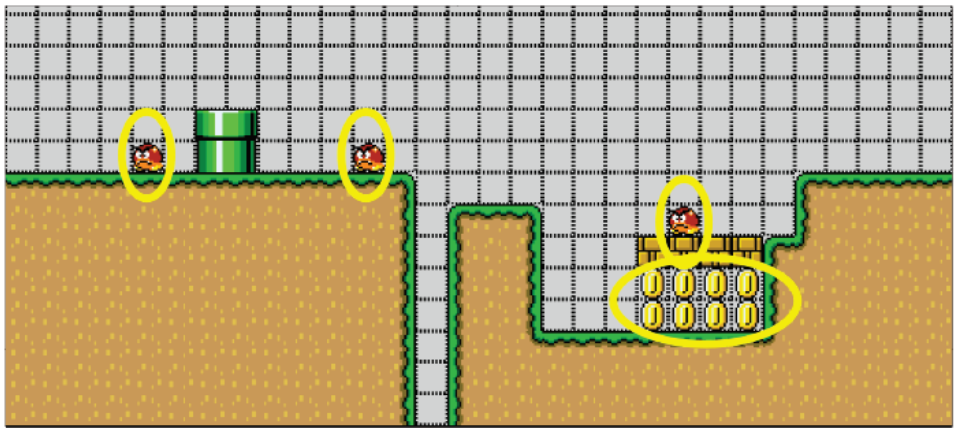

Figure 2: - A Sample of a level from*Infinite Mario Bros.* after the algorithm is applied [2]

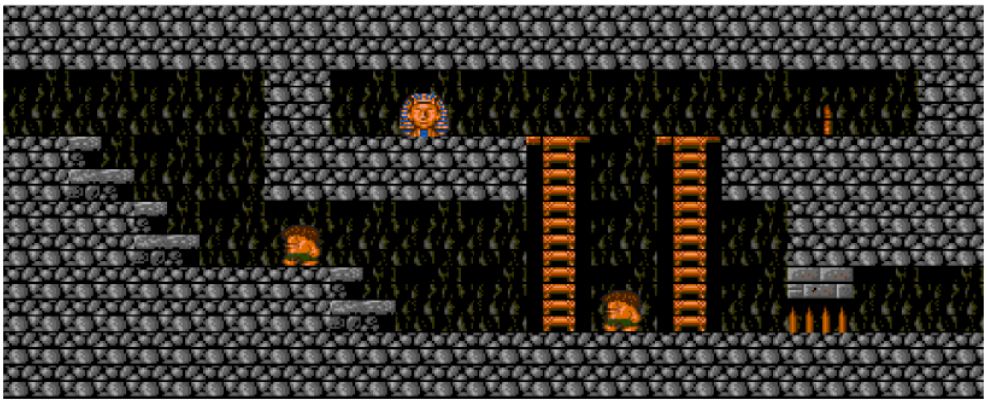

Figure 3: - A Sample of a level from*XRick* after the algorithm is applied [2]

2.1 Genetic Algorithm (GA)

Genetic Algorithm (GA) is a search heuristic that will mimic the natural selection process, a taxonomy of a evolutionary algorithm that runs through hundreds of evolutions to determine the most optimized method for the task at hand[4]. Infispel uses a Genetic Algorithm (GA) along with graphs of the rooms and the paths connecting them is used to populate the levels with items and objects Fig.4. [1].

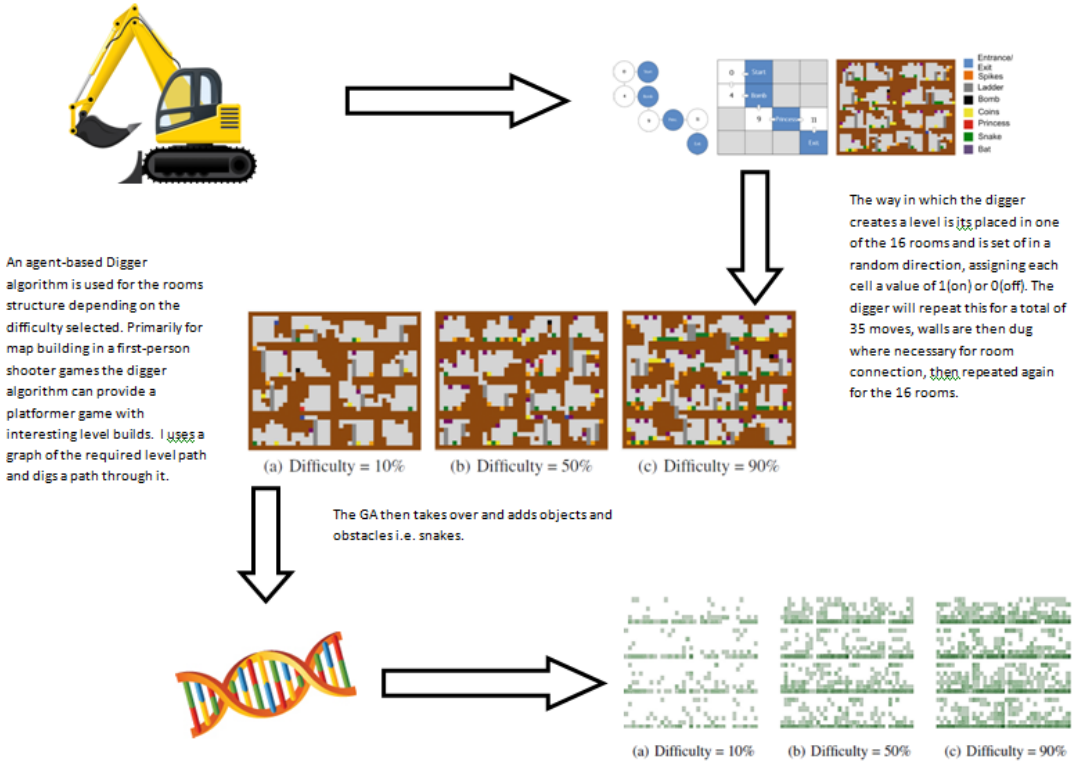

Figure 4: - A summarised flowchart for the process of the GA and Digger algorithm[1] .

Through this method a developer could create a PCG game based on these two Algorithms, but it would require modification before implementation unlike the ICA [1, 2], but like the ICA this algorithm once implemented will create an interesting level based on the difficulty chosen.

2.2 Endless Web (EW)

The last and most complex method would be using the Launchpad along with a PCG algorithm to create a game that continuously builds the level depending on your play style[3]. With EW, the approach taken was to use the Launchpads AI and as the controller followed by extensively modifying the mechanics to suit their needs.

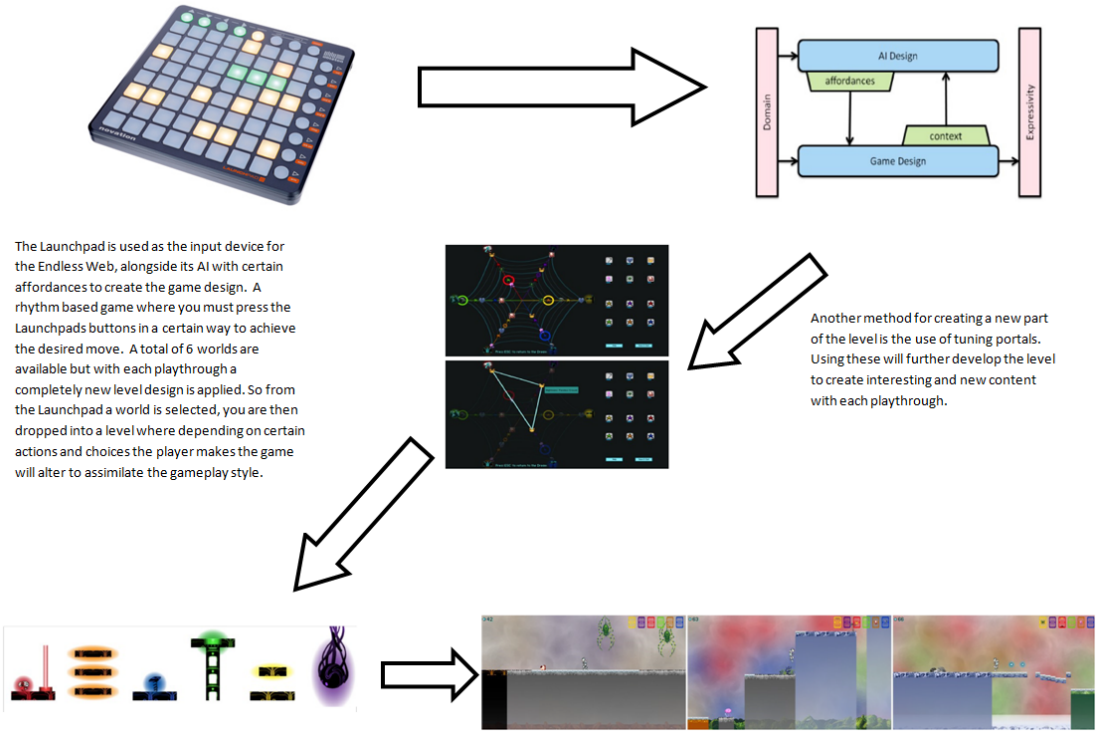

Figure 5: - A summarised flowchart for the process of Endless Web[3, 5]

.

As can be derived from Fig.5 Endless Web has a greater complexity that the first two algorithms. This being said a game of this type can produce a far more personal experience[3]. The development of such a PCG game would require a great deal of time and cost, for a small indie company it would be impractical to use such a method to create a game. For a company

with more time and experience, this method would be a wise approach.

3 Conclusion

As argued in section 2, the Iterative Content Algorithm (ICA) would be the most appropriate to use for a small 2D platform inide company. The ICA is the easiest most practical algorithm to implement into a PCG. It can support a wide variety of platform games with little to no modification [2]. While the other two methods would require greater modifications and commitment to achieve[1, 3]

Although further research could be done with the three methods presented here, it is clear that PCG is still evolving in many directions. From small item placement as seen with the ICA to churning out an entire game as you play, as is seen in The Endless Web. From this paper it is hoped that the reader can take away a basic understanding of these three ways which PCG can be implemented into a 2D platform game. Along with a good idea of which method would be useful when creating a game for themselves. This paper is based on ease of implementation. A different metric could influence the recommendation offered.

References

[1] W. Baghdadi, F. Shams Eddin, R. Al-Omari, Z. Alhalawani, M. Shaker, and N. Shaker, "A procedural method for automatic generation of spelunky levels," *18th European Conference, EvoApplications 2015*, pp. 305–317, 2015.

[2] F. Maurato, F. Birra, and M. Prspero dos Santos, "Enhancing level difficulty and additional content in platform videogames through graph

analysis," *ACE'12 Proceedings of the 9th international conference on Advances in Computer Entertainment*, pp. 70–84, 2012.

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

[4] M. El-Abd and M. Kamel, "A taxonomy of cooperative search algorithms," *Second International Workshop, HM 2005, Barcelona, Spain, August 29-30, 2005. Proceedings*, pp. 32–41, 2005.

[5] G. Smith, J. Whitehead, M. Meteas, M. Treanor, J. March, and M. Cha, "Launchpad: A rhythm-based level generator for 2-d platformers," *IEEE Transactions on Conmutational Intelligence and AI in Games 2011*, pp. 1–16, 2010.