Niğde Ömer Halisdemir Üniversitesi Mühendislik Bilimleri Dergisi
Niğde Ömer Halisdemir University Journal of Engineering Sciences

Araştırma makalesi / Research article

www.dergipark.org.tr/tr/pub/ngumuh / www.dergipark.org.tr/en/pub/ngumuh

Enhancing wave function collapse algorithm for procedural map generation problem

Prosedürel harita oluşturma problemi için dalga fonksiyonu yığılma algoritmasının geliştirilmesi

Osman Büyükşar[1], Doğan Yıldız[2,*], Sercan Demirci[3]

[1,3] *Ondokuz Mayıs Üniversitesi, Bilgisayar Mühendisliği Bölümü, 55139, Samsun, Türkiye*
[2] *Ondokuz Mayıs Üniversitesi, Elektrik-Elektronik Mühendisliği Bölümü, 55139, Samsun, Türkiye*

Abstract
In this study, the Improved Map Generation Algorithm (IMGA) method is presented to improve traditional methods in procedural map creation. Traditional procedural map generation techniques using noise generation exhibit shortcomings in the consistent composition of a real map with its uniformly distributed features. On the other hand, procedural map creation techniques that use wave function collapse require that some map pieces already exist to create a map. The observed disadvantages were eliminated by using a hybrid technique with the designed IMGA method. The developed algorithm is similar to real maps in terms of the distribution of map regions, does not need 3D model parts, and performs map creation operations without increasing the algorithm's time complexity. The evaluation of IMGA was carried out by coding the method into the Unity game engine.

Keywords: Procedural terrain generation, Wave function collapse, Noise, Algorithm.

Öz
Bu çalışmada, prosedürel harita oluşturma alanındaki geleneksel metotları geliştirmek amacıyla Geliştirilmiş Harita Oluşturma Algoritması (Improved Map Generation Algorithm, IMGA) metodunu sunulmuştur. Gürültü üretme yöntemini kullanan geleneksel prosedürel harita oluşturma teknikleri, düzgün dağılmış özellikleriyle gerçek bir haritanın tutarlı bileşiminde eksiklikler sergiler. Öte yandan dalga fonksiyon çöküşü kullanan prosedürel harita oluşturma teknikleri ise harita oluşturabilmek için harita parçalarının bir kısmının hâlihazırda bulunmasını gerektirmektedir. Tasarlanan IMGA metoduyla hibrit bir teknik kullanılarak gözlemlenen dezavantajlar giderilmiştir. Tasarlanan algoritma, harita bölgelerinin dağılımı açısından gerçek haritalara benzeyen, 3D model parçalarına ihtiyaç duymayan ve harita oluşturma işlemlerini algoritma zaman karmaşıklığını arttırmadan gerçekleştirmektedir. IMGA'nın değerlendirilmesi ise, metodun Unity oyun motoruna kodlanması ile gerçekleştirilmiştir.

Anahtar kelimeler: Prosedürel harita oluşturma, Dalga fonksiyonu yığılma, Gürültü, Algoritma.

1 Introduction

Recently, there has been a substantial escalation in the dimensions and details of maps within the gaming industry. Therefore, the process of designing and modeling extended maps requires a more extensive investment [1, 2]. While the option to counter this by opting for smaller or less intricate maps exists, such a choice is deemed unfavorable due to its potential to curtail gameplay duration.

Creating maps and terrains can be achieved by applying procedural content generation techniques. This approach involves using algorithmic generation mechanisms instead of manual content creation. Procedural content generation (PCG) finds utility in producing diverse forms of content, spanning from three-dimensional models and textures to animated sequences and auditory elements. This methodology has gained substantial traction across different domains, notably in the domains of gaming and cinematic production, facilitating the efficient generation of substantial volumes of content. Illustrative instances of its implementation can be observed in gaming titles like "No Man's Sky," "Terraria," and "Minecraft," as well as in cinematic productions such as "The Lord of the Rings" and "Avatar."

PCG has gained significant attention in academic studies as well. These studies have been directed toward enhancing established PCG methodologies, aiming to yield superior content outcomes with reduced computational overhead. Conventional techniques for procedural content generation include noise generation [3], cellular automata [4], replacement grammars [5], optimization algorithms [6, 7], and neural networks [8]. Unlike AI-driven models like WASPAS and GANs, which have gained attraction for their ability to produce terrain resembling real-world geography, particle swarm optimization (PSO) [6] offers a distinct advantage in its capacity to provide finer-grained control over the generation process. While AI models often yield impressive results, their reliance on complex algorithms sometimes leads to less malleable terrain, limiting flexibility

* Sorumlu yazar / Corresponding author, e-posta / e-mail: dogan.yildiz@omu.edu.tr (D. Yıldız)
Geliş / Recieved: 16.09.2023 Kabul / Accepted: 26.04.2024 Yayımlanma / Published: 15.07.2024
doi: 10.28948/ngumuh.1361413

for modifications such as generating neighboring tiles. While PSO excels at providing control and adaptability, it may require extensive parameter tuning for better results.

PCG can be categorized into subtopics based on the specific type of content being generated. One such subcategory is procedural terrain generation (PTG), which involves the creation of terrains through various methodologies. A prominent and pervasive technique uses a digital elevation map (DEM) to utilize terrain objects. A DEM comprises a two-dimensional grid containing elevation values within designated grid cells. The generation of a DEM employs diverse algorithms, among which neural networks and noise generation algorithms hold significance. Noise generation, recognized for its prevalence, comprises a range of algorithms differing in quality, memory demands, and processing speed. These algorithms are the diamond-square algorithm, Worley noise, value noise, simplex noise, and Perlin noise [9].

The studies use noise generation with a combination of other techniques because noise alone results in uniform terrain without special features when used as a DEM. In a recent investigation by [10], noise generation is enriched through the implementation of octaves applied to Perlin noise. While the introduction of octaves introduces intricacy to the resultant maps, it is noted that features remain evenly dispersed across larger scales. An alternative strategy involves the integration of biomes to shape maps that align with specific biome boundaries [11]. It is worth noting that while this approach demands higher computational resources, it effectively mitigates the uniformity observed in maps generated solely through noise techniques.

In the modern era, driven by the rapid progress of Artificial Intelligence (AI), there have emerged instances of terrain and game level generation employing machine learning models like WASPAS and General Adversarial Networks (GANs) [8, 12]. Notably, the utilization of satellite imagery for training GAN models and subsequently generating DEMs has gained popularity, primarily due to its capacity to yield maps resembling real-world geographical layouts [8, 13, 14]. However, while the outcomes derived from AI models indeed offer enhanced terrain, a notable limitation is their need for more direct control over the terrain generation process. Consequently, the generated terrains become less malleable and pose challenges when attempting modifications such as generating neighbor tiles.

Erosion-based simulation has also emerged as a noteworthy approach to generating terrain that resembles actual landscapes [15, 16]. For instance, Jacob Olsen's research leveraged noise generation algorithms alongside erosion simulation techniques to enhance the authenticity of the generated terrain. Furthermore, hydrology-based methodologies have been explored, yielding comparable outcomes [17, 18]. The result has been the creation of maps that closely mimic real-world topographies. However, due to their foundation in noise-based techniques, these generated terrains face challenges in terms of malleability to suit designer preferences. Additionally, their applicability to infinitely scalable real-time maps is constrained as they struggle to produce coherent neighboring maps.

Conversely, [19] presents an approach for terrain generation that adheres to constraints defined by designers. Their methodology employs terrain reasoning agents, which operate under the influence of designer-specified constraints to produce terrains. Consequently, this method yields algorithms subject to designer control; however, the outcomes deviate significantly from real-world terrains regarding resemblance.

The core problem lies in the fact that procedural maps generated using noise algorithms yield outcomes characterized by uniformity and even distribution, a departure from the intricate variations found in real-world terrains. In contrast, GANs yield more favorable results; however, their effectiveness is confined to maps boasting extensive training data. Furthermore, even after training, these models remain incapable of generating infinitely expansive terrains automatically because they can't generate neighbor maps.

Recently, a novel procedural generation technique called "wave function collapse" has emerged, circumventing the limitations observed in preceding methodologies, which are practical design constraints. This algorithm operates by taking an input map and generating content following the provided input. Research studied in [20] underscores the feasibility of customizability within this method, enabling user-driven modifications. Furthermore, a study in [21] demonstrates the algorithm's capability to execute while imposing specific states on predetermined cells. This evidence substantiates the method's potential to generate maps on a chunk-by-chunk basis, leveraging pre-defined forms of adjacent chunks [21]. Additionally, the algorithm displays an aptitude for producing more structured maps, as highlighted in the investigation by [22]. Nonetheless, it is noteworthy that this algorithm needs pre-modelled tiles provided by the user to facilitate content generation.

Cellular automata, the prominent method in procedural content generation, shares similarities with the wave function collapse technique and has been extensively explored over the years. A notable study introduces the utilization of layered cellular automata to generate structured and intricate maps [23]. However, a drawback of this approach is its susceptibility to falling into repetitive patterns since the underlying rule is predefined.

This study presents an algorithm for procedural terrain generation that exhibits captivating terrain formations and possesses scalability without reliance on pre-established tiles while maintaining real-time computational efficiency. The main contributions of this study can be summarized as follows:

- We introduce a hybrid approach that can be used with different noise generation algorithms and wave function collapse algorithms for further work in the field of terrain generation
- Unlike conventional methods such as wave function collapse and GAN-based algorithms, our approach is free from predefined 3D models, datasets, or any prior data.

- The terrains generated allow for customizations according to user preferences, distinguishing them from maps generated using conventional noise and GAN methods.

2 Material and method

2.1 Perlin noise

Perlin noise is a noise generation algorithm recognized for producing gradient noise, characterized by its smoothly varying values [3, 24]. This technique establishes a lattice of randomly oriented unit-length vectors across distinct directions. Subsequently, an interpolation procedure is applied among these unit vectors. At any given coordinate within a grid cell, the noise value is approximated by evaluating the dot product involving the displacement vectors of the respective grid cell corners and their corresponding unit-length gradient vectors. The resultant dot products are further subject to interpolation, yielding the final Perlin noise value.

2.1.1 Octaves

Perlin noise, in its standalone form, represents a uniform texture and lacks natural variability. To introduce a more intricate and authentic character, supplementary layers of noise are incorporated through octave summation along with the fundamental noise layer. These additional layers, termed octaves, are generated by multiplying the frequency of the Perlin noise with a factor greater than one while simultaneously adjusting the amplitude by a factor less than one. The cumulative outcome produces an octave endowed with distinct noise characteristics. In the context of generating DEMs, this aggregated noise contributes to the refinement of DEM quality, imbuing it with heightened intricacy and detail.

2.2 Wave function collapse

Wave function collapse (WFC) is a texture generation algorithm engineered to produce a texture akin to a provided input texture. Maxim Gumin developed it, and the algorithm was first made publicly accessible as a GitHub repository in 2016 through an initial implementation [25]. The pseudo-code of the base WFC algorithm is shown in Algorithm 1.

Algorithm 1: Base WFC Algorithm

```
1  Initialize cell grid
2  while there are cells to collapse
3      Select a cell to be collapsed
4      Collapse the selected cell
5      Update neighbour cells
```

The WFC algorithm starts by constructing a grid with cells set to superposition. Subsequently three steps are iterated for each cell, until there is no cells remain in superposition state. The primary step involves the identification of a cell hosting the minimal count of potential states, often referred to as the cell exhibiting the lowest entropy. Following this, a state is arbitrarily chosen from the possible states affiliated with the designated cell. After

establishing the cell's state, neighboring cells undergo updates according to the selected state. This cascade of updates extends to the neighbors' own neighbors, persisting until no further alterations to the potential states of neighboring cells are feasible.

2.3 Proposed method

The core problem is that the use of noise in DEM results in uniform terrain with evenly distributed features; the WFC algorithm is much more promising than the noise generation algorithm, but it necessitates the use of pre-designed tiles and adopts a tile-by-tile approach in generating the DEM. As a consequence of this approach, the resulting terrain exhibits visible tile boundaries, detracting from its resemblance to real-life landscapes. Hence, our goal with this proposed method is to generate expansive terrains that closely resemble real-life landscapes. We aim to achieve this without the uniform characteristics commonly associated with terrains generated using noise algorithms and without relying on pre-designed tiles, a requirement in the case of WFC.

By incorporating Perlin noise, we prevent the need for pre-modeled tiles and leverage its capacity to generate terrain. To rectify the issue of terrain uniformity and likeness to existing maps, we introduce a region map generated using a tailored wave function collapse algorithm. This map serves as a blueprint for adapting the terrain based on distinct region values, thereby reducing uniformity. Notably, this approach heightens the fidelity of the generated map by emulating real-world terrain characteristics, as regions are delineated in alignment with neighboring features. Ultimately, our approach sets itself apart from noise-based generated maps due to its distinct topographical regions and non-uniform nature. It also distinguishes itself from WFC-generated maps by eliminating discernible tile boundaries and negating the necessity for predefined tile models. Our proposed method, the Improved Map Generation Algorithm (IMGA), executes the steps in Figure 1. The pseudo-code of this algorithm is illustrated in Algorithm 2.

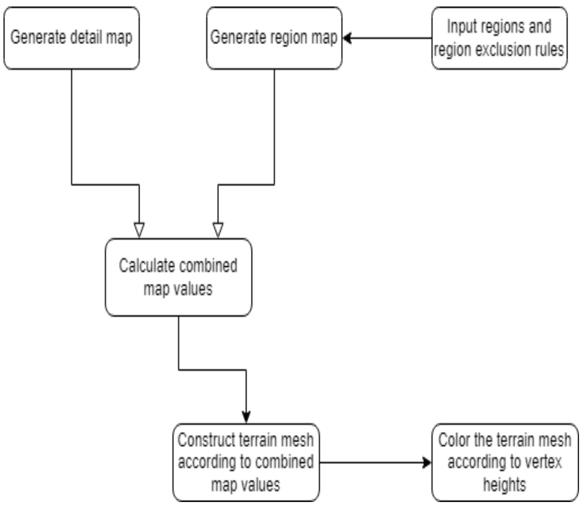
Generate detail map
Generate region map
Input regions and region exclusion rules
Calculate combined map values
Construct terrain mesh according to combined map values
Color the terrain mesh according to vertex heights

Figure 1. The proposed IMGA algorithm steps

“

Algorithm 2: Improved Map Generation Algorithm (IMGA)

Input: Mesh colors, region heights and exclusion rules
Output: Terrain mesh

```
1  Generate base map with noise generation
2  Generate region map with modified WFC based on cluster factor criteria.
3  Procedure CombineMaps()
4  Construct a mesh with the combined map
5  Paint the map according to the point heights
```

Procedure: CombineMaps()

Input: baseMap, regionMap, regionMapRatio
Output: combinedMap

```
1  Interpolate(start, end, t){
2    return (end - start) * t + start
3  }
4  scale ⟵ baseMap.length / regionMap.length
5  for (every point (x, y) in baseMap):
6    point ⟵ baseMap[x,y]
7    point *= 1 - regionMapRatio
8    region ⟵ point.regionCell()
9    if (x MOD scale < scale / 2):
10     if (region.leftNeighbor is none):
11       point += regionMapRatio *  region.value
12     else:
13       i ⟵ 2 / (scale - 1) * (x MOD scale)
14       point += regionMapRatio * Interpolate(region.leftNeighbor.value, region.value, i) / 2
15   else:
16     if (region.rightNeighbor is none):
17       point += regionMapRatio * region.value
18     else:
19       i ⟵ -2 / (scale - 1) * (x MOD scale) + 2
20       point += regionMapRatio * Interpolate(region.rightNeighbor.value, region.value, i) / 2
21   if (y MOD scale < scale / 2):
22     if (region.bottomNeighbor is none):
23       point += regionMapRatio * region.value
24     else:
25       i ⟵ 2 / (scale - 1) * (y MOD scale)
26       point += regionMapRatio * Interpolate(region.bottomNeighbor.value, region.value, i) / 2
27   else:
28     if (region.topRegion is none):
29       point += regionMapRatio * region.value
30     else:
31       i ⟵ -2 / (scale - 1) * (y MOD scale) + 2
32       point += regionMapRatio * Interpolate(region.topNeighbor.value, region.value, i) / 2
33   baseMap[x,y] ⟵ point
33 return baseMap
```

In the initial phase, a pseudo-random generator (C# UnityEngine CoreModule) is used as a seed for the Perlin noise generation algorithm to generate a base noise with dimensions of 240 units in width and length. Subsequently, another three Perlin noise maps are generated, featuring higher frequency and lower amplitude than the maps before them. Following this, these maps are summed together. This way, we generated a fractal noise, which forms the foundational basis for our subsequent procedures.

The second phase consists of generating the region map. Unlike the base map, which has 240 cells with one-unit length each, the region map has 24 cells with ten units in width and height. This alteration allows individual cells within the region map to exert influence over more extensive areas than the base noise map we generated using the Perlin noise approach. In the context of crafting a coastal or island-themed map, three key regions are introduced: mountainous, terrestrial, and aquatic. Corresponding height values of 0.9, 0.5, and 0.1 are assigned to these regions, respectively. A stipulation is imposed to foster the creation of a region map resembling shorelines or islands where no cell of a mountainous region should share a border with a cell of an aquatic region, and vice versa. In addition to the base WFC algorithm, a cluster factor is introduced, significantly influencing the cells' selection. Specifically, when a given cell finds itself encompassed by neighboring cells belonging to the same region, it is inclined to adopt the identity of those neighboring cells' regions. A predefined cluster factor value determines the extent of this influence. By applying this mechanism during the process of region assignment, the resultant map exhibits a tendency towards forming localized clusters of regions. This aspect contributes to the emergence of coherent and concentrated region areas within the generated map.

To combine the noise and region maps, the procedure CombineMaps() is executed. The lines from 5 to 34 will be executed for every point on the map.

First, the point value is squashed to reduce the influence of the detailed map on the final combinedMap values (line 7). Then, the corresponding region for the given point is stored (line 8). This region is used to access the neighboring region values for the interpolation process. In the next step, a section of the region where the given point is located must be determined to determine the region value (lines 9-32). Suppose the index x corresponds to the left part of the region area the interpolation is made with the left region cell value (lines 10-14). In that case, the interpolated value is added on top of the base map value. If the index x corresponds to the right part of the region, the interpolation is made with the right region cell, then the interpolated value is added on top of the base map value (lines 16-20). The current region value is used alone if there is no neighboring cell to interpolate (lines 10-11 and 16-17). At the end of line 20, the interpolation for the x-axis is complete, and the interpolation for the y-axis is calculated in a similar way, but checking for bottom and top instead of left and right (lines 22-32). Finally, the calculated point is updated (line 33).

Figure 2 illustrates these sections for a single region cell. The reason for this separation is that the interpolation formula and the neighboring region cell to be interpolated change depending on which section of the region the point is located in. Initially, the interpolation for the x-axis is calculated (lines 9-20), and following that, the interpolation for the y-axis is calculated (lines 21-32). These procedures are evaluated on top of the base map value when calculated to produce the final value for that given point.

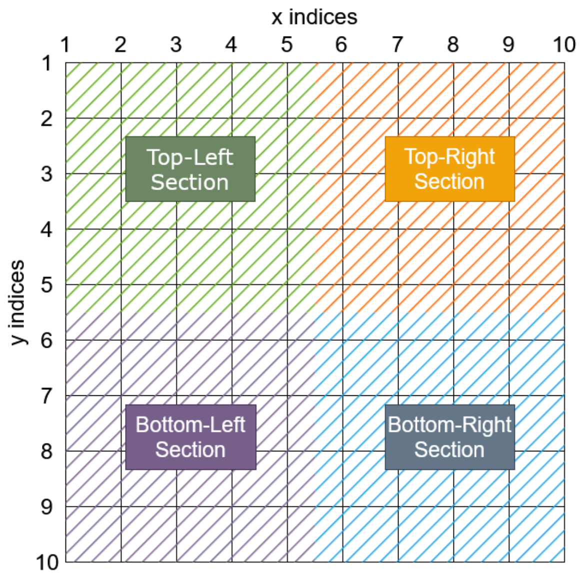

Figure 2. Illustration of four sections for a single region cell

To aid in visual understanding, Figure 3 shows the process of combining the maps for a single axis where the base map is 1000 units in total length and 250 units in length for a single region.

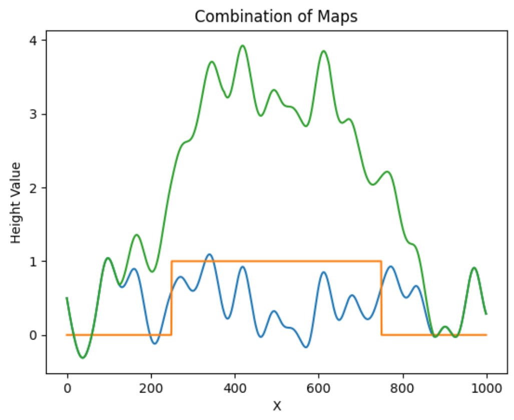

Figure 3. 1D representation of combining base map and region map

Once the computation for each point is completed, a terrain mesh object is generated, mirroring the array's dimensions. A mesh object in this context serves as a three-dimensional depiction of the terrain, encompassing its constituent vertices. The values previously computed and combined in the map are employed as a Digital Elevation Model (DEM) for constructing the terrain. Essentially, these combined map values are utilized as vertex heights and are subsequently imparted to the mesh object. Provided in Figure 4 is an overhead illustration that mirrors the fundamental indexing process for terrain meshes, which commences from the upper left corner and progresses downward and to the right. This diagram utilizes black numerical annotations to represent vertex indices, while red numbers are employed to denote triangle indices.

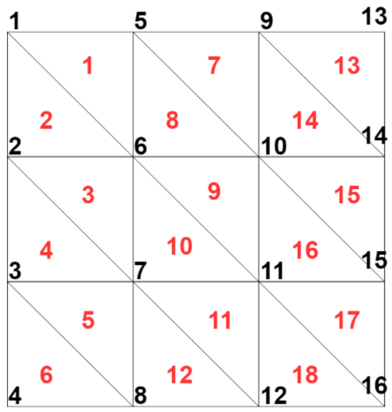

Figure 4. Vertex and triangle indexes for a 3 by 3 terrain mesh

When heights are applied to the vertexes of the mesh by using a DEM, a terrain with the elevation values of the DEM is constructed. A terrain hill constructed by applying a mountain DEM is illustrated in Figure 5.

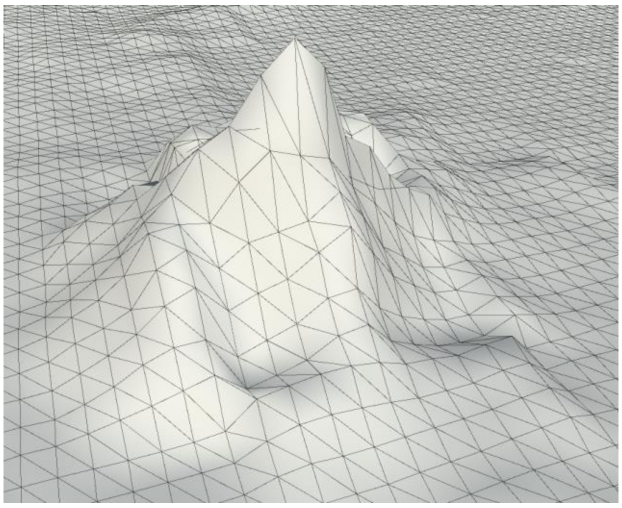

Figure 5. Small mountain constructed by using a DEM

Subsequently, the vertices of the map undergo a process of coloration based on their respective heights. The elevation range of the terrain, extending from its minimum to maximum height, is partitioned into distinct regions, each associated with a unique color. The color assigned to a particular vertex is determined by the region it belongs to.

In alignment with the objective of capturing island and shoreline aesthetics, the chosen color palette comprises blue to signify the sea, yellow to denote shores, green for forests, dark brown representing mountains, and white symbolizing snow. The colored representation of the mountain introduced in Figure 5 is showcased in Figure 6, providing a visual portrayal of the mountain using the designated color scheme.

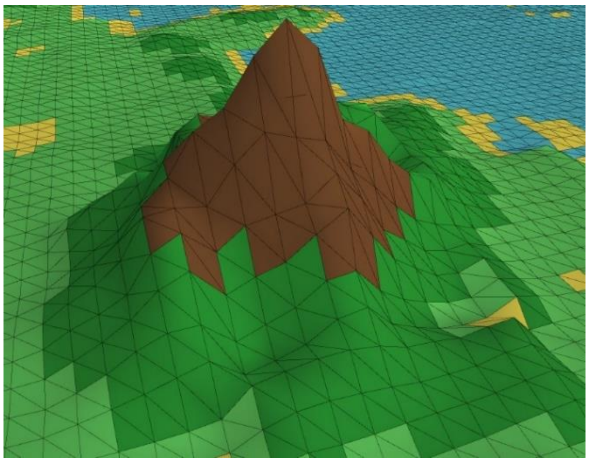

Figure 6. Small mountain painted according to vertex heights

2.3.1 Layout and structure of the map

The map generated by the IMGA process introduces enhanced structure through the implementation of neighbor exclusion rules, a heightened ratio of region map to detail map influence during their combination, and a cluster factor influencing neighboring region selection.

An increased combined ratio signifies that the region map exerts a more pronounced influence on the final map compared to the detailed map. As a result of this heightened influence, the configuration of the map is predominantly shaped by the values from the region map, effectively outlining the map's overall layout.

Conversely, the incorporation of neighbor exclusion rules serves to instill a sense of organization within the constructed regions, imparting a semblance to real-world geographical patterns.

The cluster factor plays a pivotal role in determining region selection based on the prevalence of dominant regions among neighboring vertices. By manipulating this factor, the clustering of similar or identical regions is intensified during the process of region map generation.

3 Simulation results

We integrated the IMGA procedure into the Unity game engine for execution. The map generation process is performed on a laptop with an Intel i7 9750h CPU and 16GB of RAM. The dimensions of the generated maps are set at 240 units in width and 240 units in length. The duration required to generate each map falls within the range of 0.5 to 0.7 seconds.

During the region map generation phase, only one constraint is upheld: the prevention of adjacency between water regions and mountain regions. The selection of the region map to noise map ratio is fixed at 1.4, signifying that region map values wield 1.4 times greater influence compared to the base map values within the final map outcomes. In conjunction with this, a clustering factor of 0.7 is designated, indicating that neighboring regions are 70% more inclined to match.

To enable meaningful comparisons, the map generated using the wave function collapse algorithm employs region height values for the construction of a mesh. This way, the algorithm is executed without the need for pre-designed tiles.

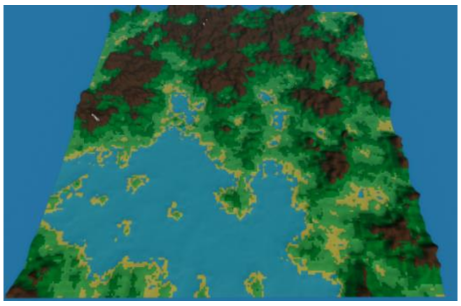
Figure 7. Procedural map generated with IMGA

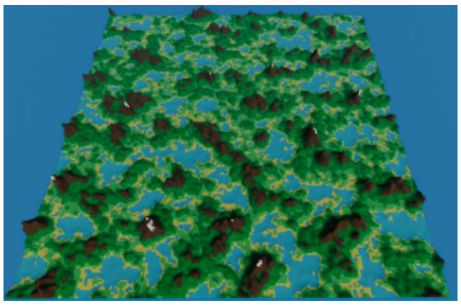
Figure 8. Procedural map generated with Perlin Noise Algorithm

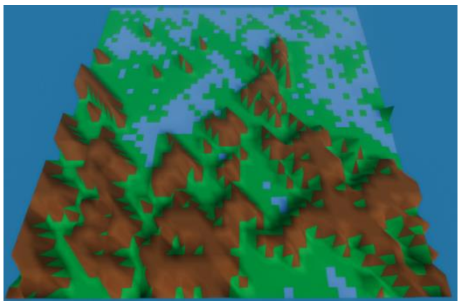
Figure 9. Procedural map generated with region map which is generated by Wave Function Collapse Algorithm

Figures 7, 8, and 9 illustrate insights into maps generated through distinct algorithms. Notably, the map depicted in Figure 7 emerges from the IMGA approach, while the one in Figure 8 is a product of the Perlin Noise method. In contrast, the map showcased in Figure 9 stems from a region map generated by the wave function collapse algorithm. Comparing Figures 7 and 8, it is evident that the proposed IMGA technique yields a map of uneven characteristics in contrast to the Perlin noise-generated map, which presents a more uniform appearance. Furthermore, the map portrayed in Figure 7 exhibits a discernible organizational structure due to the application of rule-based generation, setting it apart from the Perlin noise counterpart. Conversely, the map resulting from the wave function collapse algorithm, as depicted in Figure 9, requires a substantial 2-second interval for generation. This lag in real-time generation efficiency is noteworthy, especially given that the map's size is smaller than a quarter of the maps generated using IMGA and Perlin noise methods.

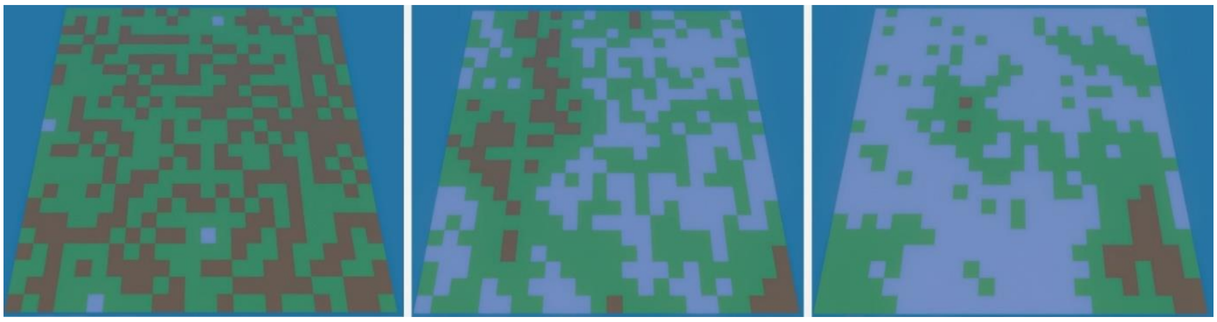
Figure 10 Region maps generated with cluster factors of 0.4, 0.62 and 0.84 from left to right

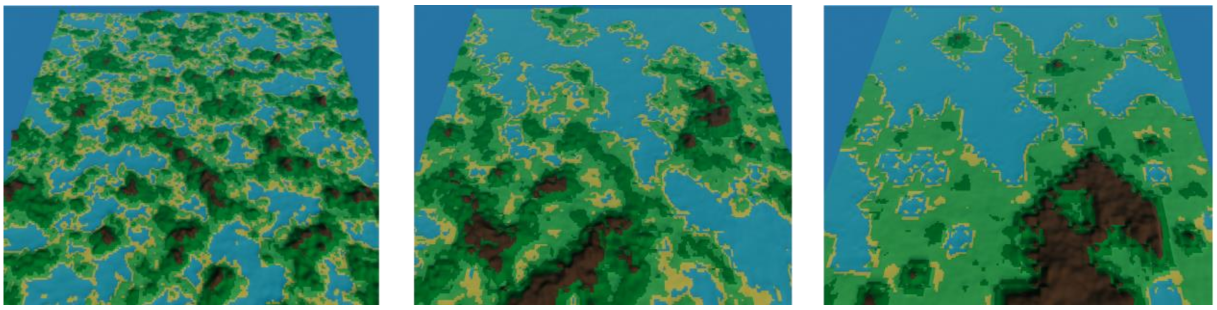
Figure 11. Maps generated with region map / base map ratio of 0.3 , 0.6 and 0.9 from left to right

Displayed in Figure 10 are region maps generated using varying cluster factor values of 0.4, 0.62, and 0.84, progressing from left to right. The cluster factor parameter contributes to map clustering, its impact growing as the value increases. The leftmost image in Figure 10 portrays a map predominantly featuring land and mountain regions. This observation aligns with the constraint that prevents water regions from neighboring mountainous regions, resulting in the generation of maps predominantly characterized by mostly land regions when the cluster factor is set to an extremely low value. Although the low values impact negatively, values starting from 0.6 generate maps that contribute to the overall structure.

Figure 11 illustrates maps generated under different conditions of the region map to base map ratio. This ratio determines the extent to which region map values influence the final map; higher values amplify the impact of region map values. The depicted ratio values in the images are 0.3, 0.6, and 0.9, progressing from left to right. Notably, lower ratio values lead to final maps that exhibit a more remarkable resemblance to maps generated purely by noise algorithms. Conversely, when the ratio becomes excessively high, the resulting map tends to resemble tile-based maps, indicating a reduction in the influence of natural variations present in noise-based maps. It is noted that extremely low or extremely high values yield maps that closely resemble either solely the base map or solely the. In contrast, intermediate values lead to enhanced map quality.

4 Discussions and conclusions

This study introduces an innovative hybrid approach for procedural terrain generation, effectively addressing the limitations inherent in using Perlin noise and wave function collapse algorithms. The proposed method overcomes the shortcomings associated with each of these approaches. Noise generation lacks the structure and characteristics of an interesting map, but its computational cost is low. On the other hand, WFC provides structure in its map through neighbor constraints, but it is computationally high and needs predefined models to work with. Our proposed approach eliminates uniformity by combining the generated map with a modified WFC. It is computationally lower than pure WFC since the algorithm uses the WFC as a region map generator, which has a much smaller cell count than the actual map.

The strengths and weaknesses of this study can be discussed as follows:

- Unlike GANs and erosion simulations, our algorithm operates without the need for training or simulation, and it can be parallelized to achieve near real-time terrain generation.
- In comparison to maps generated solely through Perlin noise, our method produces more diverse maps with structured layouts. Additionally, it eliminates the necessity for manually crafted tiles, offering a significant advantage over the wave function collapse algorithm.
- The techniques and methodologies utilized in this study facilitate the generation of neighboring terrains, a capability lacking in current state-of-the-art map generation using GANs.
- The abstract parameters render the algorithm challenging to configure.

Conflict of interest

The authors declare that there is no conflict of interest.

Similarity rate (iThenticate): 17%

References

[1] R. M. Smelik, K. J. De Kraker, T. Tutenel, R. Bidarra, and S. A. Groenewegen, A survey of procedural methods for terrain modelling. In Proceedings of the CASA Workshop on 3D Advanced Media In Gaming And Simulation (3AMIGAS), pp. 25-34, Amsterdam, The Netherlands, 2009.

[2] 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), 9 (1), 1-22, 2013. https://doi.org/10.1145/2422956.2422957.

[3] K. Perlin, Improving noise. In Proceedings of the 29th annual conference on Computer graphics and interactive techniques, pp. 681-682, San Antonio Texas, USA, 2002.

[4] C. Adams, H. Parekh, and S. J. Louis, Procedural level design using an interactive cellular automata genetic algorithm. In Proceedings of the Genetic and Evolutionary Computation Conference Companion, pp. 85-86, Berlin, Germany, 2017.

[5] R. Zmugg, W. Thaller, U. Krispel, J. Edelsbrunner, S. Havemann, and D. W. Fellner, Procedural architecture using deformation-aware split grammars. The Visual Computer, 30, 1009-1019, 2014. https://doi.org/10.1007/s00371-013-0912-3

[6] De Pontes, R. G., Gomes, H. M., and Seabra, I. S. R., Particle swarm optimization for procedural content generation in an endless platform game. Entertainment Computing, 43, 100496., 2022. https://doi.org/10.1016/j.entcom.2022.100496

[7] Volz, V., Naujoks, B., Kerschke, P., and Tušar, T., Tools for landscape analysis of optimisation problems in procedural content generation for games. Applied Soft Computing, 136, 110121., 2023. https://doi.org/10.1016/j.asoc.2023.110121

[8] C. Beckham, and C. Pal, A step towards procedural terrain generation with gans. arXiv preprint, arXiv:1707.03383, 2017. https://doi.org/10.48550/arXiv.1707.03383

[9] T. J. Rose, and A. G. Bakaoukas, Algorithms and approaches for procedural terrain generation-a brief review of current techniques. In 2016 8th International Conference on Games and Virtual Worlds for Serious Applications (VS-GAMES). pp. 1-2, Barcelona, Spain, 2016.

“

[10] F. Gürler and E. Onbaşioğlu, Applying Perlin noise on 3D hexagonal tiled maps. In 2022 International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT), pp. 670-673, Ankara, Turkey, 2022.
[11] R. Fischer, P. Dittmann, R. Weller and G. Zachmann, AutoBiomes: procedural generation of multi-biome landscapes. The Visual Computer, 36, 2263-2272, 2020. https://doi.org/10.1007/s00371-020-01920-7
[12] A. Petrovas and R. Bausys, Procedural video game scene generation by genetic and neutrosophic WASPAS algorithms.. Applied Sciences, 12(2), 772, 2022. https://doi.org/10.3390/app12020772
[13] E. Panagiotou and E. Charou, Procedural 3D terrain generation using Generative Adversarial Networks. arXiv preprint arXiv:2010.06411, 2020. https://doi.org/10.48550/arXiv.2010.06411
[14] A. Wulff-Jensen, N. N. Rant, T. N. Møller and J. A. Billeskov, Deep convolutional generative adversarial network for procedural 3D landscape generation based on DEM. In Interactivity, Game Creation, Design, Learning, and Innovation: 6th International Conference, ArtsIT 2017, and Second International Conference, pp. 85-94, Heraklion, Crete, Greece, 2017.
[15] J. Olsen, Realtime procedural terrain generation. 2004.
[16] G. C. Backes, T. A. Engel, and C. T. Pozzer, Real-Time Massive Terrain Generation using Procedural Erosion on the GPU. In Proceedings of 17th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames), pp. 675-678, Foz do Iguaçu, Brazil, 2018.
[17] H. Zhang, D. Qu, Y. Hou, F. Gao and F. Huang, Synthetic modeling method for large scale terrain based on hydrology. IEEE Access, 4, 6238-6249, 2016. 10.1109/ACCESS.2016.2612700
[18] J. D. Génevaux, É. Galin, E. Guérin, A. Peytavie and B. Benes, Terrain generation using procedural models based on hydrology. ACM Transactions on Graphics (TOG), 32(4), 1-13, 2013. https://doi.org/10.1145/2461912.2461996
[19] J. Doran and I. Parberry, Controlled procedural terrain generation using software agents. IEEE Transactions on Computational Intelligence and AI in Games, 2(2), 111-119, 2010. 10.1109/TCIAIG.2010.2049020
[20] T. S. L. Langendam and R. Bidarra, miWFC-Designer empowerment through mixed-initiative Wave Function Collapse. In Proceedings of the 17th International Conference on the Foundations of Digital Games, pp. 1-8, Athens, Greece, 2022.
[21] Q. E. Morris, Modifying Wave function collapse for more complex use in game generation and design. Computer Science Honors Theses. USA, 2021.
[22] S. Alaka and R. Bidarra, Hierarchical Semantic Wave function collapse. In Proceedings of the 18th International Conference on the Foundations of Digital Games, pp. 1-10, Lisbon, Portugal, 2023.
[23] Wu, Z., Mao, Y., and Li, Q., Procedural game map generation using multi-leveled cellular automata by machine learning. In Proceedings of the 2nd International Symposium on Artificial Intelligence for Medicine Sciences,pp. 168-172. https://doi.org/10.1145/3500931.3500962
[24] K. PERLIN, An image synthesizer. ACM Siggraph Computer Graphics, 1985, 19 (3), 287-296. https://doi.org/10.1145/325165.325247
[25] Gumin, M. 2016. Wave Function Collapse Algorithm (Version 1.0) [Computer software]. https://github.com/mxgmn/WaveFunctionCollapse, Accessed: 15 September 2023