Proc. of the International Conference on Electrical, Computer and Energy Technologies (ICECET 2022)
20-22 June 2022, Prague-Czech Republic

Regionally morphing objects for the genetic WASPAS-SVNS game scene generation algorithm

2022 International Conference on Electrical, Computer and Energy Technologies (ICECET) | 978-1-6654-7087-2/22/$31.00 ©2022 EU | DOI: 10.1109/ICECET55527.2022.9872949

Aurimas Petrovas
Faculty of Fundamental Sciences
Vilnius Gediminas Technical University
Vilnius, Lithuania
aurimas.petrovas@vilniustech.lt

Romualdas Bausys
Faculty of Fundamental Sciences
Vilnius Gediminas Technical University
Vilnius, Lithuania
romualdas.bausys@vilniustech.lt

Abstract—**Creative content assistance tools are gaining popularity as they allow one to reduce the time needed to create content in games and other media. Procedural generation allows us to create levels autonomously. The main objective of computational creativity is to replicate results, which are usually created by humans. Our tool is an extension of the genetic WASPAS-SVNS game scene generator. It adds variety to a functional game object type by adding an array of possible visual variations. The morphs of the objects are selected regionally and randomly based on a randomly selected point in the game scene. Results are generated for the 'wall' and 'empty space' object types by adding grass and tree objects to the initial object type. It adds creative value to the level of the game without breaking the rules of game design.**

Keywords—procedural generation, game scene, WASPAS-SVNS, regional object morph

I. Introduction

Procedural generation in game development is a popular tool for generating content in games. It allows to generate a lot of game assets in a short amount of time, considering that the generation tool is already created. In comparison, human-crafted game levels usually are more unique, but at the same time may lack randomness or natural patterns, which can be observed in nature. Procedurally generated levels can also include reactivity to player actions and game content personalization [1]. Procedural generation can be defined in various ways. Some authors define it as automatically generated assets with limited input data or amplification algorithms. Others define it as a fully automatic generation algorithm with no input data. The emphasis for automatic generation is on the word – minimal input [2]. In the proposed research, we try to minimize input data and generate unique layouts for game levels.

In recent years, research on autonomous game-level layout generation systems is gaining popularity and different methods are being proposed that are usually combined with evolutionary algorithms [3-7]. A goal of these algorithms is to add creativity traits with minimal or no pre-generated data. These methods usually operate on combination of game functionality requirements and aesthetic rules application for the generated levels. The games prototypes used in these games are usually minimalistic and use one game object type for one game function. The proposed algorithm is the expansion of one of these algorithms (Genetic WASPAS-SVNS (weighted aggregated sum product assessment in a single-valued neutrosophic set environment)) used for game layout generation [7]. These types of algorithms are usually limited by the size of the criterion array. When the ratio between the game scene grid size and amount of conflicting criteria reaches a certain point, it becomes difficult to distinguish features of these criterion.

One of the game levels defining values is object variety and patterns in which they are combined. There can be many different level design rules that can be used to generate patterns for game levels. The first stage to generate a layout is to have a relative pattern between objects considering the function of each object type, the next stage is to add variety to them by adding alternative looks for a specific game object type, which will increase the aesthetic value of the game level [8-10]. Proposed research is focused on the next layer of game scene generation, where the functional level is expanded with regionally morphed zones where arrays of alternative visual objects are used for the specific function of game objects.

The original algorithm [7] uses genetic algorithm to generate layouts of the game scene. The fitness function is composed of an array of functional and aesthetic game design criterion combined with multicriteria decision making algorithm using neutrosophic sets. The WASPAS algorithm is used to combine conflicting criteria and neutrosophic sets that increase the unpredictability and randomness of the algorithm. The base algorithm for a single iteration is usually used for a variety of engineering and design problems. Some of them include architectural, construction, environmental sustainability, image processing, or path-finding tasks [11-14].

The goal of this research is an increase in the aesthetic quality of the generated levels by adding another level of algorithms, which will add regionally morphing areas from the array of alternative visual objects. This algorithm is the next step towards the problem of automated game design with minimal designer supervision by increasing automated creative value.

One of the defining key factors of these algorithms is the lack of a starting seed or pattern, which is quite often used in procedural generation in game level design [15-17].

II. Genetic WASPAS-SVNS Game Scene Generator

The base for the proposed algorithm is the WASPAS-SVNS game scene layout generator. The original algorithm contains seven types of game objects as building blocks to generate layouts. These objects are: player, exit, empty space, wall, hazard, collectible, and ground. Each of these objects represents a different game function. Original criteria for the fitness function are symmetry, empty space balance, distance between objects, safe zone, player existence, exit existence, and a path between player and exit. We build on top of existing objects by expanding their visual variety and adding a new layer of post-processing algorithm on the generated level.

The base algorithm initializes a random population of generated level layouts and then initiates an evolutionary algorithm. Then it repeats the WASPAS-SVNS iteration for each evolutionary cycle and each chromosome. Individual criterion are calculated starting from the constraining ones (related to game design rules) and then applying the ones, which should be maximized (related to aesthetic rules). After that, the algorithm looks for performers and underperformers of the population. The best chromosomes are evolved. The final step of the algorithm is the level layout map decoder and visualizer [7]. Our research improves the decoder part with regionally morphing visual objects.

III. Regional Object Morph

The proposed algorithm modifies the parts 'Draw' and 'Decode' of the original algorithm and adds a regional morpher to it. The Algorithm iterates over the matrix of generated objects and then assigns an object to spawn in the game scene. Each chromosome also gets a random corner point in the matrix, which radiates changes towards other areas of the level (Fig. 1).

The next modified algorithm part selects a game object from the grid and assigns a model to it, based on the chromosome number for each cell of the grid. Visual object selection is replaced by an object morph algorithm. For this experiment, the empty space and wall objects are used to test the algorithm (Fig. 2). Each identification number represents an object type. Arrays of objects visual array can be modified and 3 objects for each game object type were used to test it. We also tried to vary the resolution of the object grid to see how it affects the final outcome.

The Morphing algorithm selects a visual object from the pool of available objects and then calculates a relative position to the selected corner point. The relative position is then multiplied by a set of two random numbers between 0 - 1 and 0.5 – 1 to add more noise to the equation. The negative axis sides of the grid use multipliers between 0 – 1 and 1 – 1.5 (Fig. 3).

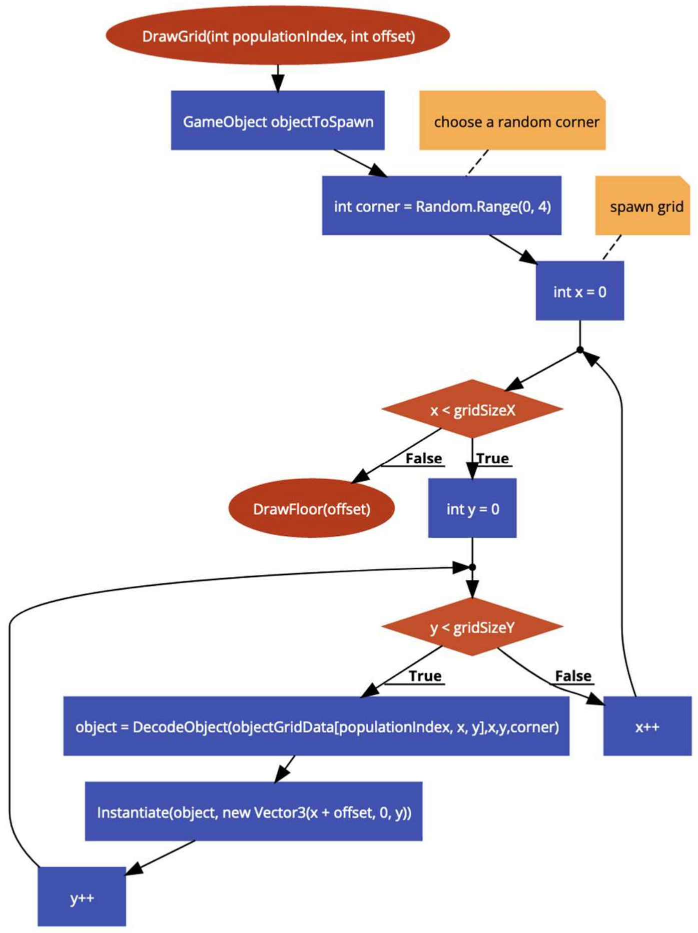
DrawGrid(int populationIndex, int offset)
GameObject objectToSpawn
choose a random corner
int corner = Random.Range(0, 4)
spawn grid
int x = 0
x < gridSizeX
False
True
DrawFloor(offset)
int y = 0
y < gridSizeY
True
False
object = DecodeObject(objectGridData[populationIndex, x, y],x,y,corner)
x++
Instantiate(object, new Vector3(x + offset, 0, y))
y++

Fig. 1. Iteration over the game objects grid.

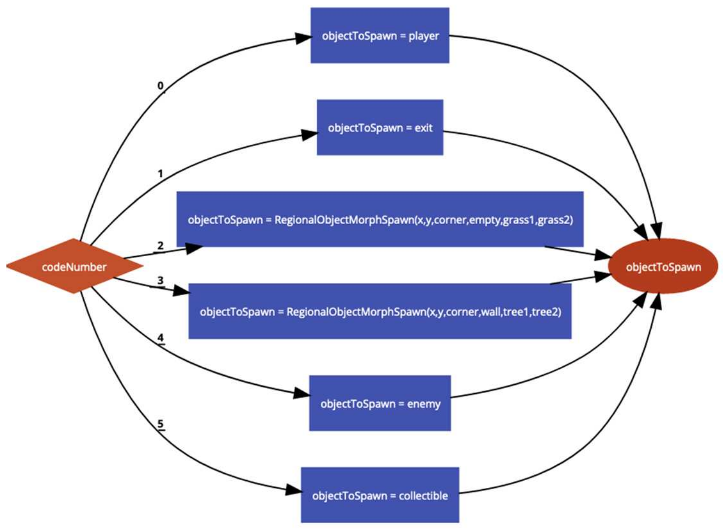
codeNumber
0
1
2
3
4
5
objectToSpawn = player
objectToSpawn = exit
objectToSpawn = RegionalObjectMorphSpawn(x,y,corner,empty,grass1,grass2)
objectToSpawn = RegionalObjectMorphSpawn(x,y,corner,wall,tree1,tree2)
objectToSpawn = enemy
objectToSpawn = collectible
objectToSpawn

Fig. 2. Inclusion of object morpher into a decoder.

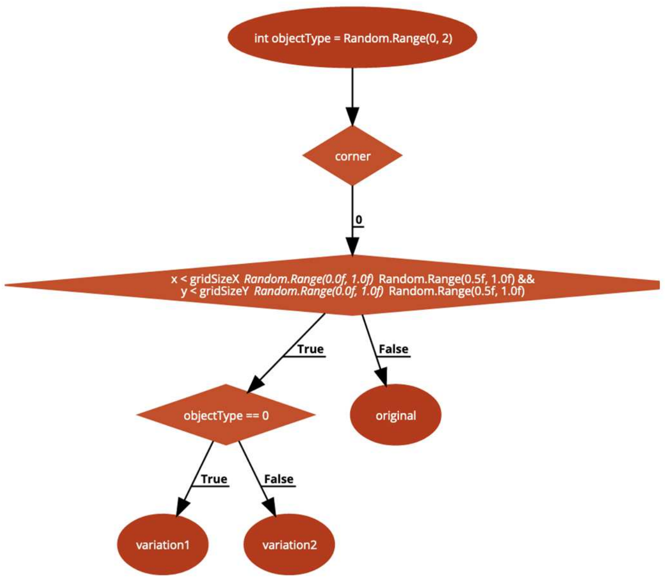
int objectType = Random.Range(0, 2)
corner
0
x < gridSizeX Random.Range(0.0f, 1.0f) Random.Range(0.5f, 1.0f) &&
y < gridSizeY Random.Range(0.0f, 1.0f) Random.Range(0.5f, 1.0f)
True
False
objectType == 0
original
True
False
variation1
variation2

Fig. 3. Visual object selection and noise function.

The equation to decide if the object should morph consists of two lines for each axis which compares object coordinates with grid length of one axis (Ax, Ay), random number (R) and noise (N). There are 4 checks with each possible comparison (*x1>x2, y1>y2; x1<x2, y1<y2, x1>x2, y1<y2, x1>x2, y1<y2* (*x1, y1* – object coordinates; *x2, y2* – randomness element (1))).

$$\begin{cases} Ax * R * N \\ Ay * R * N \end{cases} \tag{1}$$

With this approach, we can still keep all the original algorithm rules intact while expanding the visual diversity aspect of it. The same algorithm is repeated for each type of object.

IV. Results

The visual results generated with this algorithm add variety to the final layout of the game scene by adding more natural randomness to the visual side of the level. Some of the iterations generate regional grassy fields, and some of them, trees, which are aligned with the other impassable walls. In this manner, our physical-level boundaries and functionality are not impacted. Some examples of the final results are: a grassy region in the corner and some trees (Fig. 4), example with increased resolution (Fig. 5), clustered trees (Fig. 6), rocks instead of trees (Fig. 7). Increasing the size of the grid for the algorithm increases the time required to generate a game scene exponentially. Added visual object types have to be aligned with their purpose for the algorithm to place them logically. The final algorithm takes 6.3 seconds over 800 epochs to generate a level on a 2.4 GHz 8-Core Intel Core i9 CPU. The algorithm reaches around 0.7 fitness value after 100 epochs and then around 0.85 after 800 epochs. Note that it cannot go close to 1 due to conflicting criterion nature of the algorithm and the goal of varied results.

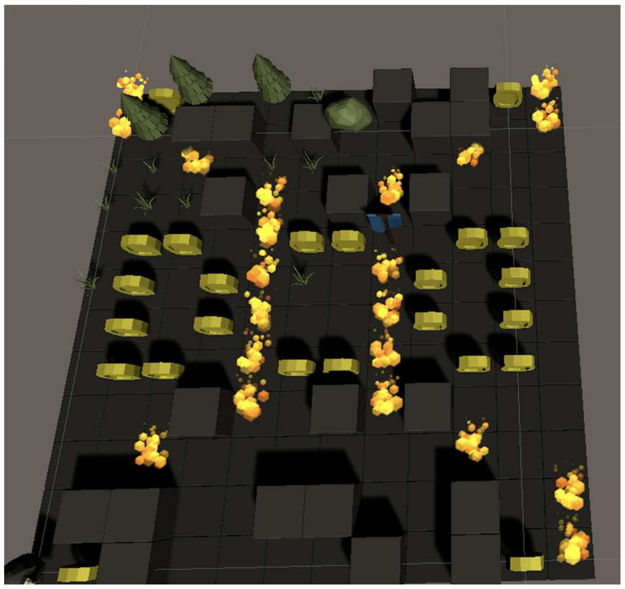

Fig. 4. Regional morph results.

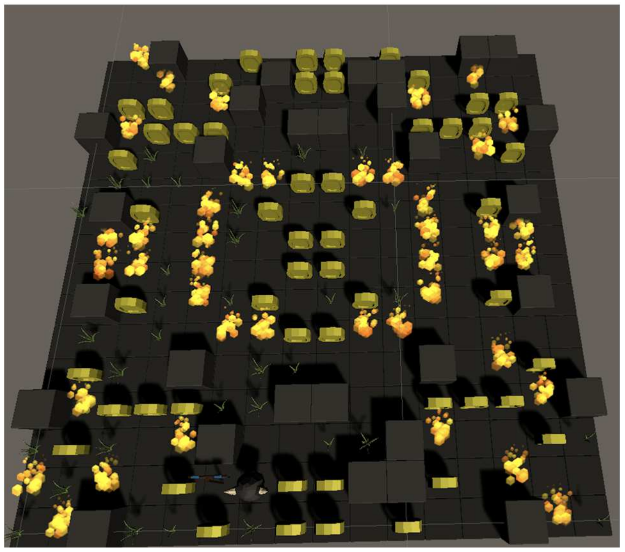

Fig. 5. Example with increased resolution.

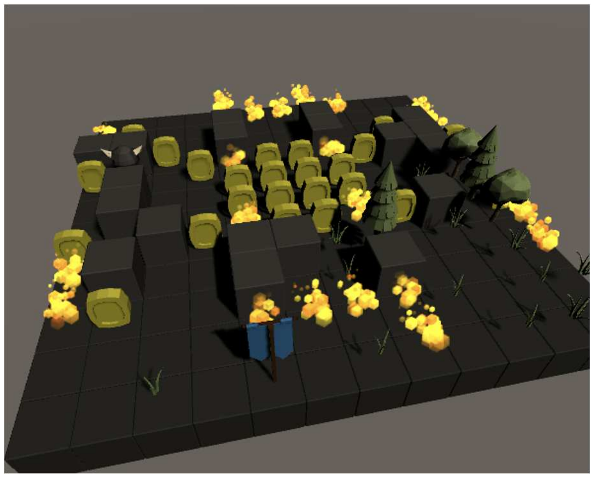

Fig. 6. Clusters of objects at the side of the level

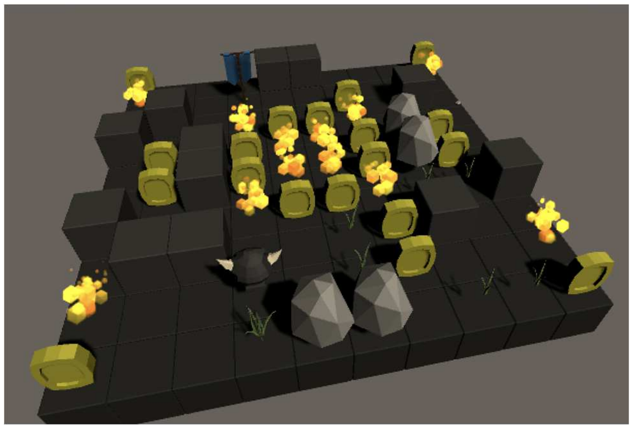

Fig. 7. Wall morph to rocks instead of trees.

V. Conclusion

Creative procedural generation without premade content snippets is an emerging branch of Computational Creativity. In this paper, we proposed and tested extension for the genetic WASPAS-SVNS gene scene generation algorithm by adding regionally morphing post-processing for the game objects. This extension adds variety to the visual game-level diversity and increases creative value without breaking game design rules. Each iteration of the algorithm can generate new patterns of the game level integrated into the composition of the game level. The modularity of the original procedural generation engine gives a lot of room to integrate new extensions for the total creative value of the generated level.

References

[1] T. Short , T. Adams, "Procedural generation in game design. CRC Press," 2017.

[2] J. Freiknecht, W. Effelsberg, "A survey on the procedural generation of virtual worlds," Multimodal Technologies and Interaction, December 2017.

[3] S. Thakkar, C. Cao, L. Wang, T.J. Choi, J. Togelius, "Autoencoder and evolutionary algorithm for level generation in lode runner," 2019 IEEE Conference on Games (CoG) 2019 Aug 20 (pp. 1-4), IEEE.

[4] A. Zafar, H. Mujtaba, M. O. Beg, "Search-based procedural content generation for GVG-LG," Applied Soft Computing, 2020 Jan 1;86:105909.

[5] V. Volz, N. Justesen, S. Snodgrass, S. Asadi, S. Purmonen, C. Holmgård, J. Togelius, S. Risi, "Capturing local and global patterns in procedural content generation via machine learning," 2020 IEEE Conference on Games (CoG) 2020 Aug 24 (pp. 399-406), IEEE.

[6] A. B. Safak, E. Bostanci, A. E. Soylucicek, "Automated maze generation for Ms. Pac-Man using genetic algorithms," International Journal of Machine Learning and Computing, 2016 Aug;6(4):226-40.

[7] A. Petrovas, R. Bausys, "Procedural Video Game Scene Generation by Genetic and Neutrosophic WASPAS Algorithms," Applied Sciences. 2022 Jan;12(2):772.

[8] A. Alvarez, S. Dahlskog, J. Font, J. Holmberg, S. Johansson, "Assessing aesthetic criteria in the evolutionary dungeon designer," Proceedings of the 13th International Conference on the Foundations of Digital Games 2018 Aug 7 (pp. 1-4).

[9] J. Schrum, J. Gutierrez, V. Volz, J. Liu, S. Lucas, S. Risi, "Interactive evolution and exploration within latent level-design space of generative adversarial networks," InProceedings of the 2020 Genetic and Evolutionary Computation Conference 2020 Jun 25 (pp. 148-156).

[10] P. Atkinson, F. Parsayi, "Video games and aesthetic contemplation," Games and culture, 2021 Jul;16(5):519-37.

[11] I. Lescauskiene, R. Bausys, E. K. Zavadskas, B. Juodagalviene, "VASMA weighting: Survey-based criteria weighting methodology that combines ENTROPY and WASPAS-SVNS to reflect the psychometric features of the VAS scales," Symmetry, 2020 Oct;12(10):1641.

[12] Z. Morkunaite, R. Bausys, E. K. Zavadskas, "Contractor selection for sgraffito decoration of cultural heritage buildings using the WASPAS-SVNS method," Sustainability, 2019 Jan;11(22):6444.

[13] R. Bausys, B. Juodagalvienė, R. Žiūrienė, I. Pankrašovaitė, J. Kamarauskas, A. Usovaitė, D. Gaižauskas, "The residence plot selection model for family house in Vilnius by neutrosophic WASPAS method," Infinite Study, 2020 May 1.

[14] R. Bausys, G. Kazakeviciute-Januskeviciene, "Qualitative rating of lossy compression for aerial imagery by neutrosophic waspas method," Symmetry, 2021 Feb;13(2):273.

[15] M. Nenad, "Designing game worlds. Coherence in the design of open world games through procedural generation techniques," Proceedings of the 2018 Annual Symposium on Computer-Human Interaction in Play Companion Extended Abstracts 2018 Oct 23 (pp. 353-363).

[16] A. Khalifa, M. C. Green, G. Barros, J. Togelius, "Intentional computational level design," Proceedings of The Genetic and Evolutionary Computation Conference 2019 Jul 13 (pp. 796-803).

[17] A. Khalifa, F. de Mesentier Silva, J. Togelius, "Level design patterns in 2D games," 2019 IEEE Conference on Games (CoG) 2019 Aug 20 (pp. 1-8). IEEE.