International Science Group

ISG-KONF.COM

INTEGRATION OF SCIENTIFIC BASES INTO PRACTICE

IV

SCIENTIFIC AND PRACTICAL CONFERENCE

12-16 October

Stockholm, Sweden

DOI 10.46299/ISG.2020.IV

ISBN 978-1-64945-864-3

INTEGRATION OF SCIENTIFIC BASES INTO PRACTICE

Abstracts of IV International Scientific and Practical Conference

Stockholm, Sweden
October 12-16, 2020

Library of Congress Cataloging-in-Publication Data

UDC 01.1
The IV th International scientific and practical conference «Integration of scientific bases into practice» (October 12-16, 2020). Stockholm, Sweden 2020. 523 p.

ISBN - 978-1-64945-864-3
DOI - 10.46299/ISG.2020.IV

EDITORIAL BOARD

APPLICATION OF PROCEDURAL GENERATION OF GAME CONTENT USING SOFTWARE ALGORITHMS

Tvoroshenko Iryna,
Ph.D., Associate Professor
Kharkiv National University of Radio Electronics

Almakaieva Anastasiia
Master in informatics
Kharkiv National University of Radio Electronics

Procedural generation (PCG) is automatic creation of game content using algorithms [1].

PCG is software that can create game content on its own or in collaboration with players or a game designer.

Content means the creation of game levels, maps of the game world, game rules, textures, plots, objects, quests, music, weapons, vehicles, characters, and more [2-6].

This paper discusses computer video games directly. The key feature of the created content is that it must be playable – the player must be able to pass the created level, use the generated weapons, and climb the generated stairs, and so on.

The terms “procedural” and “generation” refer to computers. The PCG must be run as a procedure on a computer that will output the result for later use.

The created content must meet certain conditions [7] and solve the relevant problems.

In practice, the following properties are most often considered [8-10]:

– Speed: depending on the task, the requirements vary from milliseconds to months, but in the general case, the content must be created in time to meet the needs of the gameplay;

– Reliability: Some generators create a large number of different objects and events, while others can guarantee that the specified criteria are met. For example, always provide the possibility of passage of the player to the exit of the maze;

– Controllability: the ability to control the generated content based on the situation and provides the game designer with the appropriate command. For example, generating a smooth oblong stone or creating a level with a certain atmosphere;

– Diversity: creating content that is disparate at different launches. This is especially true for games with a small amount of internal code when you want to create levels that are as different as possible from the outside. The mechanics and structure will be the same everywhere, but the player will not be bored with the gameplay;

– Creativity and plausibility: the generation of content that looks as if it was created by man, not a generator. This is one of the most difficult tasks of procedural generation. The highest level of skill is my specialist, whose code is capable of: to

recreate storytelling, the story of the character, the logical sequence of events, an interesting combination of obstacles, and non-game characters.

The procedural generation can calculate various functions [11].

Fractals are geometric patterns that can often be formed procedurally. Typical procedural content includes textures and grids. Sound is also often generated procedurally and is used in both speech synthesis and music. It was used to create compositions in various genres of electronic music by artists such as Brian Eno, who popularized the term “generative music”.

Software developers have been using procedural generation methods for years [12, 13], but this approach is widely used in a relatively small number of products.

Procedural elements have appeared in previous video games:

– The Elder Scrolls II: Daggerfall takes place in a largely procedurally formed world, giving the world about two-thirds of the actual size of the British Isles;

– Raven Software's Soldier of Fortune uses simple procedures to detail enemy models, while its sequel included a randomly generated level mode. Avalanche Studios used generations of procedures to create a large and diverse group of detailed tropical islands for Just Cause;

– The game No Man's Sky, developed by Hello Games, is based on procedurally generated elements.

The main problem of procedural generation in video games is described by the writer Kate Compton and satirically called “procedural oatmeal”. With the automatic generation of large amounts of material, procedural systems can create an infinite number of worlds to study. Without sufficient human management and rules, the result becomes meaningless [8]. It is possible to mathematically form thousands of “bowls of oatmeal” with procedural generation, the user will perceive them as the same, and they will lack true natural uniqueness [14].

Modern ephemerality of fashion dictates to developers the conditions: to invent unique plots, to reduce the size of applications, to create bigger and more interesting universes. The most important thing is to do everything together as quickly as possible [6].

The productivity of the team directly affects the success of the product on the market, because the game, which was presented after a similar or better at least a day, will be almost no one is interested. Such is the harsh conditions of the current market. The scenery generator comes to the aid of creating a game universe.

The scenery generator is software used to create landscape images, 3D models, and animations [15]. If procedural generation were not used to create the environment, then the 3D artist would create landscapes in the form of 3D models.

Then they are textured, materials are adjusted, baked in various programs (for example, Substance Painter), from a high-polygonal heavy model the so-called “lowpole” is made, which will look detailed due to baked textures. Thus, it is a difficult and painstaking process that requires a large amount of knowledge of various programs and experiences from a specialist [16, 17]. Instead, scenery generators are often used in video games or movies. The main elements of landscapes created by scenery generators include terrain, water, leaves, and clouds.

In the process of basic random generation, algorithms are used that analyze the texture on the earth model. On it, the generator distributes landscape objects [15].

Some games require maze generators. They can be implemented to create mazes, dungeons, house plans, and more.

There are the following methods for this:

– BSP-trees can be used to create the simplest mazes and the most typical for “roguelike” cards – rectangular rooms connected to each other by corridors (Fig. 1);

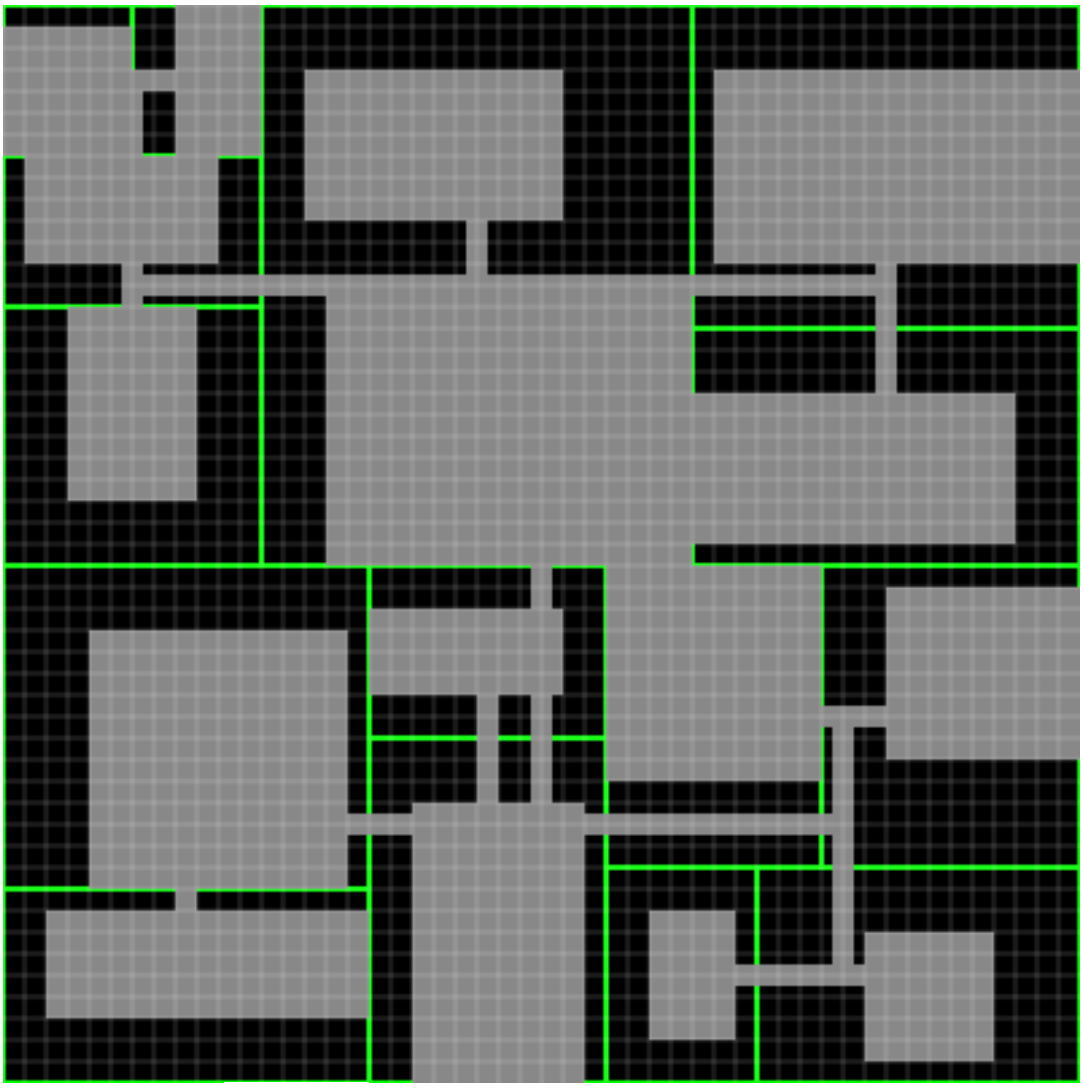

Figure 1. Example of BSP tree generation.

– Tunneling algorithms dig corridors and rooms in solid “earth”, almost in the same way as a real dungeon architect would do. When using such algorithms, unnecessary or redundant paths are often obtained (Fig. 2);

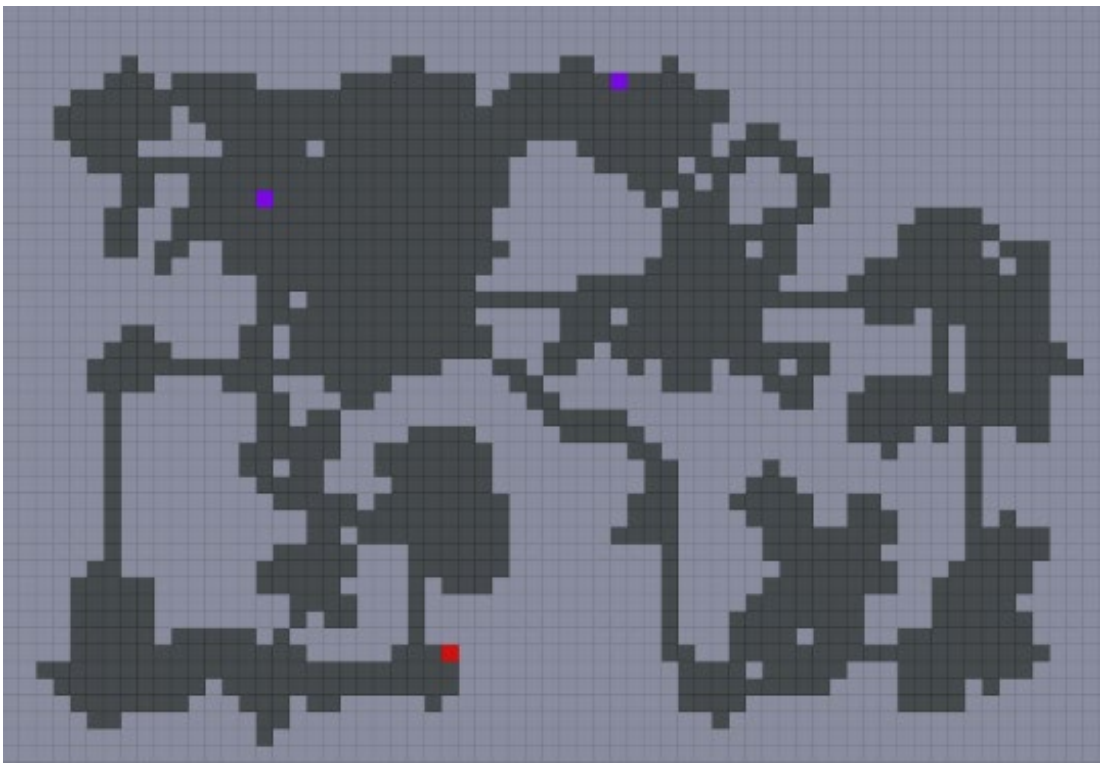

Figure 2. Example of tunneling algorithm generation.

– The tunneling algorithm with a high degree of randomization (drunkard's walk) is useful when creating cave-like maps with a mixture of open and closed spaces (Fig. 3);

Figure 3. Example of Drunkard's Walk generation.

– Cage machines are great for digging cave systems that should look natural. Unlike other methods in this developer, after generating the map, you need to provide the connection yourself, because some algorithms are more likely to create divided areas (Fig. 4).

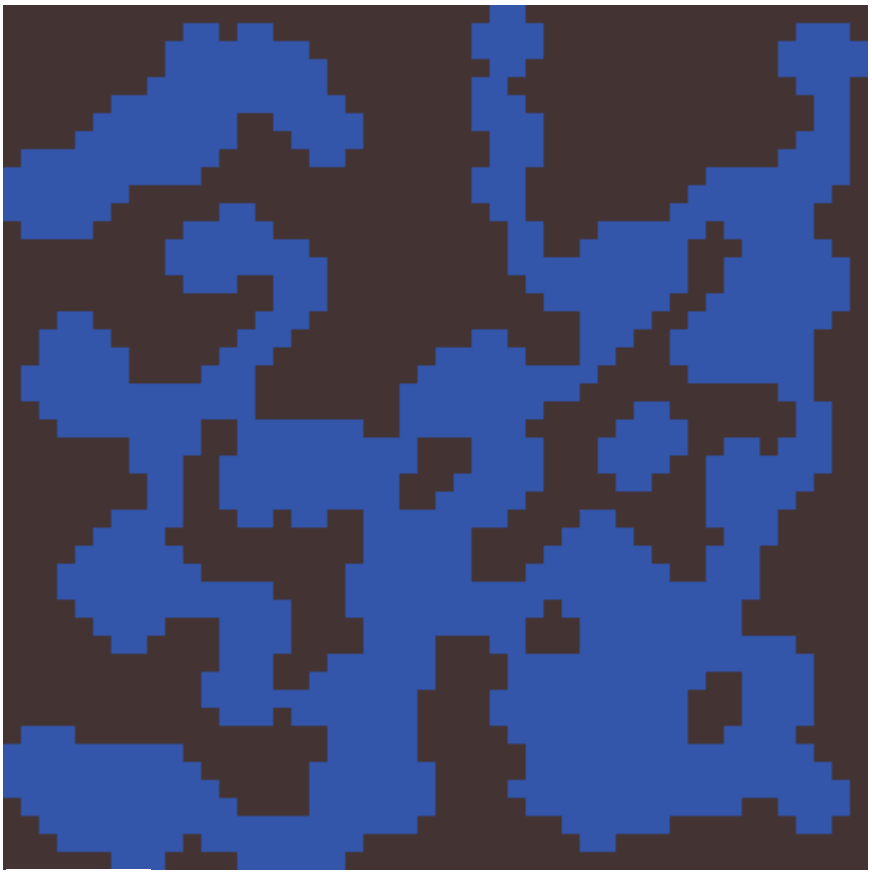

Figure 4. An example of generating a cellular automaton.

The content of the video game can be created both manually and using an algorithm. The algorithm is designed so that it can create the numerical content given above in the examples. Pseudo-randomness must be added to make the machine look unique, interesting, and natural. Both methods have their advantages and disadvantages. As mentioned earlier, video game development takes a huge number of hours. Every content must justify its existence in order to turn it into a game.

References:

1. Sonka M., Hlavac V., and Boyle R. (2014) *Image Processing, Analysis, and Machine Vision*, Atlanta, USA: Thomson-Engineering, 920 p.

2. Peters J.F. (2017) *Foundations of computer vision: Computational Geometry, Visual Image Structures and Object Shape Detection*, Cham, Switzerland: Springer International Publisher, 417 p.

3. Duda R.O., Hart P.E., and Stork D.G. (2000) *Pattern classification*, Hoboken, USA: John Wiley & Sons, 738 p.

4. Tvoroshenko I.S., and Gorokhovatsky V.O. (2019) Intelligent classification of biophysical system states using fuzzy interval logic, *Telecommunications and Radio Engineering*, 78(14), pp. 1303-1315. DOI: 10.1615/TelecomRadEng.v78.i14.80.

5. Nong Ye. (2013) *Data Mining: Theories, Algorithms, and Examples*, Florida, USA: CRC Press, 349 p.

6. Tvoroshenko I.S., and Gorokhovatsky V.O. (2019) Modification of the branch and bound method to determine the extremes of membership functions in fuzzy intelligent systems, *Telecommunications and Radio Engineering*, 78(20), pp. 1857-1868. DOI:10.1615/TelecomRadEng.v78.i20.80.

7. M. Ayaz Ahmad, Irina Tvoroshenko, Jalal Hasan Baker, and Vyacheslav Lyashenko (2019) Modeling the Structure of Intellectual Means of Decision-Making Using a System-Oriented NFO Approach, *International Journal of Emerging Trends in Engineering Research*, 7(11), pp. 460-465. DOI: 10.30534/ijeter/2019/107112019

8. Flah P. (2015) *Machine learning. The science and art of building algorithms that extract knowledge from data*, Moscow, Russia: DMK Press, 400 p., (in Russian).

9. Lyashenko V., Mustafa S.K., Tvoroshenko I., and Ahmad M.A. (2020) Methods of Using Fuzzy Interval Logic During Processing of Space States of Complex Biophysical Objects, *International Journal of Emerging Trends in Engineering Research*, 8(2), pp. 372-377. DOI: 10.30534/ijeter/2020/22822020

10. Daradkeh Y.I., and Tvoroshenko I. (2020) Technologies for Making Reliable Decisions on a Variety of Effective Factors using Fuzzy Logic, *International Journal of Advanced Computer Science and Applications*, 11(5), pp. 43-50. DOI: 10.14569/IJACSA.2020.0110507

11. Tvoroshenko I.S., and Gorokhovatsky V.O. (2020) Effective tuning of membership function parameters in fuzzy systems based on multi-valued interval logic, *Telecommunications and Radio Engineering*, 79(2), pp. 149-163. DOI: 10.1615/TelecomRadEng.v79.i2.70.

12. Yousef Ibrahim Daradkeh, and Iryna Tvoroshenko (2020) Application of an Improved Formal Model of the Hybrid Development of Ontologies in Complex Information Systems, *Applied Sciences*, 10(19). p. 6777. DOI: 10.3390/app10196777

13. Ahmad M. Ayaz, Tvoroshenko Irina, Baker Jalal Hasan, and Lyashenko Vyacheslav (2019) Computational Complexity of the Accessory Function Setting Mechanism in Fuzzy Intellectual Systems, *International Journal of Advanced Trends in Computer Science and Engineering*, 8(5), pp. 2370-2377. DOI: 10.30534/ijatcse/2019/77852019

14. Szeliski R. (2010) *Computer Vision: Algorithms and Applications*, London, Great Britain: Springer-Verlag, 957 p.

15. Tvoroshenko I.S., and Kramarenko O.O. (2019) Software determination of the optimal route by geoinformation technologies, *Radio Electronics Computer Science Control*, 3, pp. 131-142. DOI: 10.15588/1607-3274-2019-3-15.

16. Asaad Ma. Babker, Abd Elgadir A. Altoum, Irina Tvoroshenko, and Vyacheslav Lyashenko (2019) Information Technologies of the Processing of the

Spaces of the States of a Complex Biophysical Object in the Intellectual Medical System HEALTH, *International Journal of Advanced Trends in Computer Science and Engineering*, 8(6), pp. 3221-3227. DOI: 10.30534/ijatcse/2019/89862019

17. Sharma G., and Schiele B. (2015) Scalable Nonlinear Embeddings for Semantic Category-based Image Retrieval, *Proceedings of 2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 7-13.