ISSN:0975-9646

Review Paper on Using Procedural Content Generation & Difficulty Curves

Paritosh Desai[*1], Ninad Kulkarni[*2], Suraj Jaiswal[*3], Sachin Chauthe[*4], Prof. Yogini Bazaz[#1]

[*1, *2, *3, *4] *B.E. Student, Department of Information Technology, Atharva College Of Engineering, Mumbai.*

[#1] *Assistant Professor, Department of Information Technology, Atharva College Of Engineering, Mumbai.*

***Abstract* ---Procedural Content Generation (PCG) is the branch of AI that deals with generating content algorithmically. It is used to reduce the cost of content creation while creating new types of content at a much greater speed with reduced effort. We aim to implement PCG by using Vasconcelos Genetic Algorithm (VGA) and the concept of Difficulty Curves. The obstacles patterns in the game will be generated procedurally at run time. Since the game focuses on endless content generation, the random or repetitive obstacle patterns would reduce the 'fun' factor of the game because user can get used to it and can also predict the content generated in such games. The game's content will be generated on the basis of a difficulty curve which will be adjusted depending on the progress of the user. The fitness function will compare the difficulty at any point of the generated content with the difficulty curve in order to create a game segment which is as close to the desired difficulty curve as possible.**

Keywords—procedural content generation; games; genetic algorithms; difficulty scaling; game design

I. INTRODUCTION

Gaming, as an industry is growing at an amazing rate and the expectations that players have from game content are rising with each released game title. To meet these demands game development studios incur rising costs in terms of payment to artists and programmers that supply that content along with the time required to develop such content. This gives rise to a unique application of AI algorithms, focusing more on the creative and artistic side of the game content rather than the strategic and tactical aspect of it; thus saving significant expense by producing desirable content algorithmically.

Procedural content generation (PCG) refers to creating game content automatically, through algorithmic means. In this paper, the term game content refers to all aspects of the game that affect gameplay other than non-player character (NPC) behaviour and the game engine itself [1].

Even established game companies can benefit from PCG, using it to generate 3D worlds, missions and other types of content. However, the first problem facing a game designer wanting to incorporate PCG techniques is the loss of control over the generated content. One of the main arguments against procedural content generation by the representatives of the gaming industry, at least when discussing online content generation, difficulty scaling and artificial intelligence adaptation, is the lack of reliability. Due to the manner in which most commercial games are designed, presenting content that is unplayable to the player is simply unacceptable.

II. RELATED WORK

Diaz-Furlong Hector Adrian and Solis-Gonzalez Cosio Ana Luisa [2] have defined a fitness function that doesn't depend on the game or the type of content. It calculates the difference between a difficulty curve set by the designer and the difficulty curve calculated from the candidate content. They have stressed on the importance of reviewing how the designer has set the difficulty pacing throughout the level.

Pieter Sponck & Co. [3] have listed High-Fitness Penalizing, Weight Clipping and Top Culling methods for the purpose of difficulty scaling.

Decker-Davis [4] states that procedurally generated content reduces the control that a designer has over the content but the dynamic adjustment of difficulty is one of the most important aspects of PCG. She also states that the level of challenge must be pegged at such a level that the game is found to be engrossing by the player. She mentions that the player must be put in a position where he has to consciously choose the path and its consequences.

S. Bakkes and J. van den Herik [5] have proposed a novel approach towards opponent A.I. which is based on information mined from previous games, called Case-Based A.I. They suggest building a cache of data which is gathered by observing simulations of the games. When a game is in progress, the A.I. strategy is determined by forming and matching a player model to the data cache. This method is able to dynamically adjust the difficulty according to the strategy employed by the player.

Nathan Sorenson [6] has presented a generative system which automatically creates video game levels. It allows high-level design features to be described in a top-down manner. This method uses a two-population genetic algorithm of feasible-infeasible candidates. The genotypes used by the author are Design Elements (DEs) i.e. modular building blocks that represent units of the level.

II.PROPOSED SYSTEM

A. Difficulty Curves & Procedural Content Generation

Though PCG overcomes challenges that occur in general game development, it has certain problems of its own; dynamic scaling of the difficulty level so as to match to the

skill of the player. Also, the content must be correct and playable. Testing content for correctness and playability adds to the overhead of PCG content generation.

By using difficulty curves, some of these issues can be addressed. The curves allow the difficulty of a level to be tuned with gradual increments or decrements as per the wish of the designer.

We aim to extend the system of difficulty curves used by Diaz-Furlong [2] so that the curve generation process is automatic and continuous. It will be based on certain parameters and randomization. Cubic spline interpolation is utilized to form a difficulty curve based on several control points. The number of position of the control points is based on the following:

- Minimum difficulty point.
- Progress of the player (Time for which the game has been active)
- Number of difficulty "peak" points.
- Number of difficulty "trough" points.

The numbers of peak and trough points themselves are picked randomly over a range depending on the progress made by the player. The probability of peak point occurrence will go up as the player moves further into the game.

III. CONTENT REPRESENTATION

A genetic algorithm will be used to generate a population of interesting obstacle patterns for the game. Consider one chunk to be of 5x3 cells. The first and last row will have "empty" cells. The middle three rows will contain the obstacles.

For simplicity, consider only three kinds of obstacles:

- Ground level obstacle – The player can jump over it.
- High-level obstacle – The player can move under the obstacle.
- Empty-level obstacle – The player can move simply by running through the obstacle.

Three types of obstacles and an empty cell gives us a chunk of 9 cells with 4 possibilities for each cell, thus a total of 9^4 = 6561 different possibilities. However, not all of them will be playable. Some of them might be playable but will hold no practical utility value. An algorithm must be used to identify the interesting patterns by taking into account factors such as the amount of keystrokes required by the player to navigate through the chunk. Using this, a "difficulty rating" will be assigned to each chunk. This collection of useful patterns will be stored for use in the game.

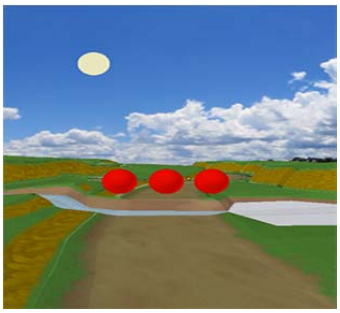

Fig 1. All obstacles at top

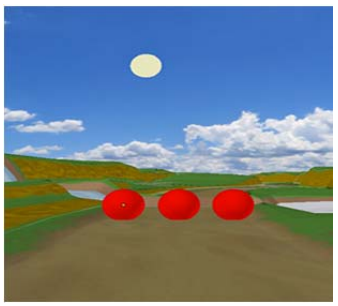

Fig 2. All obstacles at bottom

Fig 3. No obstacles

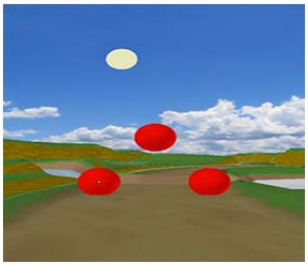

Fig 4. Combined top and bottom

IV. LEVEL GENERATION PROCESS

The difficulty curve is designed keeping the theory of flow in mind [7]. The curve should find a balance between the player's skill level and a sense of challenge that's needed to engage the player. The curve is defined by specifying certain points on it. The curve needs to be virtually limitless since there is no concept of a fixed level length in the game. The min and max levels of difficulty on the curve are gradually increased on the curve as the player makes progress in the game.

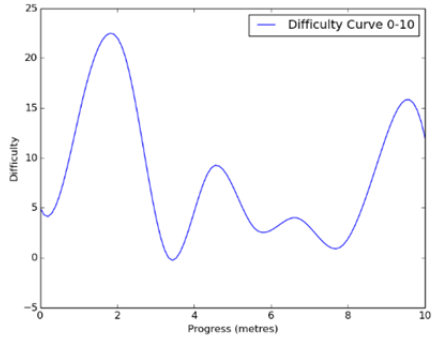

Fig 5. A sample curve generated for the distance 1-10

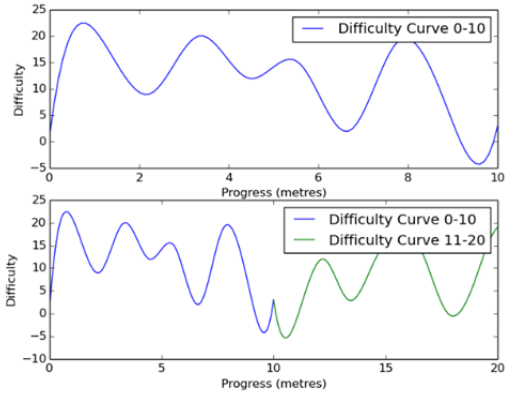

Fig 6. Comparison of two adjacent sample curves

1. *A. Vasconcelos Genetic Algorithm*

VGA (Vasconcelos Genetic Algorithm) [7] will be run while the game is being played, to map game content to the difficulty curve. The game designer defines the content's representation and designs the difficulty curve that he wants the content to have. The representation defines the length of the chromosome for each individual in the population. These are processed by the difficulty curve calculator providing the difficulty curve for each individual. This curve is compared to the curve created by the designer and calculates the fitness value of the chromosome. Finally, the chromosome with the best fitness is output from the generator.

The two points to be noted are that firstly, the difficulty curve is a non-constant in our game. The shape and characteristics of the curve will change to reflect the progress of the player. Secondly, the genetic algorithm will be creating the game level at a level of abstraction since it is essentially combining chunks which have been assigned a difficulty rating already. However, these chunks will be re-rated depending on the previously held rating of the chunk itself as well as the ratings of the chunks preceding and succeeding it.

B. Curve Generation Algorithm

Step 1:

a) N ← 30-50 (individuals in the population)
b) Pc ← 0.7-0.9 (probability of crossover)
c) Pm ← 0.005 (probability of mutation)
d) G ← 20-200 (number of generations)

Step 2: Calculate β (the number of bits to mutate) as:

β = ceiling [N × l ×Pm] where l = bits in the individuals genome (throughout we assume binary encoding of the solutions).

The above steps are part of the VGA with their key parameter values. Further experimentation is required to identify the optimum values. It is our intent to run simulations and determine the parameter values which would allow for the most efficient execution of the algorithm, particularly the number of generations. Since the level generation process is never-ending in a game of this nature, it is imperative that a feasible candidate must be generated in as few generations as possible.

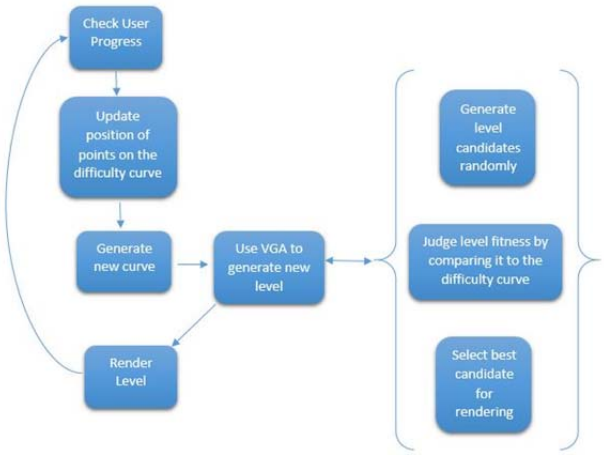

Fig 7. Block diagram-overview of PCG process

V. CONCLUSION

Thus, we believe that it is possible to create game levels that can reasonably compete with the quality of content that is created by a human game designer and in some cases go even beyond the ability of a human game designer. The content meets the requirements of the scenario, is perfectly playable and has a fun value as well.

VI. REFERENCES

[1] J. Togelius, G. N. Yannakakis, K. O. Stanley and C. Browne, "Search-based procedural generation: a taxonomy and survey" *IEEE Transactions on Computational Intelligence and AI in Games*, September 2011.

[2] Diaz-Furlong Hector Adrian, An Approach to Level Design Using Procedural Content Generation and Difficulty Curves, *Computational Intelligence and AI in Games*, IEEE Transactions.

[3] P. Spronck, I. Sprinkhuizen-Kuyper and E. Postma, "Dificulty scaling of game AI", Proc. *Fifth Int'l Conf. Intelligent Games and Simulation*, pp. 33-37, 2004.

[4] H.M. Decker-Davis, http://www.gamecareerguide.com/features/1120/tuning_difficulty_when_making_.php

[5] S. Bakkes, P. Spronck and J. van den Herik, "Rapid and Reliable Adaptation of Video Game AI", *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 1, no. 2, 2009

[6] Nathan Sorenson, Philippe Pasquier, "Towards a Generic Framework for Automated Video Game Level Creation" *Applications of Evolutionary ComputationLecture Notes in Computer Science*, Volume 6024, 2010.

[7] M. Csikszentmihalyi, Flow: The Psychology of Optimal Experience, Harper Collins, 1990.

[8] A. F. Kuri-Morales, "Solution of simultaneous non-linear equations using genetic algorithms", *WSEAS Transactions on Systems, pp. 44-51, WSEAS Press*, Issue 1, Vol 2