ISSN 2224-087X. Електроніка та інформаційні технології. 2020. Випуск 13. С. 77–87
Electronics and information technologies. 2020. Issue 13. P. 77–87

УДК 004.02 DOI: https://doi.org/10.30970/eli.13.8

IN GAME MAP GENERATION USING RANDOM PATTERN GENERATION

V. Kushnir, B. Koman, R. Shuvar

Lviv Ivan Franko National University,
50 Drahomanova St., 79005 Lviv, Ukraine
vasylll95@gmail.com , bohdan.koman@lnu.edu.ua , rshuvar@gmail.com

In this work was conducted analysis in machine learning sphere for self-driving cars. Machine learning helps to extend spheres where of Artificial Intelligence where simple algorithm could not help.. That's why developers are trying to create, find and form different data sets for their Artificial Intelligence training and improvement.

Every Artificial Intelligence needs their specific data set, for executing some actions. For example, for face identification we need dataset with faces, for text segmentation – text corpuses. That's why Artificial Intelligence for self-driving cars needs their own data set.

Methods, which are used for its implementation, they need to build an iterative training on game maps, which will give results depends on training object on the map. In this work is described reinforcement learning method, which is a basis for building self-driving systems. Also was conducted examples of a training such Artificial Intelligence and displayed game map that are developed and available nowadays.

To provide such game maps, different algorithms are used to generate such game maps. Such algorithms are used in game industries for building levels that can have different sizes, event bigger than our planet. That's why in this work were conducted descriptions to the companies and games that are using automatic game map generation.

In this work was conducted verification of an algorithm for road infrastructure generation and investigated algorithms for game map generation and as a result was implemented simple and fully automated algorithm for generating infrastructure. To build a full picture about such algorithm was used game engine called Unreal Engine 4 and method optimizations for more descriptive illustration.

To make investigation more clearer was considered in details Unreal Engine 4 and made a comparison between next most popular game engine called Unity. So it is provided with advantages and disadvantages between those engines. Also pros and cons were described.

Keywords: Unreal Engine 4, game engine, reinforcement learning, map generation, OpenGym, algorithm, graph, Artificial Intelligence, game map.

Introduction.

Map generation and development is one of the key to open virtual world in different shapes. For example, there are a lot of games that has area over thousands like: THE ELDER SCROLLS II: DAGGERFALL (160579 km^2), JUST CAUSE 3(1036 km^2) [1], etc. They called as games with open world. Hundreds of level designers are working to implement such maps. To make map development faster, developers creates a lot of plugins and tools like: splines, modular foliage generation, terrain generation tools, etc. Some companies are working on full generated maps. They are generating maps using various algorithms. Company like Mojang

has developed a game called Minecraft with fully generated map [1]. When the player creates a new world, game automatically generates new map. Area of this world equals 60000 km^2. Their algorithm could generate an infinity map but it decrease game performance tremendously [1].

As self-driver cars shows us that generated maps are not only for games. They are also helping in machine & deep learning fields. Precisely maps are helpful in reinforcement learning when you are trying to train your car to self-drive or a plane to fly by himself. Such type of learning uses trial and error method. It means when model make a right decision it gets a reward and if it makes a wrong decision – we take away a reward [2]. In fig.1, it is described as a cycle of actions, state and rewards that are going it positive or negative direction, depends of the agent and the state.

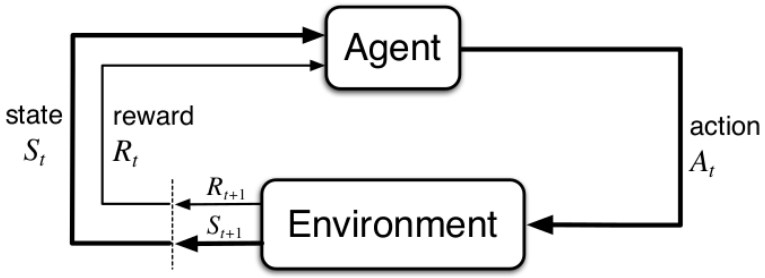

Fig. 1. Reinforcement learning workflow.

For these models was created training areas called gyms. Most popular of them is OpenGym [2]. It contains different areas to train depends of you model and a task. In Fig. 2, we can see different games that contains such tool [2].

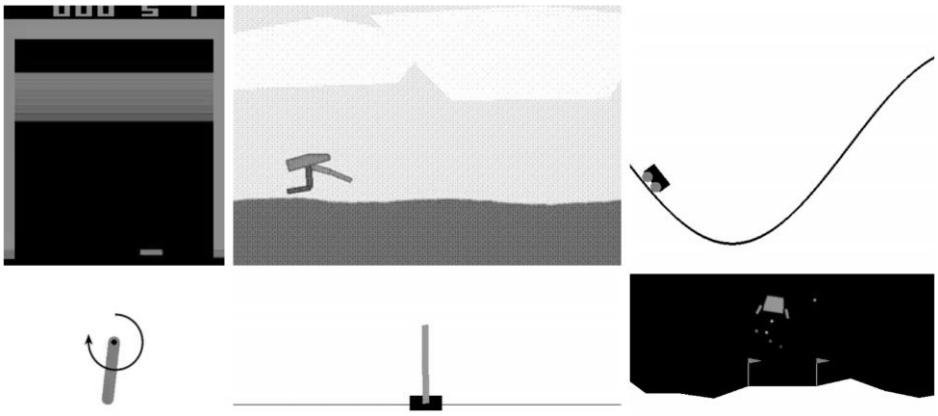

Fig. 2. OpenGym preview games for training.

Unreal Engine 4.

For this work was chosen Unreal engine 4 as an engine for map generation because it contains a lot of helpful feature like:

1) it has much better cross-platform integration. It enables you to deploy your game across all major mobile, VR, desktop, console, and TV platforms plus the Web. The supported platforms include Android, iOS, Windows Phone, Tizen, PC, Mac, Linux, PS4, Xbox One, PlayStation Mobile, PlayStation Vita, Wii U, tvOS, Android Tv, and Samsung Smart TV. Native support is available for major VR platforms including Oculus Rift, Gear VR, Playstation VR, Microsoft HoloLens and Steam VR/Vive [10];
2) it is simple to use Unity engine in mobile development, however Unreal engine 4 is commonly used to implement simulations and prototypes with use of visual programming that called Blueprints [10];
3) unreal engine 4 best fits for large scale projects because it uses C++ instead of C# like in Unity [10];
4) Unreal Engine 4 is great for high-tech triple-A games with big budgets where you need to achieve the ultimate quality. So in terms of graphics it really takes the lead. For example, we've picked Unreal for one our recent VR projects because the client's main request was to make this demo presentation as realistic as possible;
5) it has available instruments for many cases and needs, which Unity lacks. For instance, Unity is limited to a set of standard shaders and Unreal has visual editors for them. With Unity, you'll have to write the same things in code;
6) the Unreal Engine has better access to visual debugging, you can easily test what, when and how is being drawn, you can control the whole process and monitor how much time is spent on a certain element.

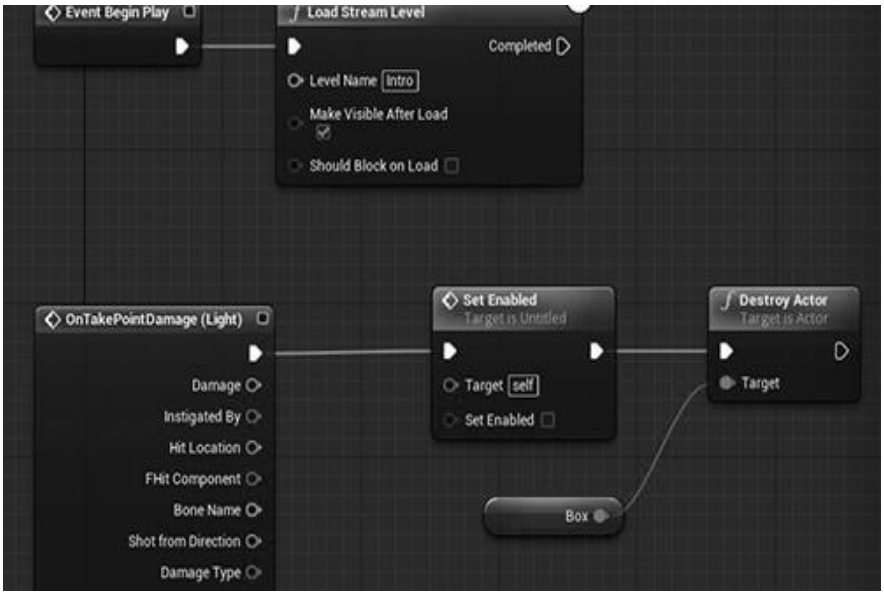

Fig. 3. Visual programming language "Blueprint".

Alogorithms for map generation.

During investigation was found different algorithms that helps to improve road generation a basis of building whole map [4]. First algorithm pattern map generator. It must contain patterns (Fig.4) which is used to connect into map[5].

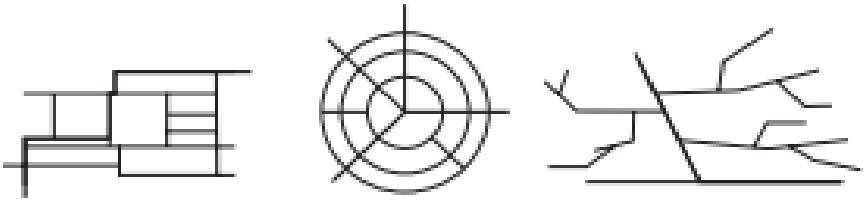

Fig. 4. Patterns for map pattern generation.

As a result by pattern concatenation it gives them very interesting results (Fig.5).

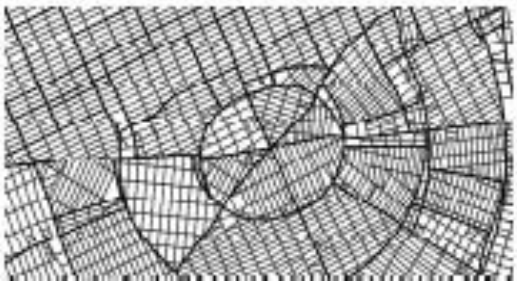

Fig. 5. Generated map using patterns.

Next algorithms was Perlin noise method. This algorithm is used not to generate roads, but very effective to generate terrains. In fig. 6 we can see that by using such method we can setup that white is a high point and dark is a low point. With these parameters we can generate map that is on fig. 7 [9].

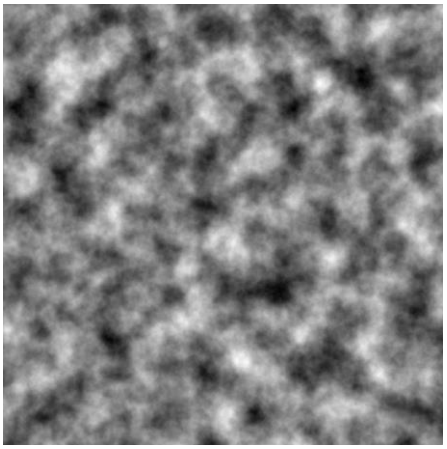

Fig.6. Perlin noise image [7].

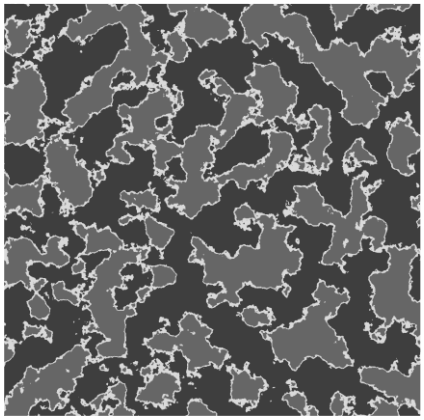

Fig. 7. Map after applying Perlin noise method.

The third was L-Systems map generation algorithm. It is regarding of generating sequence of letters (fig. 8) [8]. So this sequence converts to horizontal and vertical directions which helps to build a sequence of routes, mostly in rectangle manner. Very interesting approach because it gives ability to build long routes across maps which is more realistic (fig. 9).

Also to note this method is not a route generation. It is more like building generation [8].

Start=A. Rules: AB; B=BA

1. A
2. AB
3. ABBA
4. ABBABAAB
5. …

Fig.8 Sequence of string for map generation.

Implementation of Simple Grid Map Generation using Pattern generation.

To make map generation was started from route infrastructure generation. This is fully automative method to generate such map [3]. Algorithm is to generate patterns and then combine them into one route graph. To generate pattern was generated initial field performed as a grid and then randomly removes edges from grid and it was completed (fig. 10) [4]. Such patterns has pros and cons. As pros: it simple to generate, it is unique of random removal, it independent from other patterns, easy to use because each edge contains only 1 point length. As cons: does not contain real data, need to improve to make other types of routes like bridges, tunnels, circle routes, etc.

By combining new generated patterns we got a route infrastructure (fig. 11). With whole route infrastructure we can build our own map [5]. As each pattern is unique it helps to generate new map each time. Then we need to add a plane object as a surface and pass route on it (fig .12).

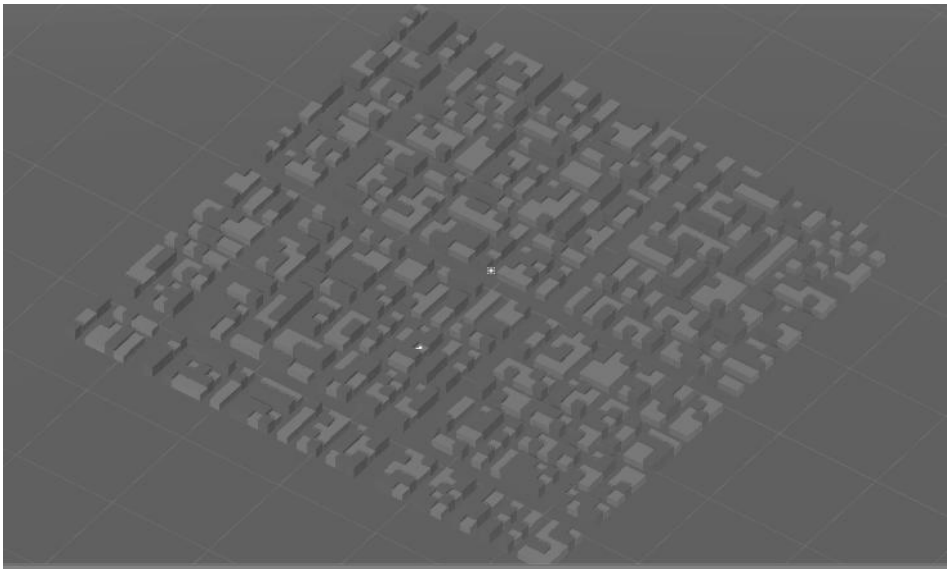

Fig. 9. Result of L-Systems generation.

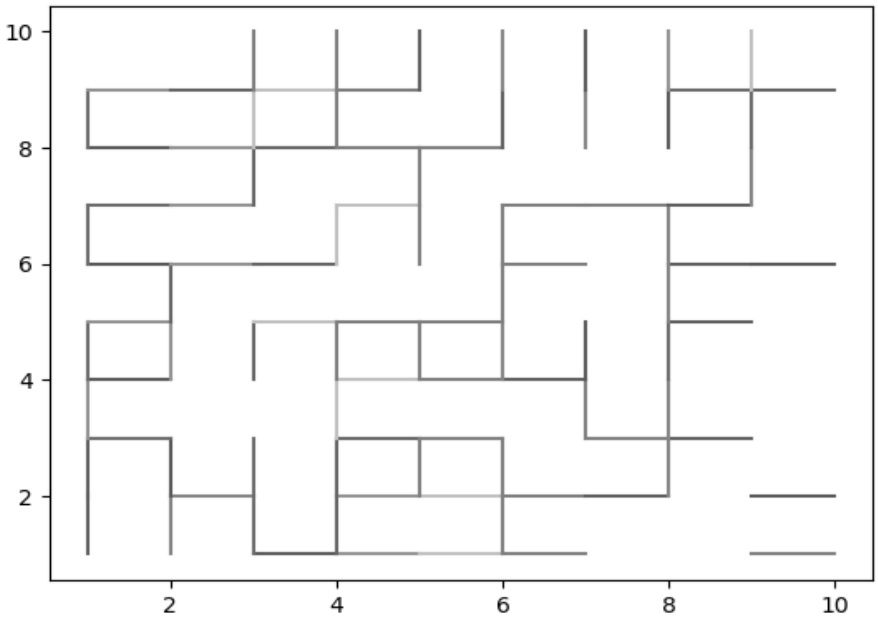

Fig. 10. Generated pattern from the Grid.

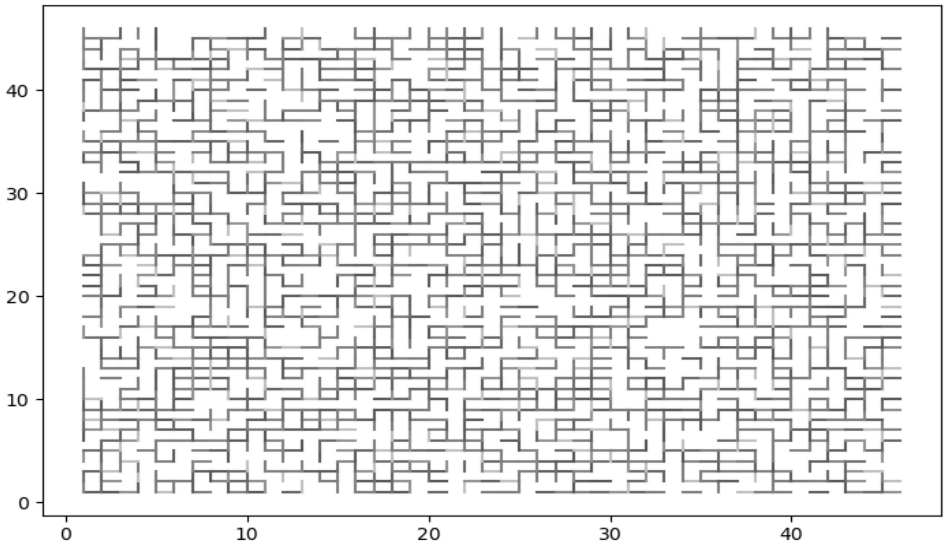

Fig. 11. Whole Generated route infrastructure.

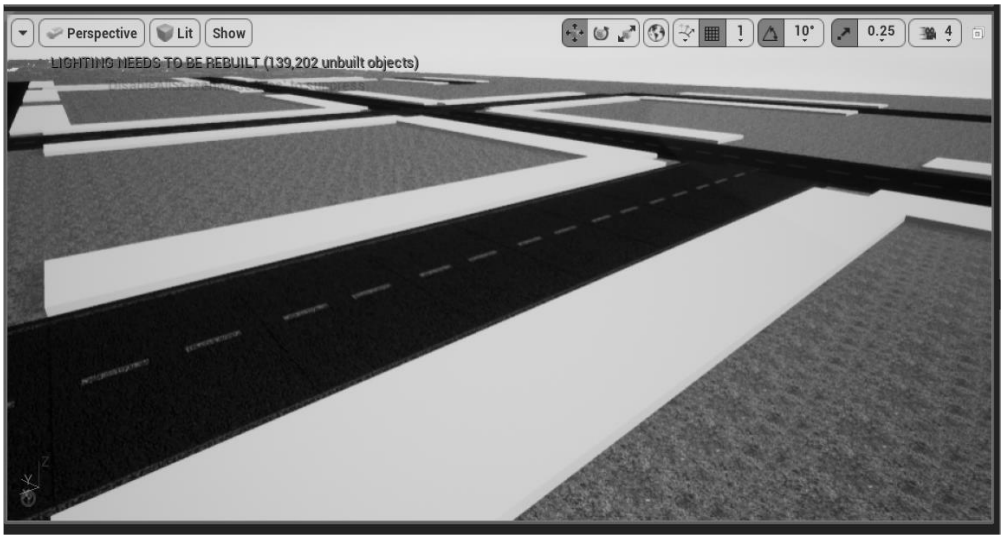

Fig. 12. Generated plane surface with route and sidewalks according generated infrastructure.

Next step was to add buildings to the map. To add them was proposed to build an array of buildings and using shuffle method it is chosen a building to put on the land.

To prevent colliding and interception between routes, sidewalks and building was added tracer to each building. In Unreal Engine 4 it is called BoxTraceForObjects method. This method helps to get which of this objects are colliding with building and it make a decision should it be on this location or not.

By iterating through map buildings was installed (fig. 13). As a result we get a map with routes and buildings.

Overall algorithm is:

1. Generate grid.
2. Remove edges from the grid randomly.
3. Repeat several times.
4. Combine it.
5. Generate planes as land.
6. Generate routes.
7. Generate sidewalks near routes by adding Sockets to routes.
8. Add buildings by adding near routes using BoxTraceForObjects method.

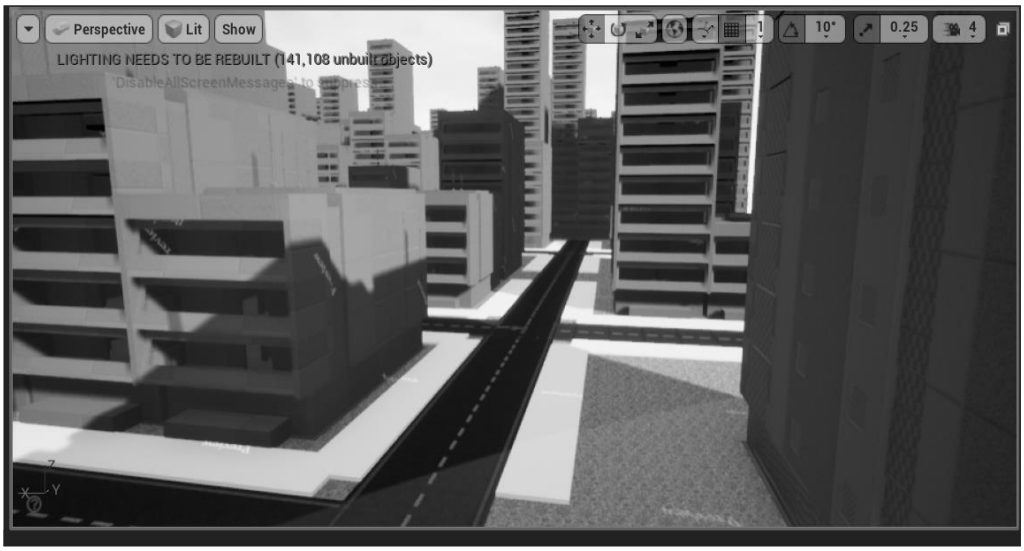

Fig. 13. Generated map detailed.

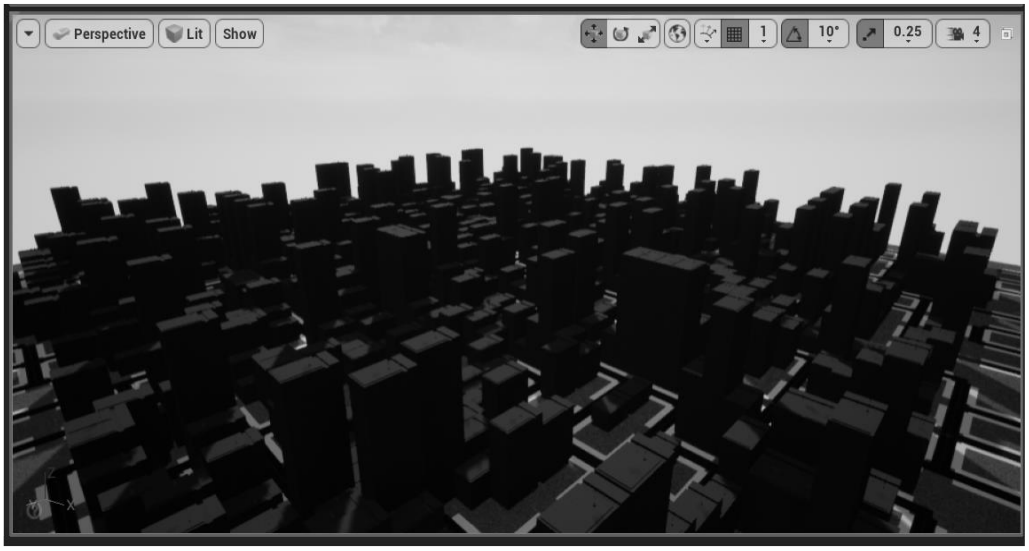

Fig. 14. Generated map.

Comparison among map generator algorithms.

Map generator using patterns

Pros:

1. Map depends on which patterns will be passed.
2. O(N) difficulty for generation.
3. Can be provided additionally.

Cons:

1. Less used.
2. Not fully automated.

Perlin noise method

Pros:

1. Fully generated
2. Used perlin algorithms.
3. Mostly used in games.

Cons:

1. Mostly used for terrain generation.
2. Perlin noise therefore scales with complexity $O(2^N)$for N dimensions

L-Systems generation

Pros:

1. Using alphabet to generate map.
2. Complexity is very dependant from rules you pass of L-System

Cons:

1. Almost automated because we need setup rules.
2. Complexity can be O(N^2)

Grid Map Generation using Pattern random generation

Pros:

1. Complexity is O(N)
2. It is fully automated.
3. Can be improved by passing different algorithm instead random.
4. Generates set of unique patterns that covers more real life route chains.

Cons:

1. Random is not always a solution for some cases.
2. It is represented as a grid so it needs to be improved by adding more complex solutions and shapes.

This work shows investigation around map generation and it seems that there is a lot of methods that can be found. This field is very important nowadays in several spheres which opens opportunity to work further and improve from algorithmic and visual perspective. Compare to other, proposed algorithm shows that it could generate diverse map because it is consists of unique patterns that was randomly generated, that helps to cover most of real life route situations. Also using random generated patterns map will be fully automated.

REFERENCES

1. AI Gym [Electronic source] – Available from: https://gym.openai.com .
2. Simple Grid Map Generation Algorithm [Electronic source] – 2016 – Available from: https://projectluckyluciano.wordpress.com/2016/02/18/simple-grid-map-gen
3. *J.C. Hart*. The Object Instancing Paradigm for Linear FractalModeling. In Proceedings of Graphics Interface 92, pages224-231, 1992.
4. *Yoav I.* Procedural Modeling of Cities / I. Yoav, M. Pascal. – 2001.
5. *B. M. Blumberg and T. A. Galyean*. Multi-Level Direction of Autonomous Creatures for Real-Time Virtual Environments. In SIGGRAPH 95 Conference Proceedings, pages 47-54, August 1995
6. Making maps with noise functions [Electronic source] – Available from: https://www.redblobgames.com/maps/terrain-from-noise/
7. L-Systems Map Generation Algorithm [Electronic source] – Available from: https://projectluckyluciano.wordpress.com/2016/04/18/l-systems-map-generation-algorithm/.
8. Playing with Perlin Noise: Generating Realistic Archipelagos [Electronic source] – Available from: https://medium.com/@yvanscher/playing-with-perlin-noise-generating-realistic-archipelagos-b59f004d8401.
9. Unreal Engine 4 documentation – [Electronic source] – Available from: https://docs.unrealengine.com/en-US/index.html

ГЕНЕРАЦІЯ ІГРОВОЇ КАРТИ З ВИКОРИСТАННЯМ ВИПАДКОВО ЗГЕНЕРОВАНИХ ПАТЕРНІВ

В. Кушнір, Б. Коман, Р.Шувар.

Львівський національний університет імені Івана Франка,
вул. Драгоманова, 79005, м. Львів, Україна
vasylll95@gmail.com , bohdan.koman@lnu.edu.ua , rshuvar@gmail.com

В роботі проведено аналіз галузі машинного навчання для самокерованих автомобілів. Машинне навчання дає змогу розширити сфери використання Штучного інтелекту де не справиться звичайний алгоритм. Тому розробники створюють, збирають та формують різні набори даних для тренування і покращення Штучного Інтелекту.

Кожному Штучному Інтелекту потрібен свій набір даних, для виконання певних дій. Наприклад, для розпізнавання обличчя потрібен набір картинок з обличчями, а для сегментації тексту – корпуса текстів. Тому для Штучного Інтелекту для самокерування транспортом потрібен свій набір даних.

Методи, які використовують для побудови самокерованого автомобіля проводять ітеративне навчання на ігрових картах, які дають змогу отримати бажаний результат за допомогою навчального об'єкта та стану на карті. Тому в роботі описано навчання з підкріпленням, яке є базисом у побудові самокерованих систем. Також приведені приклади навчання такого Штучного Інтелекту та зображено ігрові карти, які існують і доступні станом на сьогодні.

Генерують ігрові карти з використанням різних алгоритмів. Такі алгоритми використовуються в ігровій індустрії для побудови рівнів, які можуть мати різні розміри, навіть більше ніж наша планета. Тому в роботі приведено опис компаній та ігор, в яких використовується автоматична генерація ігрової карти.

В роботі проведено перевірку алгоритмів для генерації дорожньої інфраструктури та досліджено алгоритми побудови ігрової карти, в результаті якого було реалізовано простий та повністю автоматичний алгоритм генерації інфраструктури з використання випадкових дорожніх патернів. Він є основною складовою автоматизації алгоритму. Він дає змогу отримувати неочікувані результати, які складно згенерувати людині. Для повного представлення та візуалізації даного алгоритму було використано ігровий рушій Unreal engine 4 та методи оптимізації для ілюстрації карти. Рушій та ігрова карта

Для чіткого дослідження було детально розглянуто рушій Unreal Engine 4 та порівняння з іншим відомим рушієм Unity. Були приведені переваги та недоліки даних рушіїв. Окрім представлення результатів алгоритму, було описано порівняння між існуючими алгоритмами та описані їхні переваги і недоліки.

Ключові слова: Unreal Engine 4, ігровий рушій, навчання з підкріпленням, генерація карт, OpenGym, алгоритм, граф, Штучний Інтелект, ігрова карта.

Стаття: надійшла до редакції 27.05.2020,
доопрацьована 06.06.2020,
прийнята до друку 08.06.2020