DOI https://doi.org/10.15407/csc.2022.03.063
UDC 004.94

O.S. BULGAKOVA, PhD (Eng.) Sciences, Associate Professor, Department of Applied Information Systems, Taras Shevchenko National University of Ukraine,
Bohdan Hawrylyshyn str. 24, Kyiv, 04116, Ukraine,
ORCID: https://orcid .org/0000-0002-6587-8573, Scopus Author ID 57188687900,
sashabulgakova2@gmail.com

V.V. ZOSIMOV,D. (Eng.) Sciences , Professor, Department of Applied Information Systems, Taras Shevchenko National University of Ukraine,
Bohdan Hawrylyshyn str. 24, Kyiv, 04116, Ukraine,
ORCID: https://orcid .org/ 0000-0003-0824-4168, Scopus Author ID 57188682230,
zosimovvv@gmail.com

A.V. KUDRIAVTSEV, Master degree of Computer Science, GlobalLogic, Protasov Business Park, str. M. Grinchenko, 2/1, Kiev, 03038, Ukraine,
extosis.vt@gmail.com

APPLICATION OF PROCEDURAL GENERATION ALGORITHMS IN REAL-TIME GAME STRATEGY ENVIRONMENT BASED ON THE MVC CONCEPT

This paper presents an example of using procedural generation methods in a real-time strategy environment created on the basis of the MCC concept. The most important feature of the presented project is its modularity. All game level objects are independent of each other. An algorithm for visualizing objects based on procedural generation is described. The problems and their solutions that arose during the creation of the game are considered.

Keywords: *procedural generation, MVC concept, game, 3D modeling, real-time strategy, unity.*

Introduction

Procedural generation algorithms are indispensable for creating artificial clouds, mountains, sea surface [1–3]. That is why in the development of modern games algorithms for generating three-dimensional structures are used.

However, it is easy to make mistakes in understanding and applying procedural generation: it is very important to understand that it is not a tool to solve all problems. It can be used to get a lot of digital content, or to add an element of randomness to objects that are long and difficult to make by hand. It takes a programmer time to write and test algorithms, especially when they interact with other systems. Therefore, this paper considers an example of using procedural generation methods in a game environment created on the basis of the *MVC* concept with Unity.

The most important feature of the presented project is its modularity, the user should be able to change parts of the project at will with the same ease as the developer. All game level objects are independent of each other — adding new ones or removing existing ones should not harm the entire project as a whole.

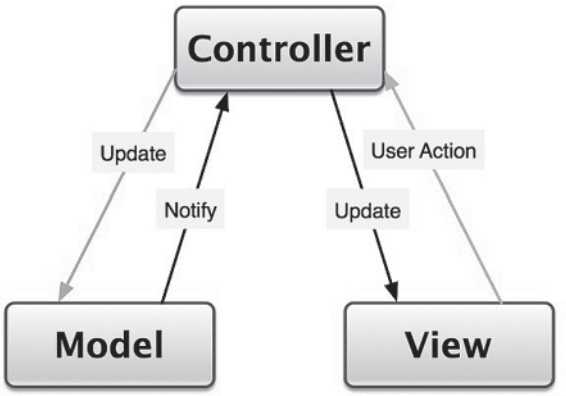

Fig. 1. MVC concept

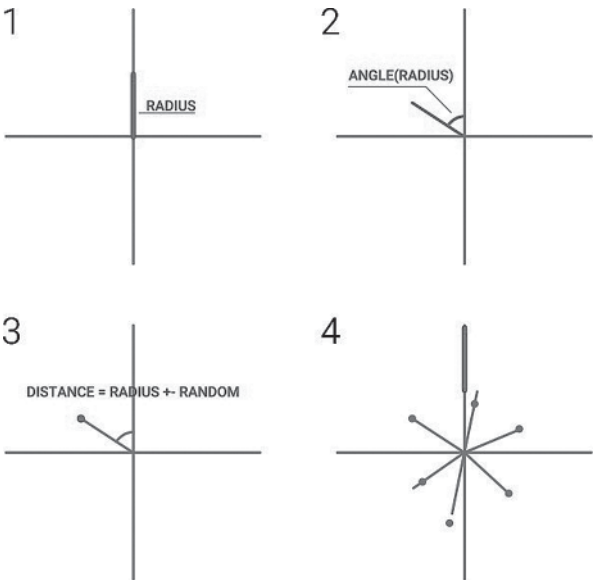

Fig. 2. Algorithm visualization

MVC Concept

Model-View-Controller (*MVC*, "Model-View-Controller") is a program data separation scheme designed to present the user interface and control logic into three separate components: model, view and controller — modification each component can be carried out independently (Fig.1). For these properties, it was decided to take this template as the basis for the project [4].

With the development of object-oriented programming and the concept of design patterns, a number of modifications of the *MVC* concept have been created, which, when implemented by different authors, may differ from the original. So, for example, E. Vermi in 2004 described an example of a generalized *MVC*. The main purpose of applying this concept is to separate the logic (model) from its visualization (view). Due to this division, the possibility of code reuse increases. This concept is most useful when the user needs to see the same data at the same time in different contexts and/or from different points of view.

The concept of *MVC* allows you to separate the model, view and controller into three separate components.

The model provides data and methods for working with them: queries to the database, checking for correctness. The Model is independent of the View (doesn't know how to render the data) and the Controller (has no user interaction points), just providing access to and manipulation of the data.

The model, due to independence from the visual representation, can have several different representations for one model. The view is responsible for getting the required data from the model and sending it to the user.

The controller provides the "communication" between the user and the system — controls and directs data from the user to the system. Conversely, it uses the model and view to implement the required action.

Since *MVC* does not have a strict implementation, it can be implemented in different ways. There is no generally accepted definition of where logic should be placed. It can be in both the controller and the model. In the latter case, the model will contain all objects with all data and functions. Some frameworks rigidly specify where the logic should go, others do not have such rules.

It is also not specified where the validation of data entered by the user should be located. Simple validations can even occur in a view, but they are more common in a controller or model.

Procedural Generation in Global Real-Time Strategy with Elements of an Economic Simu-lator

On the basis of the *MVC* concept, the strategy of the stellar world was implemented with its infra-

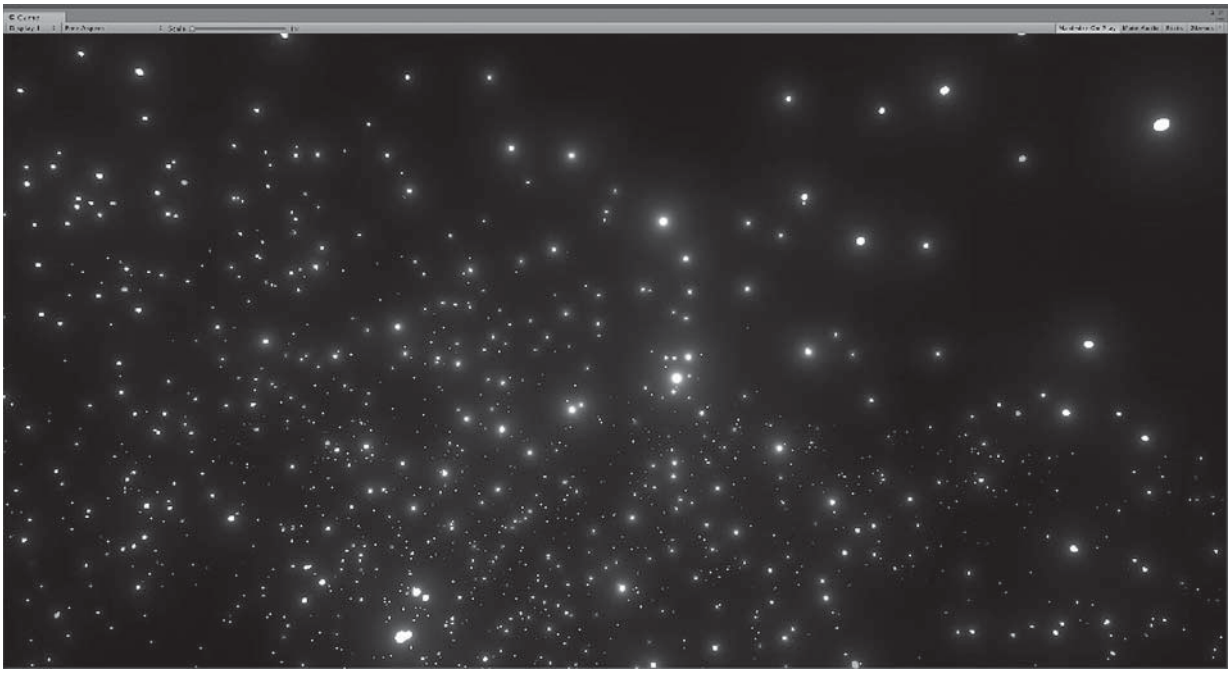

Fig. 3. The result of the algorithm

structure, production, and resources. The most important feature of this project is its modularity, the user should be able to change parts of the project at will with the same ease as the developer. All game level objects are independent of each other — adding new ones or removing existing ones should not harm the entire project as a whole. This point is the most important in the project – its implementation is primary.

The second feature of the project is the in-depth economic part of the game, the player needs to set quite difficult, but solvable tasks, this point is achieved by the depth of processing the severity of the industrial chains of the game world.

The main elements of the economic process of the game are the extraction of resources, the processing of resources, technological research, the development of infrastructure on the planets.

Galaxy Procedural Generation Algorithm

Structure generation algorithm includes 4 steps, Fig.2:

S t e p 1. Offset from the origin by a radius distance.

S t e p 2. Rotate by an angle, depending on the value of the radius and the calculation of the point.

S t e p 3. Sequential calculation of point coordinates, conversion to Cartesian coordinate system and saving to the queue.

S t e p 4. Full rotation and offset by a radius.

The developed method for generating the galactic structure works on the basis of the polar coordinate system; it takes the number of stars as an argument and returns an array of coordinates in the form: vector4{float *x*; float *y*; float *z*; float *w*;}, which are then used to build stars by coordinates.

The algorithm for generating the galactic structure starts from the beginning of the coordinate grid, successively increasing the radius until it reaches the end of iterations, the step of increasing the radius is calculated depending on the number of stars — _iteration Size = 1/(_size/4,5*f*); where _iteration Size is the step, _size is the number of stars (function argument). At each iteration, a sequential rotation is performed by some angle, which can take a value from zero to the function boundary, which depends on the current radius.

After the rotation, a random value is added to the radius and a transformation into a Cartesian coordinate system is performed, during which the height of the coordinate is calculated.

a	10	20	50	40	35	15
b	15	20	25	16	5	0
c	0	0	5	10	15	10
d	0	0	0	0	5	15

Fig. 4. View of arrays of distribution of objects a-d

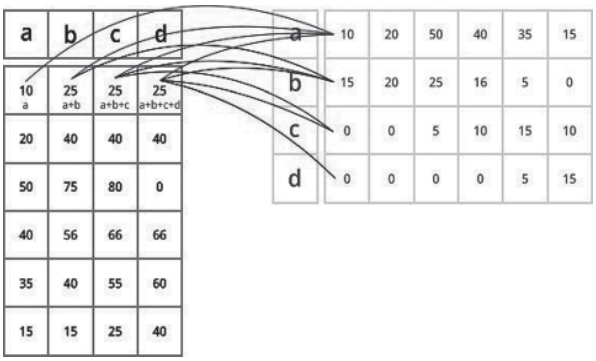

Fig. 5. Type of array of frequencies of objects a-d and the principle of its construction

Subsequently, the coordinate is added to the queue and the next rotation is performed until a full rotation is completed.

The algorithm presented in this paper has the following disadvantages:

- the speed of work is almost impossible to calculate, because it depends on a random variable;
- It is impossible to control the operation of the algorithm without direct intervention and modification of the program code.

Algorithm advantages:

- despite the fact that the speed of the algorithm cannot be calculated, experiments show that the algorithm works quite fast even on weak systems;
- the algorithm practically does not use RAM to process values only for storage;
- the result of the work can be applied without current processing by other algorithms.

The result of the algorithm is shown in Fig. 3.

Ancillary Generation Algorithms

After the operation of the map generator, static algorithms of ancillary generation are used to further calculate the map. To separate the generation sectors of a galaxy, we use such a concept as an array-distribution.

Array-distribution — describes the frequency of meeting an object in certain areas of the game map. The distribution array is a part of separate objects — generation participants. In work, stars and constellations have such vulnerability. The view of the distribution arrays of objects a-d is shown in Fig. 4

Using the construction function, based on the distribution arrays of all objects, a two-dimensional array of frequencies is built, which in turn is the main tool for pseudo-random selection of an object from the list.

The random selection function works on a separate part of the frequency array, recalculating the weight coefficient for each object, Fig.5.

In the case of stars, using this method, you can specify in which part of the galaxy which star should have generation priority by setting a certain gradation of the distribution of objects. For example, in the work, white and blue stars are generated mainly in the center of the galaxy, while red and brown ones are closer to the edge of the galaxy. However, it is these algorithms that need a new generation system, which, in addition to the usual coordinates, indicates the dis-tance from the center of the galaxy, Fig.6.

The division of the game map levels and its effect on the generation

The method of dividing the map into two parts was used in the work: global and local. The global map includes the galaxy and builtin user interface tools. The local map includes the planetary system, which is described on the global map. During the game, the user can only open one of these cards, the other is automatically disabled.

The construction of both maps at once makes it necessary to generate the positions of objects placed on both maps. For example, a function for generating the positions of stars in a constellation must simultaneously calculate global coordinates, local coordinates, and orbital centers.

On the other hand, such a division of the game level reduces the simultaneous display of a large number of objects, and this is one of the types of CPU time optimization.

To simplify the interaction of such objects during generation, they are mostly generated separately, and some of them are generated at the request of the user. For example: a graphical user interface is assigned to a separate planet, it is generated immediately when the user needs to view the resources of the planet.

Problems and Solutions

Optimization of the number of frames/second while viewing a global map

In computer games, frame rate refers to the rate at which the game process updates the image in the frame buffer. Typically, the frame rate of the gameplay is not a multiple of the frame rate of the monitor, resulting in a ragged image. This is due to the order in which the object is drawn by the video card, namely the number of draw calls sent — draw call [5].

To reproduce an object on the screen, the engine sends a command (draw call) to a graphics API (for example, OpenGL or Direct3D). The graphics API does a lot of work for each instruction, which greatly affects the performance of the CPU. In the process of rendering any object on the screen, the CPU must find out which light sources affect the object, adjust the shader and its parameters, send playback commands to the graphics driver, which will prepare commands for sending to the video card. All these processes load the CPU.

There are two approaches to solving this problem:

1. Geometry Instancing.

Geometry Instancing is a software technique in mostly real-time 3D computer graphics. The essence of Geometry Instancing is to render multiple copies of the same polygonal mesh in a 3D scene in one go. This technique is used, as a rule, for many objects of the same type on the stage, which are far enough from the virtual camera [6].

Geometry Instancing is primarily an optimization technique designed primarily to increase rendering speed without compromising quality.

When using the standard approach to rendering a scene and all its objects, only one object is formed per Direct3D call (by vertices, lighting, etc.). Before Geometry Instancing, distant identical objects could be rendered as sprites, that is, two-dimensional textures. However, when approaching such two-dimensional objects and changing the observer's point of view, their two-dimensional nature becomes visible. Geometry Instancing solves this problem. When using it, you can immediately render the geometry of all similar identical objects in one call to Direct3D. This will save system resources and increase the realism of the scene as a whole compared to sprites. Despite the fact that the geometry data is the same when duplicating, however, each copy can have different parameters. This reduces the visibility of repeating objects in the scene.

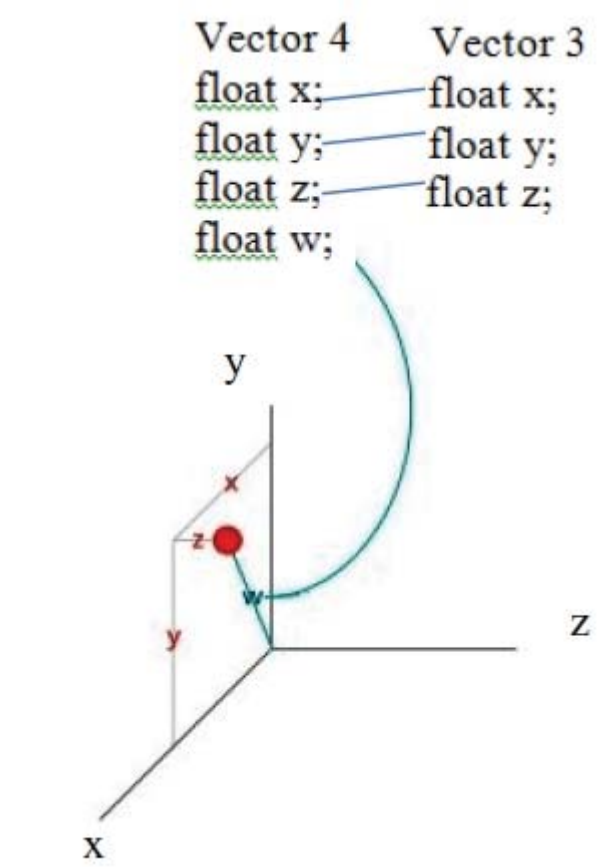

Fig. 6. Description of coordinate type "Vector4"

2. Mesh Combine

Mesh Combine is a way to create a combined object from multiple objects. So instead of having mul-

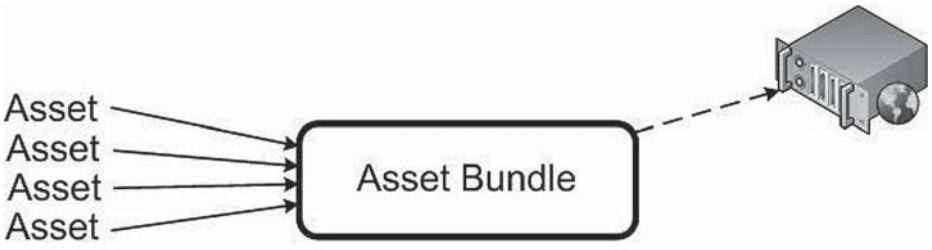

Fig. 7. Loading resources on the server

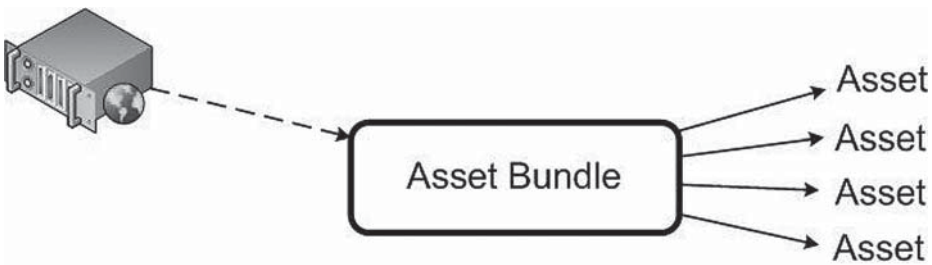

Fig. 8. Loading resources from the server

tiple objects and therefore multiple play calls, you end up with one big game object containing several smaller ones connected together. This allows you to increase the performance of the game [7].

In this project, we had to use a combined approach to optimize the geometry: the part of the geometry that supports duplication is duplicated, and the part of the geometry that does not support is merged, which allowed us to significantly increase the frame rate.

The problem of streaming loading resources into the project

Compilation in the Unity engine works in such a way that all used materials, models, images, are packed into the internal format of the game, without the possibility of further changes. To be able to change files at the request of the user, it was necessary to implement streaming downloads.

There are two solutions to this problem: streaming normal files and converting them to formats that the game engine can handle, or packaging already converted files into assets and loading them at run time. In the work on the project, from the point of view of implementation, the second option was chosen.

AssetBundles are files that can be exported from Unity to package resources (assets) into a single file of your choice. These files use their own compression format and can be downloaded by the user's application as needed. This allows you to load different content, such as models, textures, audio clips, or even entire scenes, separate from those in which they will be used.

Here is a typical workflow using a server architecture as an example: during the development process, the developer prepares AssetBundles and sends them to the server. Collection of AssetBundles. Asset bundles are created in the scene resource editor, Fig.7.

While running on the user's computer, the application will load AssetBundles on demand and allow individual Assets within each AssetBundle to be managed as needed, Fig.8.

Content extension problems and the possibility of adding it to the compiled project

The problem of expanding content with custom modifications is partially solved by using a system for loading add-ons and custom modifications from a special folder in the project's file system.

With the help of stream loading, all necessary components can be added to the project, which have been previously packaged in asset format and moved to the appropriate folder in the uncompiled or compiled project.

Conclusion

The paper proposed the main tools that are currently used to create procedurally generated game content, and identifies the capabilities and advantages of game engines. Based on procedural generation algorithms, a real-time economic strategy was developed using the Unity environment. The most important feature of the presented project is its modularity.

All game level objects are independent of each other. An algorithm for visualizing objects based on procedural generation is described. The problems and their solutions that arose during the creation of the game are considered.

REFERENCES

1. Bulgakova, O.S., Kudriavtsev, A.V., Zosimov, V.V., Pozdeev V.O. (2019). "Algorithmic modifications in procedural generation systems". Control Systems and Computers. Vol. 3, pp. 52-59.
2. Pereira, L., Viana, B., Toledo, C. (2021). "Procedural Enemy Generation through Parallel Evolutionary Algorithm". Proceeding of SBGames 2021.
3. Yannakakis, G., Togelius, J. (2018). "Artificial Intelligance and Games". Springer, 2018.
4. Unity With MVC: How to Level Up Your Game Development. [online]. Available at: <https://www.toptal.com/unity-unity3d/unity-with-mvc-how-to-level-up-your-game-development> [Accessed 06 Sept. 2022].
5. Unity Scripting Reference. [online]. Available at: <https://docs.unity3d.com/ScriptReference/> [Accessed 10 Aug. 2022].
6. GPU instancing on Geometry Shader. [online]. Available at: <https://forum.unity.com/threads/gpu-instancing-on-geometry-shader.1106152/> [Accessed 22 Aug. 2022].
7. Simple Mesh Combine. [online]. Available at: <https://unityassets4free.com/simple-mesh-combine/> [Accessed 02 Aug. 2022].

Received 30.10.2022

ЛІТЕРАТУРА

1. Bulgakova O.S., Kudriavtsev A.V., Zosimov V.V., Pozdeev V.O. Algorithmic modif-?cations in procedural generation systems. Control Systems and Computers. 2019. Vol. 3, pp. 52-59.
2. Pereira L., Viana B., Toledo C. Procedural Enemy Generation through Parallel Evolutionary Algorithm. Proceeding of SBGames 2021.
3. Yannakakis G., Togelius J. Artificial Intelligance and Games. Springer, 2018.
4. Unity With MVC: How to Level Up Your Game Development. URL: https://www.toptal.com/unity-unity3d/unity-with-mvc-how-to-level-up-your-game-development (дата звернення: 6 вересня 2022).
5. Unity Scripting Reference. URL: <https://docs.unity3d.com/ScriptReference/> (дата звернення: 10 серпня 2022).
6. GPU instancing on Geometry Shader. URL: <https://forum.unity.com/threads/gpu-instancing-on-geometry-shader.1106152/> (дата звернення: 22 серпня).
7. Simple Mesh Combine. URL: <https://unityassets4free.com/simple-mesh-combine/> (дата звернення: 02 серпня 2022).

Надійшла 30.10.2022

О.С. Булгакова, кандидат техн. наук, доцент, Київський національний університет України імені Тараса Шевченка, 04116, м. Київ, вул. Богдана Гаврилишина, 24, Україна,
ORCID: https://orcid .org/0000-0002-6587-8573, Scopus Author ID 57188687900,
sashabulgakova2@gmail.com
В.В.Зосімов, доктор техн. наук, професор, Київський національний університет України імені Тараса Шевченка, 04116, м. Київ, вул. Богдана Гаврилишина, 24, Україна,
ORCID: https://orcid .org/ 0000-0003-0824-4168, Scopus Author ID 57188682230,
zosimovvv@gmail.com
А.В. Кудрявцев, магістр, GlobalLogic, Протасов Бізнес-Парк,
03038, м. Київ, вул. М. Грінченка, 2/1, Україна,
extosis.vt@gmail.com

ЗАСТОСУВАННЯ АЛГОРИТМІВ ПРОЦЕДУРНОЇ ГЕНЕРАЦІЇ В СЕРЕДОВИЩІ ІГРОВОЇ СТРАТЕГІЇ РЕАЛЬНОГО ЧАСУ НА ОСНОВІ КОНЦЕПЦІЇ MVC

Вступ. Алгоритми процедурної генерації є незамінними для створення штучних хмар, гір, морської поверхні. Саме тому при розробці сучасних ігор використовуються алгоритми генерації тривимірних структур. Однак легко помилитися в розумінні та застосуванні процедурної генерації: дуже важливо розуміти, що це не інструмент для подолання всіх проблем. Його можна використовувати для отримання великої кількості цифрового контенту або для додавання елемента випадковості до об'єктів, які довго й важко виготовляти вручну. Програмісту потрібен час, щоб написати та перевірити алгоритми, особливо коли вони взаємодіють з іншими системами. Тому в цій статті розглядається приклад використання методів процедурної генерації в ігровому середовищі, створеному на основі концепції *MVC* з *Unity*.

Мета статті: розглянути основні інструменти, які використовуються для створення процедурно згенерованого ігрового контенту, а також визначити можливості та переваги ігрових рушіїв; на основі процедурних алгоритмів генерації розробити економічну стратегію реального часу з використанням середовища *Unity*.

Методи. Системний підхід, аналіз.

Результати. Представлено приклад використання процедурних методів генерації в середовищі стратегії реального часу, створеного на основі *MVC* концепції. Найважливішою особливістю представленого проєкту є його модульність. Усі об'єкти рівня гри є незалежними один від одного. Описано алгоритм візуалізації об'єктів на основі процедурної генерації. Розглянуто проблеми та способи їх розв'язання, що виникли під час створення гри.

Висновки. Результати цього дослідження показують, що використання методів процедурної генерації в ігровому середовищі є достатньо ефективним. Найважливішою особливістю представленого проєкту є його модульність, користувач має можливість змінювати частини проєкту за бажанням з такою ж легкістю, як і розробник. Усі об'єкти рівня гри є незалежними один від одного — додавання нових або видалення наявних не шкодить усьому проєкту як цілому.

Ключові слова: *процедурна генерація, концепція MVC, гра, 3D моделювання, стратегія реального часу, ігрові системи.*