International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

Further Enhancement of Planned Random Algorithm Applied in Character Creation for Video Games

Lance Jubilee Jumaquin[1], Tom Harvey Dionisio Licudo[2], Florencio V. Contreras Jr[3], Jamillah S. Guialil[4], Leisyl M. Mahusay[5], Jonathan C. Morano[6], Vivien A. Agustin[7]

[1,2,3,4,5,6,7]CISTM, Pamantasan Lungsod ng Maynila, Philippines

Abstract

This paper introduces the Further Enhanced Planned Random Algorithm (FEPR Algorithm) applied in character creation for video games, addressing limitations of the previous Enhanced Planned Random Algorithm by employing various enhancements. The study's purpose was to further increase the apparent randomness by reducing the repetitions of data results. Employing an experimental quantitative research design, the study used the C++ programming language and libraries such as the "random" library which includes the Mersenne Twister PRNG (Pseudocode Random Number Generator). The effectiveness of the FEPR Algorithm was tested against the 2022 Enhanced Planned Random Algorithm through a series of 1000 iterations in 5 test cases, evaluating the randomness of the generated sequences. Results indicated an 84.47% reduction in pattern repetitions in both the total and average repetitions with the FEPR Algorithm compared to the control. This study's conclusion posits that the FEPR Algorithm successfully further increases the apparent randomness compared to the EPR algorithm. Recommendations include extending the research to other gaming elements and fields where randomness is crucial. The research implications discuss the advancement of randomization processes in algorithmic design. Practical implications highlight the potential for improved game design and player experience, while social implications consider the broader impact on digital entertainment and diversity.

Keywords: Character Creation, Algorithm Enhancement, Planned Random Algorithm, PRNG (Pseudo-Random Number Generation), Fischer-Yates algorithm

INTRODUCTION

Randomness, defined by the absence of predictable patterns, holds significant importance across various applications such as statistics, encryption, science, and gaming. The generation of random numbers, integral to these applications, often relies on standard algorithms embedded in programming libraries. However, the apparent unpredictability of computer-generated random numbers can be compromised by inherent patterns or repetitions, a consequence of the systematic nature of computers. Overcoming this challenge requires addressing the human tendency to perceive meaningful connections or patterns, known as apophenia, even in random datasets.

In response, the Planned Random Algorithm was introduced, seeking to heighten the apparent randomness of generated sequences and counteract the human inclination to discern patterns within them. This algorithm deviates from conventional methods by incorporating a planning element in the value-returning process, disrupting any perceived patterns. Leveraging time-based randomization, the algorithm demonstrates efficiency comparable to the widely used Modern Fisher–Yates Algorithm, sharing the same time complexity of O(n) ().

While the current iteration of the Planned Random Algorithm is adept at enhancing the randomness of specific numerical lists, its limitations surface when applied to more diverse datasets, such as those encountered in video games. Shuffling elements like character features without considering crucial attributes—such as face, skin complexity, size (height & width), starter clothes (from head to toe), etc.—may result in suboptimal randomization. The algorithm's evolution is imperative, drawing inspiration from Apple's "Smart Shuffle" feature in playlist management, which considers factors like song artist and album to enhance randomness [3].

For this simulation, the element list consists of 10 elements, each with 3 attributes. This is then looped by 1000 to insert records in the character list and have a dataset of 1000. The algorithm then shuffles for 1000 iterations with each iteration is a new shuffled list.

In the first part of the above code snippet simulation, the structure of the element consists of only three attributes and does not limit the number of times each element is checked. While this has no issue in a less complex dataset, it poses a problem when it comes to a more complex and larger dataset where an element may consist of more than 3 attributes which can lead to a loss in performance adaptability, and a less even distribution of elements in the shuffled list.

The Pseudocode Random Number Generator (PRNG) has a significant role in producing random numbers for a much random shuffling. However, the current PRNG used in the existing algorithm is using the srand() in the stdlib.h library. In such cases, given its inefficiency, this may produce predictable patterns during the process potentially making the output appear less random.

During the shuffling process, the algorithm does not take account of the number of times each element has been selected making the selection inaccurate and unfair between elements, resulting in the elements with the same attributes appear more consecutively.

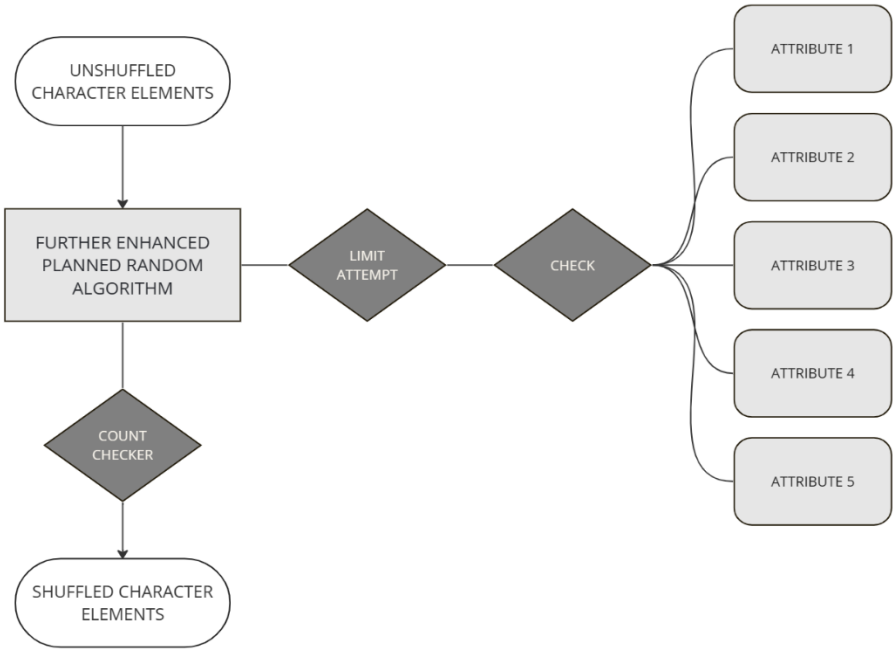

Fig. 2 Visualization of the Further Enhancement of Planned Random Algorithm

LITERATURE REVIEW

Randomization in Video Games

In the realm of video games, injecting a sense of variety and thrill often involves the strategic use of random numbers. These numbers play a dual role, orchestrating change in how the game unfolds. On one hand, they influence the game's outcomes, determining the success of actions– be it hitting a target in X-COM, dealing bonus damage in League of Legends, or drawing a crucial card in Hearthstone. On the other hand, they shape the game's starting conditions, weaving a unique tapestry for each playthrough. Think of them as the architects behind distinctive characters in Lord of the Rings Shadow of Mordor, the architects of intricate narratives in FTL, or the architects of diverse game content as seen in Borderlands. It is this delicate dance with randomness that breathes life into the gaming experience, ensuring no two journeys through the digital realms are quite the same. (Random Numbers in Gaming, n.d.) Additionally, random numbers in video games bring diversity, complexity, realism, and replayability to the gameplay while also keeping players involved (How, Why and When Random Numbers Are Used in Video Games..., n.d.). Based on the same article they expanded the following

1. **Randomized events keep games fresh**

Random numbers enable developers to generate unpredictable events and occurrences in the game environment. These can be opponent spawning, loot dumps, or bullet hits, but the overall

2. **Increase replayability**

Random numbers can be used to produce procedural material like randomized maps or levels. This allows players to explore new settings every time they play, increasing the sensation of discovery while making it less determined and hence more difficult to beat. This implies fewer patterns and more opportunities for replay.

3. **For simulation and realism**

Random numbers can be utilized to create minor fluctuations in forces, velocities, and trajectories, resulting in more realistic motions and collisions. This is particularly useful for simulating complicated systems such as fluid dynamics and fabric simulations. Again, they might be useful in creating various

environmental effects such as weather systems or natural occurrences like wind gusts or turbulence, making the scene feel more alive and immersive. Furthermore, to provide more realistic human-like behavior in a game's non-player characters (NPC), their decision-making process may be altered utilizing randomization by integrating elements such as selecting between several alternatives, reacting to changing situations, and so on.

Role of Randomized Events

This article provides an overview of procedural generation, a technique for creating data using algorithms, in the context of game development. The author explains the history, uses, guidelines, and pros and cons of procedural generation, and the difference between random and procedural generation. The author also demonstrates a simple algorithm for generating a grid of objects in Unity. The article aims to teach the reader about the basics and benefits of procedural generation and encourage them to use it in their own games. (Kenny, 2021)

Pseudo-Random Number Generators

Algorithms for generating random numbers are frequently utilized in simulation, computer science, and many other domains. Nevertheless, most of these methods are pseudo-random rather than completely random. This implies that they generate a random sequence of integers that is, in fact, predetermined by a preset beginning point known as the seed value (What Is the Significance of the Seed Value in Random Number Generation Algorithms?, n.d.). The article explained that seed value is important. Applications that need secrecy or unpredictability may encounter issues since the seed value, which is essential to the pseudo-randomness of the PRNG, can be anticipated if known. Adding to this, the article stated that the PRNG Mersenne Twister is more complex and slower but is high quality and secure.

A pseudo-random number generator is typically used to generate randomness in games. The random number is created by selecting a seed number and entering it into an algorithm. Because the seed directly affects the pattern of random numbers produced by the computer, it must be carefully chosen. Certain games, like Pokemon Emerald, have a set seed of zero, which causes patterns to emerge because the numbers are generated in the same order. This has the effect of making random events predictable, and many have taken advantage of this to guarantee they always obtain the finest Pokemon. (Random Numbers in Gaming, n.d.)

In a blog post (Generating Random Numbers Using C++ Standard Library: The Problems, 2020), he mentioned the problems of the C++ standard library when it comes to generating random numbers. One of the problems mentioned is the resulting numbers will not be uniformly distributed. The results will be biased towards lower numbers, due to modulo use. The second difficulty is easier to understand, although it is more obscure. The range that std::rand can produce is defined as [0, RAND_MAX), where RAND_MAX is any constant that is greater than or equal to 32767. Platforms that employ this lower bound [1] will never see the above example print a value greater than 32767.

METHODOLOGY

The approach taken in this research is a quantitative experimental method, specifically designed to assess the limitations of the enhancements on the existing Planned Random Algorithm. The experimental design was bifurcated into two simulations: a control simulation featuring the 2022 Enhanced Planned Random Algorithm and an experimental simulation incorporating the Further Enhanced Planned Random Algorithm. The latter simulation incorporated additional enhancements such as increasing the scale of attributes into the algorithm making the distribution of elements more even, replacing the “rand” with

"mt19937" or the Mersenne Twister PRNG for a much high-quality random numbers generation, and adding a count checker for a fair selection of elements.

The foundation of the experiments relied on a dataset that was specially curated for character creation in video games, derived from the one used by the 2022 Enhanced Planned Random Algorithm. Each simulation utilized this dataset to evaluate the changes made to the algorithm. The simulation parameters, such as the number of attributes, dataset size, and specific algorithmic enhancements, were carefully controlled and varied to measure their influence on the algorithm's performance.

Following the same test case of (Jemuel Fontila, et.al 2022), an array of 50-character elements were shuffled repeatedly to determine the robustness of the randomness in the output sequences, with the same 1st attribute but various 2nd, 3rd, 4th, and 5th attribute used in all test cases. These attributes are based on what a video game character can have. For this test the attributes are as follows: the character Race, Body Size, Arm Length, Footwear, and Skin Color. 1000 iterations were run for each simulation, and the experiments were replicated five times to ensure reliability. The generated sequences were then analyzed for patterns and repetitions, comparing the current shuffled list with the previous iterations to ascertain improvements in randomness.

The effectiveness of the Further Enhanced Planned Random Algorithm was gauged by a comparative analysis with the control algorithm, evaluating the variations in performance and the algorithm's success in reducing the repetitions and increasing apparent randomness. The comprehensive results from this methodical experimental design aimed to substantiate the enhancements made to the algorithm and their practical implications for video game character creation.

RESULTS

The table below shows the simulation results comparison between the Enhanced Planned Random Algorithm and the proposed further enhancement regarding the total and average repetitions.

ENHANCED PLANNED RANDOM ALGORITHM		**PROPOSED FURTHER ENHANCEMENT**	
Simulation	**Values**	**Simulation**	**Values**
1	5649	**1**	2293
2	5653	**2**	2261
3	5658	**3**	2294
4	5655	**4**	2366
5	5662	**5**	2269
Total Repetitions: 28,277		**Total Repetitions:** 11,483	
Avg. Repetitions: 5,655.4		**Avg. Repetitions:** 2,296.4	

***Table 1. Results of the* Further Enhancement of Planned Random Algorithm and the Enhanced Planned Random Algorithm**

DISCUSSION

The results show that the Existing Enhanced Planned Random Algorithm in 5 simulation test runs has a total repetition of 28,277 with an average of 5,655.4 repetitions. The Proposed Further Enhancement on the other hand has a total repetition of 11,483 with an average of 2,296.4 repetitions, signifying a profound enhancement in increasing the algorithm's apparent randomness.

The data gathered from these simulations resulted in an 84.47% decrease in both the total repetitions and average repetitions per case from the Enhanced Planned Random Algorithm, indicating that the Proposed Further Enhancement significantly increased the shuffle's apparent randomness.

CONCLUSIONS AND RECOMMENDATIONS

This study concludes that the Further Enhanced Planned Random Algorithm can make significant use in the field of gaming which employs character creation. With the enhancements, the algorithm shuffles the character elements more balanced and evenly distributed that the game developers can take advantage of in having a randomized character creation. Making the gaming experience of the players much more immersive and engaging, with a random character experience in each of their playthrough.

Simulation tests were conducted to evaluate this new algorithm against the previously established Enhanced Planned Random Algorithm. The outcomes of these tests were notably impactful, highlighting a dramatic decrease in both the total and average repetitions of data- with the total repetitions plummeting from 28,277 to 11,483, and the average repetitions dropping from 5,655.4 to 2,296.4. This transition denotes an 84.47% reduction in repetitions.

These findings solidify the Further Enhanced Planned Random Algorithm's role in enhancing randomness, effectively reducing predictable patterns, and fostering the production of distinct and varied characters. This advancement is critical for delivering richer and more engaging gaming experiences to players.

Future research should continue to refine this algorithm, focusing on its application across a wider range of gaming elements beyond character creation. There is also an opportunity to explore the algorithm's effectiveness in other fields requiring randomness, such as encryption or data sampling. Moreover, integrating feedback from game developers and players could provide insights into practical implications and areas for further enhancement. Lastly, comparisons with other random algorithms could offer a broader perspective on the algorithm's efficiency and effectiveness in diverse applications.

IMPLICATIONS

Research Implications

The results of this study reinforce the importance of algorithmic sophistication in random number generation for character creation in video games. Further research could investigate the adaptability of the Further Enhanced Planned Random Algorithm (FEPR Algorithm) across various game genres and its scalability in larger datasets, potentially illuminating new pathways for algorithmic design in gaming and other industries reliant on randomization.

Practical Implications

The FEPR Algorithm has significant applications in the realm of video game development, where the demand for unique and diverse character creation is high. The reduction in pattern repetition and increased randomness could lead to more engaging gaming experiences, offering a broader array of outcomes for character generation. This improvement might not only apply to character creation but also to the generation of dynamic environments and unpredictable gameplay scenarios, which are crucial for player retention and satisfaction. Incorporating this enhanced algorithm could lead to innovations in the development of non-player character behavior, offering more human-like unpredictability and complexity in interactions (Fontila et al., 2022; Random Numbers In Gaming, n.d.).

Social Implications

On a societal level, the advancements in randomization algorithms have the potential to enhance the immersive experience of video games, contributing to the medium's cultural and entertainment value. As games become more reflective of the diverse and complex nature of life, they offer richer narratives and experiences that can impact social interaction, storytelling, and the exploration of virtual spaces, paralleling the diversity and unpredictability of real-world interactions (How Why and When Random Numbers Are Used in Video Games..., n.d.).

ACKNOWLEDGEMENT

This study became a reality with great participation and support from many people. The researchers would like to dedicate their accomplishments in this study to every one of them to express their genuine gratitude and heartfelt appreciation. To Almighty God, for bestowing the researcher's wisdom, understanding, fortitude, will, and knowledge for the completion of the study. Without His presence, this study would not have been a success. To Sir Morano and Sir Contreras, for providing us with the gift of education, the chance to learn, and the encouragement that enabled us to complete this research. To Alma Mater, Pamantasan ng Lungsod ng Maynila, PLM College of Information System and Management (PLM CISTM), in accordance with the requirements of the Bachelor of Science in Computer Science. Also, we would like to thank all our relatives, family, and friends who supported us in one way or another.

FUNDING

The study did not receive funding from any institution.

DECLARATIONS

Conflict of Interest

The researcher declares no conflict of interest in this study.

Informed Consent

Not applicable for this study, as it focuses solely on the development and testing of a computational algorithm. There were no human participants from whom to obtain informed consent, as the research did not involve any personal or sensitive data collection that would require it.

Ethics Approval

This research did not require ethics approval since it did not involve human subjects, personal data, or any intervention that would affect individual rights or privacy. The study was conducted in compliance with all relevant institutional and governmental guidelines concerning computational simulation and algorithm testing.

REFERENCES

1. Alex. (2022, February 18). Learncpp.com. [Webpage]. Retrieved from https://www.learncpp.com/cpp-tutorial/introduction-to-random-number-generation/?fbclid=IwAR0sFO1idmk8jgiuaZGz1Pgtz4QVeWKOzlVe8dSaVtBvKtyTdez1w1wgZow
2. Random Numbers In Gaming. (n.d.). Quantum Base. [Webpage]. Retrieved January 11, 2024, from https://quantumbase.com/sse/gaming/#:~:text=Randomness%20in%20games%20are%20mainly
3. How, Whyand When Random Numbers are used in Video Games... (n.d.). Www.linkedin.com. [Webpage]. Retrieved January 11, 2024, from https://www.linkedin.com/pulse/how-why-when-random-numbers-used-video-games-kavindu-priyanath

4. What is the significance of the seed value in random number generation algorithms? (n.d.). Www.linkedin.com. [Webpage]. Retrieved January 11, 2024, from https://www.linkedin.com/advice/1/what-significance-seed-value-random-number-generation-xvxzc
5. Hořeňovský, M. (2020, May 17). Generating random numbers using C++ standard library: the problems. The Coding Nest; The Coding Nest. [Webpage]. Retrieved from https://codingnest.com/generating-random-numbers-using-c-standard-library-the-problems/?fbclid=IwAR1iZWjEqggD4Voby-1Yh_4mNuKMFP-9YYVTpTWZ2_wYIQmxd08bpvjYf3U
6. Jemuel Fontila, Jiro Mark Garcia, Mark Jimwell Lagarta, Mark Christopher Blanco, & Michael Cortez. (2022). An enhancement of planned random algorithm applied on an online questionnaire. South Asian Journal of Engineering and Technology, 12(1), 121–125. https://doi.org/10.26524/sajet.2022.12.018
7. Pant, A., Agrawal, K., & Tripathy, B. K. (2018). Planned random algorithm. Advances in Intelligent Systems and Computing, 701, 119–127. https://doi.org/10.1007/978-981-10-7563-6_13
8. Properly seeding the Mersenne Twister. (2023). Olemiss.edu. [Webpage]. Retrieved from https://www.phy.olemiss.edu/~kbeach/guide/2020/01/11/random/?fbclid=IwAR0iaU5DxKreKntVHxhT1fDpEBqbx3TXx0dG2zR5jXFs06Z_Wtvzni1H5-c
9. std mt19937 Class in C. (2021, March 25). GeeksforGeeks; GeeksforGeeks. [Webpage]. Retrieved fromhttps://www.geeksforgeeks.org/stdmt19937-class-in-cpp/?fbclid=IwAR22_Jer8yuqQ1ES2orUtnmbvJe2U-7dKmrHWTUJk3pFu5GojTF48HRxV3U
10. Bellos, A. (n.d.). And now for something completely random, by ALEX BELLOS. Mail Online. [Webpage]. Retrieved from https://www.dailymail.co.uk/home/moslive/article-1334712/Humans-concept-randomness-hard-understand.html
11. Cohen, B. (2020). Spotify Madeits Shuffle featureless random sotha it would actually feel more random to listeners here's why. Business Insider. [Webpage]. Retrieved from https://www.businessinsider.com/spotify-made-shuffle-featureless-random-to-actually-feel-random-2020-3?international=true&r=US&IR=T
12. Dijana Oreški, Stjepan Oreški, & Božidar Kliček. (2017). Effects of dataset characteristics on the performance of feature selection techniques. Applied Soft Computing, 52, 109–119. https://doi.org/10.1016/j.asoc.2016.12.023
13. Unlocking the Secrets: Why Shuffle Algorithms Play a Crucial Role in Modern Computing. (2023, May 7). Locall.host. [Webpage]. Retrieved from https://locall.host/why-shuffle-algorithm/
14. What is the significance of the seed value in random number generation algorithms? (n.d.). Www.linkedin.com. Retrieved January 11, 2024, from https://www.linkedin.com/advice/1/what-significance-seed-value-random-number-generation-xvxzc
15. Understanding the Weighted Random Algorithm. (2023, October 24). DEV Community. https://dev.to/jacktt/understanding-the-weighted-random-algorithm-581p#:~:text=Imagine%20you%20have%20a%20collection