What procedural generation algorithm Is best to use for puzzle generation?

COMP160 - Software Engineering Essay

1606695

April 2, 2017

With this essay I would like to examine the benefits of adding procedural generation to your games. What are good and bad practices to adding procedural generation. And what is the community reaction to procedurally generated content. Some people say that generated content can never surpass hand designed levels and characters.

1 Introduction

With this essay I would like to find out what is the best procedural generation algorithm that could be used to generate 2d puzzles. What algorithm you could use based on what you are looking for. By researching randomly generated puzzles I will find out what is being used by current developers and also what is the publics reaction to puzzles generated by computers and not made by humans through meticulous design. Is it even worth investing time and money into a system that will be hard to work with?

2 Chunk Based generation

Games like Spelunky and Infinite Mario use chunk-based generation to create their levels [1]. I feel like this is a great compromise between completely randomly generated levels and designing each level from scratch. By cutting up the level into chunks the game designer can design a bunch of x by x squares and the algorithm puts them together like puzzle pieces, as the algorithm is described by the developer of Spelunky [2]. Spelunky is a great example of balance between design and randomly generated levels with a score of 90 on metacritic [3] it is a beloved game by both critics and users. So if you are looking for something that still has a touch of human design a chunk based generation algorithm would be a great choice.

3 Rule Based Generation

An example of a Rule based generation would be logic based games like Sudoku [4]. There are simple rules to Sudoku that can be input into an algorithm and a completely unique puzzle would be created. But you cant just generate a board full of numbers and then delete numbers at random and hope it ends up solvable you also have to add logic rules of what amount of numbers are needed to solve the puzzle. This means that for a Rule based procedural generation algorithm a lot of additional coding is needed to create a design AI, something that will test if the puzzle is solvable [5]. I see this algorithm being used to add onto a game rather than create an entire game based on this something like a mini-game, because the more rules you add the more complex your code is going to get.

4 PROS of Procedural Puzzle generation

One of the most common reasons for adding procedural generation to games is replayability as said by Gillian Smith [6]. The argument is that procedural generation brings infinite possibilities thus a user would return to the game experiencing something completely new every time. That only works if the player does not see what is behind the curtain. Games like Spelunky and Binding of Isaac some people can't put down having an average of 100 hours of total playtime on steam[7]. But on the other end more recently No Man's Sky has been heavily criticized as having a quintillion soulless, lifeless worlds when players and critics would have been happy with a couple of hand designed, human build planets, as Mike Rose talks about avoiding these problems in this Gamasutra article[8].

5 CONS of Procedural Puzzle generation

The negatives of procedural generation are that you have limited control over the game design of your levels. There could be millions of possible puzzle variations in your game that doesn't mean that all of them are fun to solve, and that is not something that can be easily programmed into the rule set of your algorithm. Humans are complex beings and game development involves knowing how a certain person will approach a certain puzzle, this is called game theory as Raph Koster talks about it in his book[9]. You can't really teach game theory to an algorithm.

6 Conclusion

In conclusion using procedural generation algorithms are a very delicate balance between game design and randomly generated content. Finding

that sweet spot between well crafted levels and unique experiences is the main goal, as described in this article by Mark Hendrikx[10]. When using an algorithm you have to adapt it to your strengths if you are great at game design you can create simple rules and logic and make a game with millions of iterations with a chunk based algorithm. If you are a great programmer and would not like to focus on meticulous game design, you can add rule based puzzle generation as game play element to your game. To get a really great end result you will have to invest a lot of time into planning the rules for your algorithm and even then your result might not end up being fun to play. The greatest success in the industry for procedurally generated puzzles has been with chunk based algorithms. If a perfect rule based algorithm game existed no one would know it was randomly generated, because its inherent rule would be to create a game that feels like it was created by a human.

References

[1] 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.

[2] D. Kazemi, "Spelunky generator lessons." [Online]. Available: http://tinysubversions.com/spelunkyGen/

[3] Metacritic, "Spelunky metacritic." [Online]. Available: http://www.metacritic.com/game/pc/spelunky

[4] C. Browne, "Metrics for better puzzles," in *Game Analytics.* Springer, 2013, pp. 769–800.

[5] A. M. Smith, E. Andersen, M. Mateas, and Z. Popović, "A case study of expressively constrainable level design automation tools for

a puzzle game," in *Proceedings of the International Conference on the Foundations of Digital Games*. ACM, 2012, pp. 156–163.

[6] G. Smith, "Understanding procedural content generation: a design-centric analysis of the role of pcg in games," in *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*. ACM, 2014, pp. 917–926.

[7] "Steamspy binding of isaac." [Online]. Available: http://steamspy.com/app/250900

[8] M. Rose, "5 tips for using procedurally-generated content in your game." [Online]. Available: http://www.gamasutra.com/view/news/181853/5_tips_for_using_proceduralllygenerated_content_in_your_game.php

[9] R. Koster, *Theory of fun for game design*. " O'Reilly Media, Inc.", 2013.

[10] M. Hendrikx, S. Meijer, J. Van Der Velden, and A. Iosup, "Procedural content generation for games: A survey," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 9, no. 1, p. 1, 2013.