Chaotic Variations for Procedural Content Generation in Games

Alex Burnley
University of Colorado Boulder

I. Introduction

Demand for large amounts of content in games has only increased as the gaming industry has grown. This has lead to the development of a variety of procedural content generation (PCG) techniques, where new levels are created at run time for a player to enjoy. These techniques provide almost infinite amounts of content, however they can be complex to implement and achieve the desired results.

Chaotic variations, a method of generating variations on a symbol sequence using a chaotic mapping, has yet to be studied as a method of PCG for games. This paper applies this technique to create new levels for the game Super Mario Bros [1].

II. Related Work

Diana Dabby proposed a technique for generating musical variations of an original piece by mapping musical pitches to a chaotic trajectory then generating a second chaotic trajectory with a different initial condition and inverting the mapping [2]. Her results found that musical variations created through this method could retain characteristics of the original piece or become completely different. Bradley et al. extended this technique to movement sequences including dance and martial arts forms [3]. While the chaotic variations caused harsh transitions that had to be smoothed out, their results found that the generated sequences were natural in the context of the original sequence.

A substantial amount of research has been done on techniques for PCG. These extend from generating terrain to placing objects to fulfill feasibility and difficulty constraints.

Terrain generation is typically done through a noise based approach, where the values generated by noise can be mapped to tile types for 2D games or used to determine terrain height in 3D games. Gurler et al. defined a technique for constraining 3D Perlin noise based terrain in tile based games [4]. Another technique for tile based terrain was described by Macedo et al, where cellular automata and Hilbert curves are used to create levels that are aesthetically similar but have very different paths through the level.

Evolutionary algorithms, and other search algorithms, are commonly used in PCG to create levels that fit certain constraints. Brown et al. used an evolutionary algorithm to generate level layouts for the game Hotline Miami and used fitness functions that could be changed to adjust characteristics of the generated level [5]. Adrian et al. created a fitness function for measuring level difficulty, allowing evolutionary algorithms to be used to create levels with a target difficulty [6]. These approaches to PCG provide a lot of flexibility and can generate levels for almost any goals, although designing fitness functions can become very complex.

Recently, machine learning has also become a popular method for PCG. Nam et al. used reinforcement learning and stochastic noise parameters to build diverse multi-stage levels for a turn based role playing game [7]. Siper et al. trained a convolutional network to generate new levels for 2D games by treating level creation as a repair task [8]. Zhang et al. developed a deep learning model for repairing reachability issues in platforming games, ensuring that any level created is completable [9]. While effective, each of these works require creative, sometimes complex, methods to create a training dataset that emphasizes desired level characteristics.

III. Chaotic Variations

The technique used by Diana Dabby [2] and Bradley et al. [3] takes advantage of the sensitive dependence on initial conditions of chaotic attractors to create variations of a symbol sequence. New symbol sequences generated in this way can retain patterns and characteristics from the original, or be very different. In PCG, creating new content that has specific characteristics but is different enough to create a new experience for the player is the main goal, but it can be difficult to accomplish effectively. This technique provides a method for accomplishing this based on a single example level.

To apply this technique, a symbol sequence is mapped to a trajectory around a chaotic attractor. This is done by selecting a chaotic system and an initial condition, then using fourth order Runge Kutta to generate the points in the trajectory. Points are removed from the start of the trajectory to remove the transient, then each symbol is mapped to a point in the trajectory. In Diana Dabby's case, symbols were mapped to the x coordinate of the points, while Bradley et al. mapped the points to the three dimensional coordinates of the points. This paper uses the method used by Bradley et al. and maps the symbols to the three dimensional coordinates. To generate a variant of the symbol sequence, a new trajectory is generated from a different initial condition. The transient is removed, then the trajectory is converted into a symbol sequence: for each point, the nearest point in the original trajectory is found and the symbol mapped to it is used in the new sequence. This method is a deterministic way to create variants of a symbol sequence, and the sensitive dependence on initial conditions means a near infinite number of variants can be created.

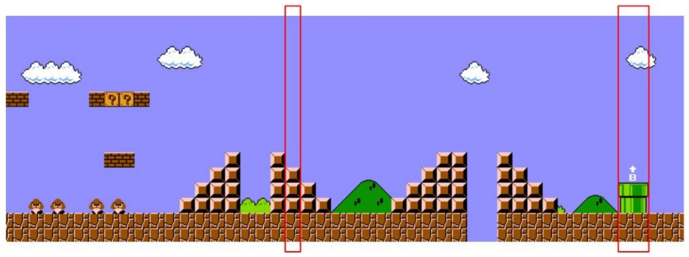


Fig. 1. When representing a level as a symbol sequence, symbols should represent the smallest section of level possible, but must fully contain any components of the level that cannot be broken apart. Here, a single column of tiles in the staircase can be represented by one symbol since the staircase can be broken apart, but the symbol representing the pipe must represent at least two columns to fully contain the pipe.

IV. Applying Chaotic Variations to Game Levels

Diana Dabby and Bradley et al. were very successful in applying this technique to music and movement sequences [2], [3]. These sequences are inherently one dimensional and easily represented as a symbol sequence. Levels in games are rarely one dimensional, usually being two or three dimensional. The main challenge of applying this technique is then to represent the game level as a symbol sequence. The game Super Mario Bros [1] was chosen to apply this technique to because levels are two dimensional, but the player's experience can be interpreted as one dimensional. As the player progresses through the level, they encounter obstacles one at a time, but the ordering of the obstacles defines the experience: a pipe is a single obstacle but can be used to easily jump over an enemy. A symbol can then be assigned to each section of level containing an obstacle, which is then used to represent the entire level as a symbol sequence.

What a single symbol represents must completely encapsulate any object that cannot be broken apart in the context of the game. For example, in Super Mario Bros, the level is constructed with 2D tiles, so the smallest section of the level that can be represented by a symbol is a column of tiles. Some obstacles, such as pipes, are wider than one column and cannot be broken apart, so a symbol must represent two columns to completely contain the pipe. Other obstacles, such as stairs, can be interpreted as a series of increasingly tall walls, so a symbol can represent a single column within the obstacle. This is shown in Figure 1.

Once the level was represented by a symbol sequence, each symbol was mapped to a point in a trajectory on the Lorenz attractor. A time step of .01 was used in fourth order Runge Kutta to generate the trajectory. Due to the relatively short length of the symbol sequence and small time step, the trajectory only wrapped around each side of the attractor approximately two times, seen in Figure 2. In practice, this results in new levels created using this technique all being very similar and containing the same few large sections of the original level. One way to increase the variety in level variants is to repeat the sequence. This helps to fill out the attractor, so that new trajectories that visit different parts of the attractor

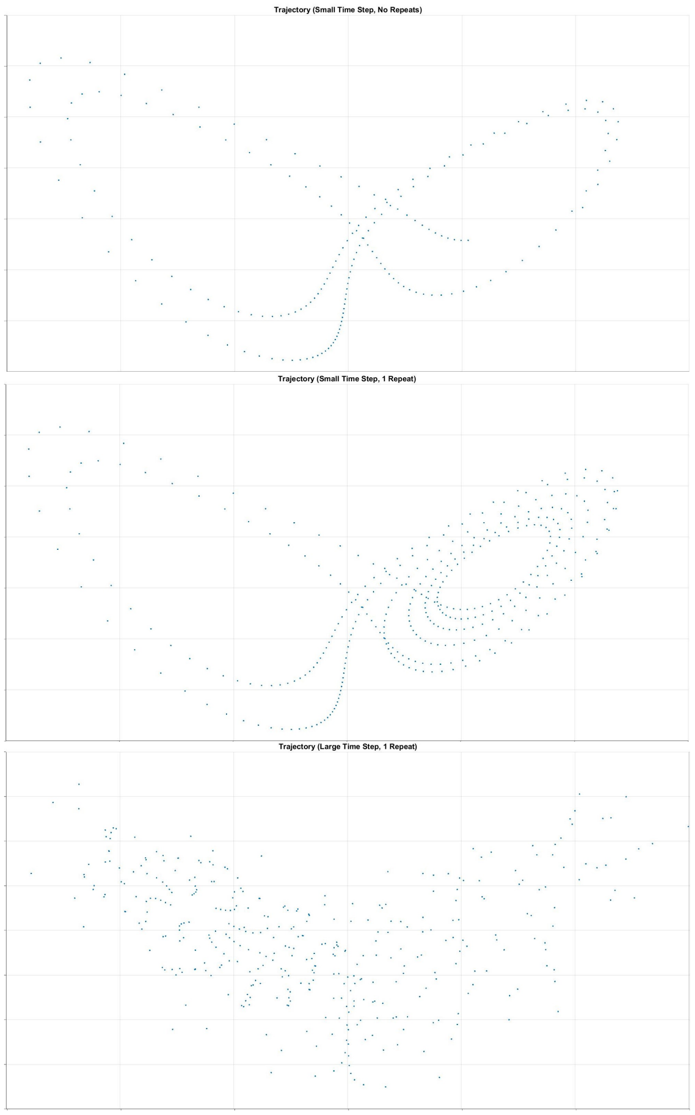


Fig. 2. These plots show the points that the symbol sequence representing a level was mapped to. From top to bottom, the plots show a trajectory with a small time step and no repeats, a trajectory with a short time step and 1 repeat, and a trajectory with a large time step and 1 repeat. When a symbol sequence is relatively short, the points the symbols are mapped to don't fill out much of the attractor. This can limit the possibilities when variants are generated. Repeating the sequence can help fill out the attractor and increase the variety in generated sequences. Increasing the time step used by Runge Kutta can also increase how much of the attractor is covered, although it also increases how much new symbol sequences will differ from the original.

may map to different parts of the level. However this still results in variants containing large parts of the original level, making them appear very similar to the original.

One of the most important parameters in this algorithm is the time step used in Runge Kutta when generating trajectories. This parameter allows a user to loosely control how similar variants are to the original level. When the time step is small, the symbol sequence is mapped to a short trajectory with points spread a across a short period of time. When the time step is large, the symbol sequence is mapped to a longer trajectory that captures chaotic behavior over a longer period

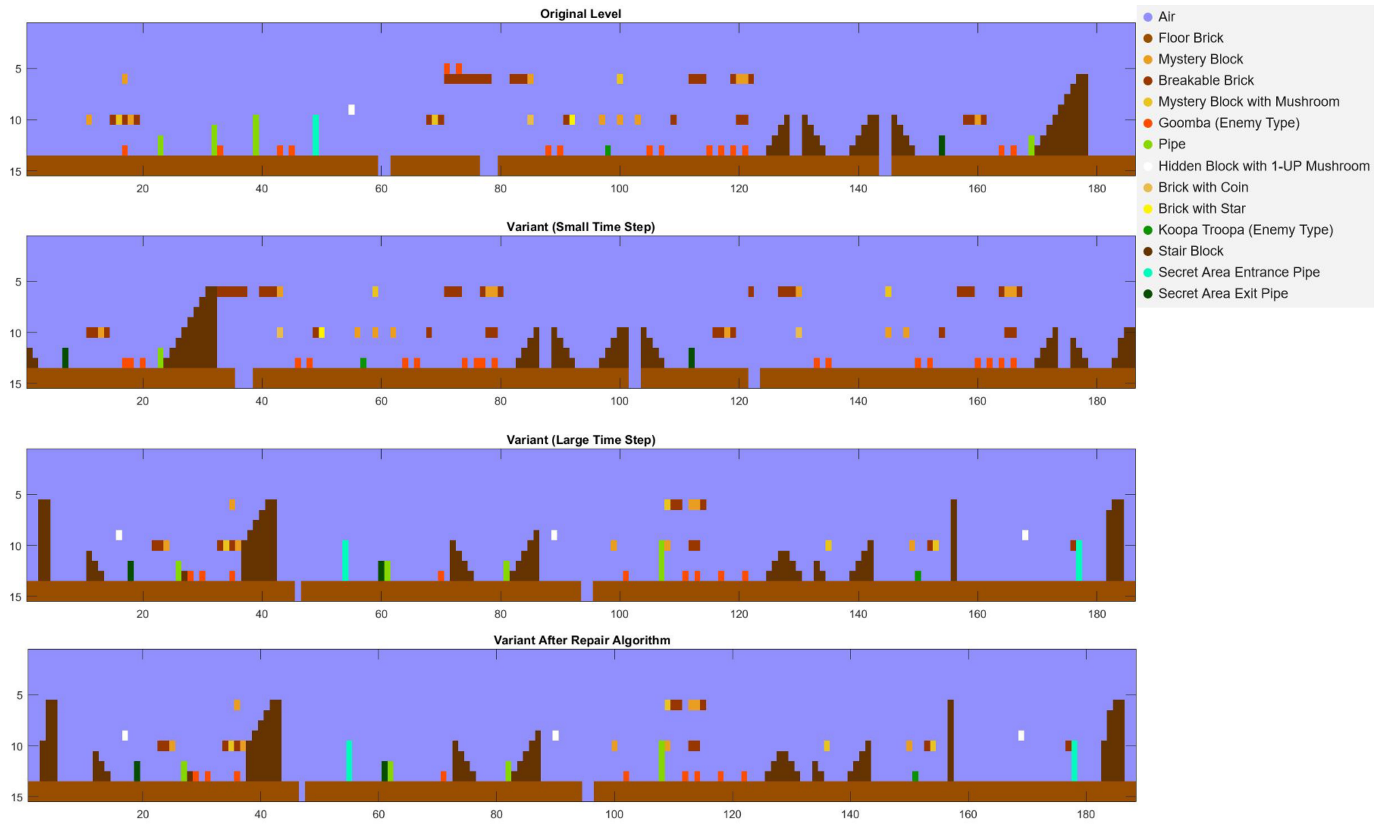


Fig. 3. Visual Comparison of Variants and Original Level. The top image is World 1-1 from Super Mario Bros. The following two images are variants created from the original, using a small and large time step respectively. The last image shows the results of running the repair algorithm on large time step variant to ensure it is possible to complete the level.

of time. In chaotic systems, perturbations grow over time. This means that when the second trajectory is generated, the first and second trajectories diverge in fewer steps when the time step is larger. In the context of reordering a symbol sequence, a small time step tends to result in larger parts of the original symbol sequence to be used in the new sequence, while a large time step tends to result in smaller parts of the original sequence being used in the new sequence.

V. Repairing Issues Caused by Chaotic Variations

Chaotic variations can place sections from completely different parts of a level right next to each other. This can cause issues where an obstacle is created that is not possible to overcome. The start of the level created with a large time step seen in Figure 3 is an example of this. At the start of the level, a wall was placed with no way of overcoming it.

This lead to the development of a simple algorithm for ”repairing” levels that can no longer be completed, described in Algorithm 1. The algorithm resolves issues by inserting a column from the original level, prioritizing columns located near the instigating column's location in the original level. This helps maintain the style and types of obstacles from the original level in the variant. An example of the results of this algorithm can be seen at the bottom of Figure 3.

The algorithm is specifically designed around the mechanics of Super Mario Bros, and takes advantage of them to simplify the algorithm. It only accounts for Mario's jumping capabilities, and does not check if there are tiles blocking Mario's movement. It also makes the assumption that only movement in the positive X direction allows progress, as this is a property of the original level.

VI. User Study

To determine whether players enjoyed levels created with this technique, a user study was designed to compare the experience of playing a level variant against the original level. Participants were chosen as players who either frequently played games, or had previously played Super Mario Bros and were familiar with its gameplay mechanics. Participants were asked to play three levels in random order: the original World 1-1 from Super Mario Bros, a variant created with a small time step, and a variant created with a large time step that had the repair algorithm run on it. These levels can be seen in Figure 3. After completing each level, the participant was asked to complete a survey about their experience.

The Game Experience Questionnaire (GEQ) [10] was used to measure the player experience of the levels as it is well suited for measuring enjoyment as well as other metrics about the player experience. Since the levels were not very long,

Algorithm 1 Reachability Repair Algorithm

```
For every column $C_1$ in the level variant:
    1) Identify tiles the player can stand on
    2) Of these tiles, determine which can be reached from previous tiles. This can be done using a basic model of Mario's movement and iterating backwards through previous columns. For each previous column, check if Mario can jump from any of the reachable tiles to any tiles in $C_1$.
    3) If there are tiles that can be stood on but none are reachable, then perform a repair:
        a) Determine the original location of $C_1$ in the original level.
        b) Iterate backwards through columns preceding $C_1$ in the original level.
        c) For each of these columns, determine if inserting this column into the level variant directly before $C_1$ would fix the reachability issues. This is done by checking if tiles in this column could be reached from previous columns in the level variant, then checking if tiles in $C_1$ can be reached from reachable tiles in this column.
        d) Once a column from the original level is selected, insert it into the level variant directly before $C_1$.
```

the In Game Module of the GEQ was used as it is a concise version of the Core Module. The survey, as well as responses, can be found in Appendix A.

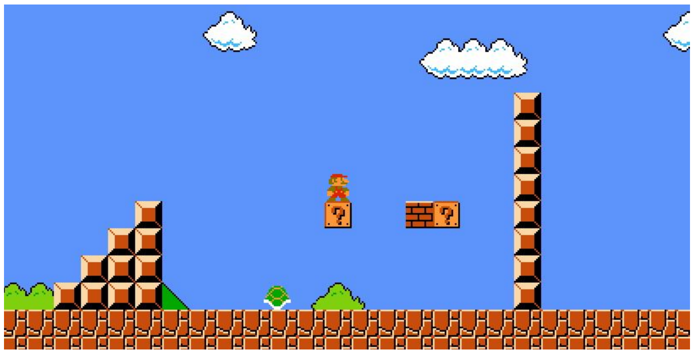


Fig. 4. This is an obstacle found in the level variant created with a large time step. By combining components from different obstacles, a new obstacle is created which requires a precise jump to overcome. This obstacle is very different to and requires more precision than any obstacles in the original level.

VII. Results

Survey results, seen in Figure 5, show that player enjoyment of the variants generated by chaotic variations matched enjoyment of the original level. Differences in score distributions between the variants and the original level were not statistically significant, with the exception of the flow score for the variant with a large time step ($p < .05$). Flow measures how much a player is absorbed in playing the game. This increase can likely be attributed to new obstacles being created by combining small components of obstacles from the original level. An example can be seen in Figure 4, where a new obstacle was created that required a precise jump to overcome. This shows that chaotic variations can not only produce levels that players enjoy as much as the original, it can also produce levels where the player's experience is notably different to playing the original level. Statistical results can be seen in Appendix B.

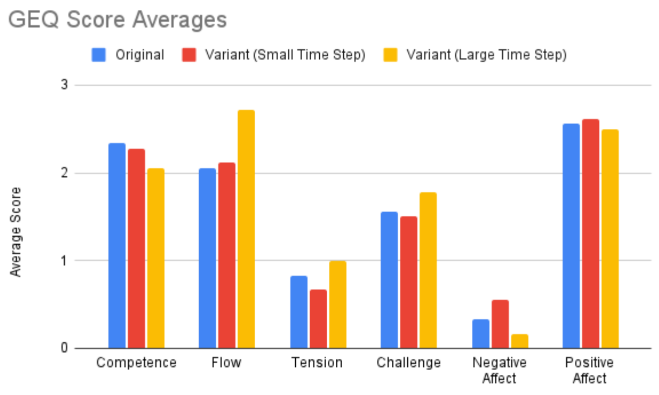


Fig. 5. Comparison of GEQ scores for World 1-1 from Super Mario Bros and two variants created using chaotic variations.

VIII. Conclusion

This paper has shown how chaotic variations can be used as a technique for PCG. In this case, the technique was able to produce levels that were very similar to the original, as well as levels that were very different. Player enjoyment of new levels matched enjoyment of the original level, and variants contained obstacles not seen in the original level, creating new experiences for players. Only requiring a single example level and potentially a repair algorithm, this technique can be used to create level variants more easily than other PCG methods. In a single game, many levels may have their own styles and types of obstacles. This technique could be used on each level to create variants without developing a PCG method for each level.

Some of the success in the application of this technique can be attributed to the ability to break the level into very small pieces to be represented in the symbol sequence. Symbols often represented a component of an obstacle rather than the entire obstacle, which allowed new levels to contain new obstacles created by combining components of entirely separate obstacles from the original level. In many games, breaking the level into such small components may not be possible. This is especially true in 3D games, where breaking 3D meshes apart could cause a number of issues, including gaps in the geometry of the level. In these cases, application of this technique would need to be more limited and could lead to differing rates of success.

References

[1] *Super Mario Bros.* Nintendo, 1985.

[2] Diana S. Dabby. Musical variations from a chaotic mapping. *Chaos*, 6(2):95, 1996.

[3] Elizabeth Bradley and Joshua Stuart. Using chaos to generate variations on movement sequences. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 8(4):800–807, 1998.

[4] Fatih Gürler and Esin Onbaşioğlu. Applying perlin noise on 3d hexagonal tiled maps. In *2022 International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, pages 670–673, Ankara, Turkey, 2022.

[5] Joseph Alexander Brown, Bulat Lutfullin, and Pavel Oreshin. Procedural content generation of level layouts for hotline miami. In *2017 9th Computer Science and Electronic Engineering (CEEC)*, pages 106–111, 2017.

[6] Diaz-Furlong Hector Adrian and Solis-Gonzalez Cosio Ana Luisa. An approach to level design using procedural content generation and difficulty curves. In *2013 IEEE Conference on Computational Inteligence in Games (CIG)*, pages 1–8, 2013.

[7] SangGyu Nam and Kokolo Ikeda. Generation of diverse stages in turn-based role-playing game using reinforcement learning. In *2019 IEEE Conference on Games (CoG)*, pages 1–8, 2019.

[8] Matthew Siper, Ahmed Khalifa, and Julian Togelius. Path of destruction: Learning an iterative level generator using a small dataset. In *2022 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 337–343, 2022.

[9] Jin Zhang, Tianhan Gao, and Qingwei Mi. Side-scrolling platform game levels reachability repair method and its applications to super mario bros. In *2022 RIVF International Conference on Computing and Communication Technologies (RIVF)*, pages 232–237, 2022.

[10] W.A. IJsselsteijn, Y.A.W. de Kort, and K. Poels. *The Game Experience Questionnaire*. Eindhoven University of Technology, 2013.

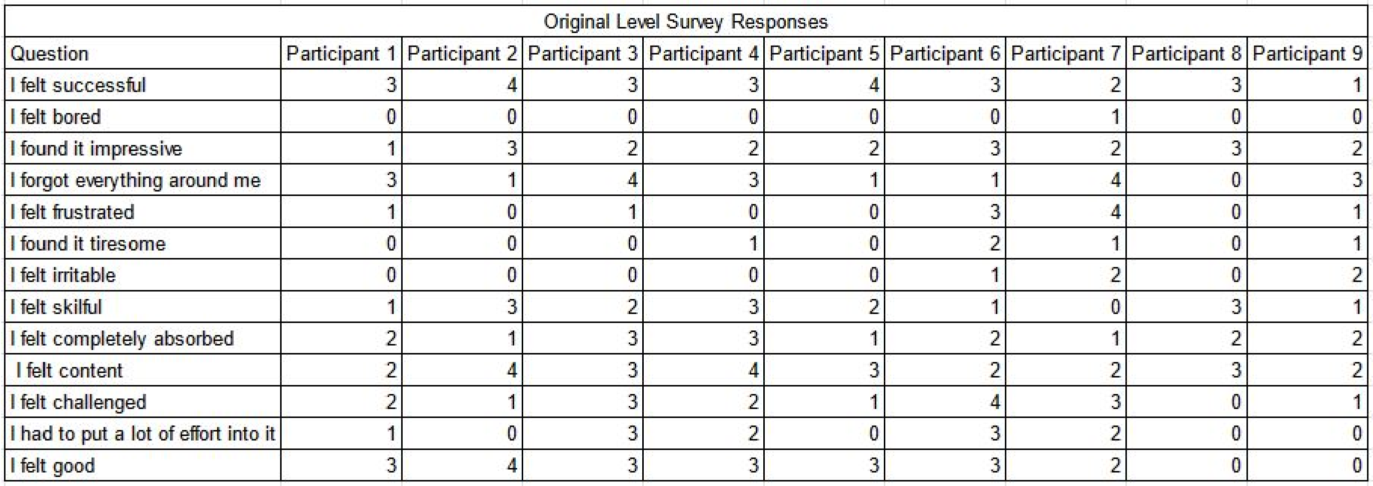
IX. Appendix

A. Survey and Responses

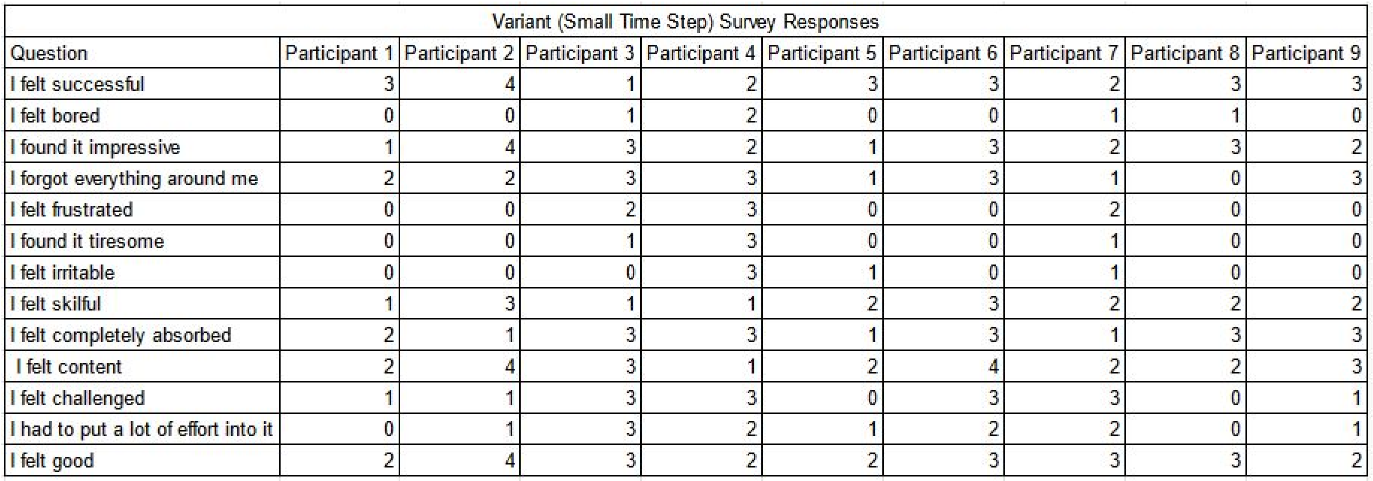
Participants were asked to answer the survey after completing each level. Responses were on the following scale:

0 - not at all, 1 - slightly, 2 - moderately, 3 - fairly, 4 - extremely

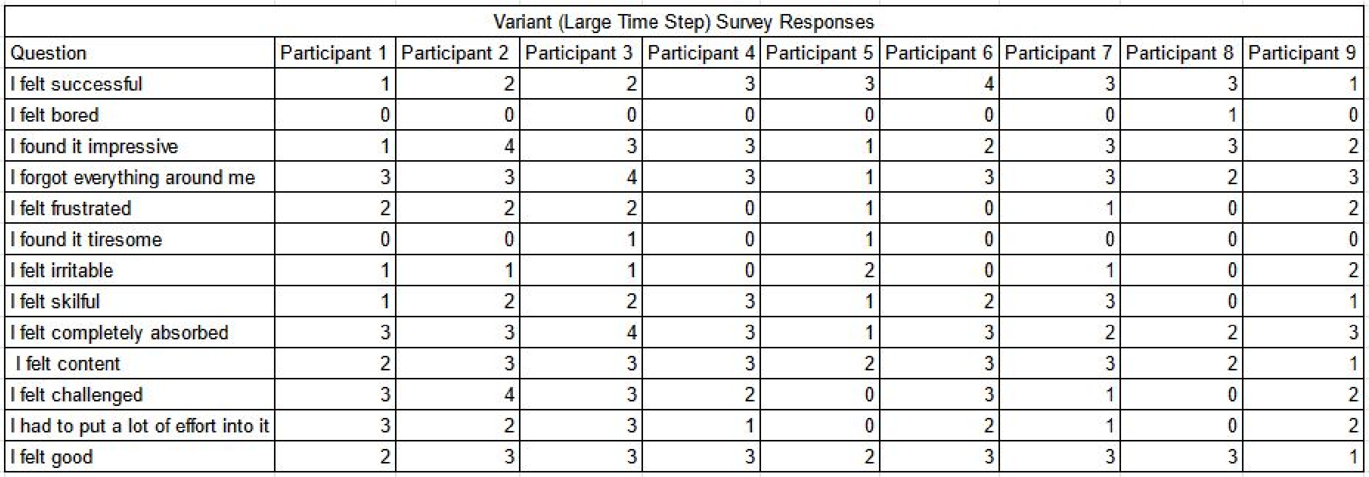
Responses to the survey for each participant are displayed below.



Original Level Survey Responses									
Question	Participant 1	Participant 2	Participant 3	Participant 4	Participant 5	Participant 6	Participant 7	Participant 8	Participant 9
I felt successful	3	4	3	3	4	3	2	3	1
I felt bored	0	0	0	0	0	0	1	0	0
I found it impressive	1	3	2	2	2	3	2	3	2
I forgot everything around me	3	1	4	3	1	1	4	0	3
I felt frustrated	1	0	1	0	0	3	4	0	1
I found it tiresome	0	0	0	1	0	2	1	0	1
I felt irritable	0	0	0	0	0	1	2	0	2
I felt skilful	1	3	2	3	2	1	0	3	1
I felt completely absorbed	2	1	3	3	1	2	1	2	2
I felt content	2	4	3	4	3	2	2	3	2
I felt challenged	2	1	3	2	1	4	3	0	1
I had to put a lot of effort into it	1	0	3	2	0	3	2	0	0
I felt good	3	4	3	3	3	3	2	0	0



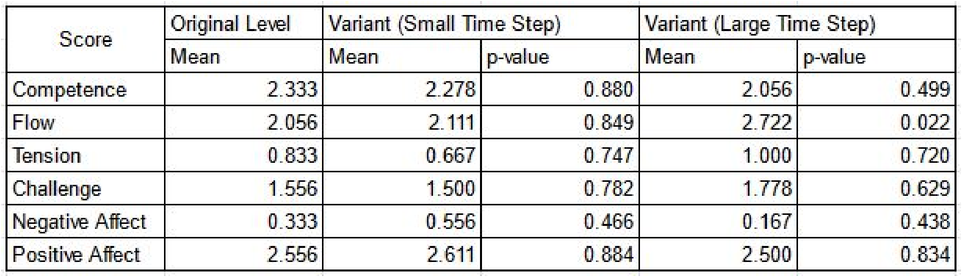
Variant (Small Time Step) Survey Responses									
Question	Participant 1	Participant 2	Participant 3	Participant 4	Participant 5	Participant 6	Participant 7	Participant 8	Participant 9
I felt successful	3	4	1	2	3	3	2	3	3
I felt bored	0	0	1	2	0	0	1	1	0
I found it impressive	1	4	3	2	1	3	2	3	2
I forgot everything around me	2	2	3	3	1	3	1	0	3
I felt frustrated	0	0	2	3	0	0	2	0	0
I found it tiresome	0	0	1	3	0	0	1	0	0
I felt irritable	0	0	0	3	1	0	1	0	0
I felt skilful	1	3	1	1	2	3	2	2	2
I felt completely absorbed	2	1	3	3	1	3	1	3	3
I felt content	2	4	3	1	2	4	2	2	3
I felt challenged	1	1	3	3	0	3	3	0	1
I had to put a lot of effort into it	0	1	3	2	1	2	2	0	1
I felt good	2	4	3	2	2	3	3	3	2



Variant (Large Time Step) Survey Responses									
Question	Participant 1	Participant 2	Participant 3	Participant 4	Participant 5	Participant 6	Participant 7	Participant 8	Participant 9
I felt successful	1	2	2	3	3	4	3	3	1
I felt bored	0	0	0	0	0	0	0	1	0
I found it impressive	1	4	3	3	1	2	3	3	2
I forgot everything around me	3	3	4	3	1	3	3	2	3
I felt frustrated	2	2	2	0	1	0	1	0	2
I found it tiresome	0	0	1	0	1	0	0	0	0
I felt irritable	1	1	1	0	2	0	1	0	2
I felt skilful	1	2	2	3	1	2	3	0	1
I felt completely absorbed	3	3	4	3	1	3	2	2	3
I felt content	2	3	3	3	2	3	3	2	1
I felt challenged	3	4	3	2	0	3	1	0	2
I had to put a lot of effort into it	3	2	3	1	0	2	1	0	2
I felt good	2	3	3	3	2	3	3	3	1

B. Statistical Analysis of Survey Results

Scores for each participant were computed according to the GEQ guidelines. Two tailed t tests were performed for the distributions of scores for each metric to compare results for the variants against results for the original level. Only the flow score for the variant with a large time step was statistically significant.



Score	Original Level	Variant (Small Time Step)		Variant (Large Time Step)	
	Mean	Mean	p-value	Mean	p-value
Competence	2.333	2.278	0.880	2.056	0.499
Flow	2.056	2.111	0.849	2.722	0.022
Tension	0.833	0.667	0.747	1.000	0.720
Challenge	1.556	1.500	0.782	1.778	0.629
Negative Affect	0.333	0.556	0.466	0.167	0.438
Positive Affect	2.556	2.611	0.884	2.500	0.834