ISSN 2079-5459 (print)
ISSN 2413-4295 (online)

УДК 004.94 doi:10.20998/2413-4295.2019.02.08

О ПРОЦЕДУРНОЙ ГЕНЕРАЦИИ ТРЕХМЕРНЫХ ВИРТУАЛЬНЫХ ПРОСТРАНСТВ

М. А. ХРЕБЕТ*, Г. Н. КОДОЛА, И. В. СЕРБУЛОВА

кафедра Информационных систем, ГВУЗ УГХТУ, Днепр, УКРАИНА
**e-mail: khrebet.michael@gmail.com*

АННОТАЦИЯ *В современных реалиях процедурная генерация контента является одним из наиболее востребованных и перспективных направлений в среде разработки игр. В наше время качественно выполненный и хорошо продуманный алгоритм процедурной генерации может взять на себя большую часть работы по созданию уникального контента во время разработки игры, или же во время самого игрового процесса. С помощью процедурной генерации можно в кратчайшие сроки создать набор уникальных объектов. Существует большое разнообразие всевозможных алгоритмов процедурной генерации, каждый из которых имеет свои плюсы и минусы, и лучше всего подходит к определенному типу заданий. В данной статье речь пойдет об алгоритмах, которые могут быть использованы для создания трехмерного игрового уровня, что позволит значительно сократить время на работу по созданию таких уровней, а также позволит повысить уникальность каждого сгенерированного уровня. В качестве примера для ознакомления в тексте статьи изложено пять существующих алгоритма процедурной генерации, которые так или иначе позволяют сгенерировать трехмерный игровой уровень. Конечной целью является нахождение оптимального алгоритма процедурной генерации, который будет удовлетворять главному критерию отбора – создание полноценного игрового уровня с ландшафтом и объектами, который не нуждается в дальнейшей доработке. В данный момент существует не так много игр, которые успешно используют алгоритмы процедурной генерации при создании полноценного трехмерного игрового мира – к наиболее известным играм можно отнести Minecraft, No Man's Sky и Deep Rock Galactic. Итоговым результатом исследования стало нахождение наиболее приемлемого алгоритма процедурной генерации, с помощью которого можно в реальном времени генерировать уникальные игровые уровни. В качестве основного требования к алгоритму выступало отсутствие необходимости дальнейшей работы с созданным уровнем, потому как использование данного алгоритма планируется в первую очередь для разнообразия игрового процесса и создания большего элемента случайности. В результате фаворитом оказался алгоритм разделения уровня на сектора, ввиду выполнения им основного условия, а также возможности программирования нескольких условий.*
Ключевые слова: *процедурная генерация контента; алгоритм; уровень игры; трехмерная генерация; компьютерная игра; игровой ландшафт*

ABOUT THREE-DIMENSIONAL PROCEDURE VIRTUAL SPACES

M. KHREBET, G. KODOLA, I. SERBULOVA

Department of Information Systems, Ukrainian State University of Chemical Technology, Dnipro, UKRAINE

ABSTRACT *In modern realities, procedural content generation is one of the most popular and promising areas in the game development environment. Nowadays, a well-executed and well-designed procedural generation algorithm can take on most of the work of creating unique content during game development, or during the game itself. Using procedural generation, you can quickly create a set of unique objects. There is a wide variety of various procedural generation algorithms, each of which has its pros and cons, and is best suited to a specific type of task. This article focuses on algorithms that can be used to create a three-dimensional game level, which will significantly reduce the time for creation of such levels, and will also increase the uniqueness of each generated level. As an example, in the text of the article, five existing algorithms of procedural generation are described, which in one way or another makes it possible to generate a three-dimensional game level. The ultimate goal was to find the optimal procedural generation algorithm that will satisfy the main selection criterion – the creation of a full-fledged game level with landscape and objects, which does not need further refinement. At the moment, there aren't many games that successfully use procedural generation algorithms to create a full-fledged three-dimensional game world – the most famous games are Minecraft, No Man's Sky and Deep Rock Galactic. The final result of the study was to find the most acceptable procedural generation algorithm with which it is possible to generate unique game levels in real time. The main requirement for an algorithm was the lack of the need for further work with a created level, because the use of this algorithm is planned primarily for a variety of gameplay and creation of a larger element of randomness. As a result, the best one was the algorithm for dividing a level into sectors, due to the fulfillment of the main condition, as well as the possibility of programming several conditions.*
Keywords*: procedural content generation; algorithm; level of play; three-dimensional generation; computer game; game landscape*

Введение

Процедурная генерация в современном мире снискала себе большую любовь, как разработчиков, так и рядовых игроков. Она нашла себе применение в самых разных областях, но наиболее часто ее применяют при разработке компьютерных игр.

При помощи различных алгоритмов процедурной генерации можно наполнить игру несчетным количеством игрового контента. Один и тот же алгоритм процедурной генерации может отвечать, как за создание непохожих друг на друга деревьев, так и за полную генерацию игрового уровня.

В последнее время с помощью алгоритмов процедурной генерации часто создают различные виды оружия, присутствующие в игре, здания и их наполненность, лица и имена неигровых персонажей, также известных как НПС, а также большую часть растений в игре. В качестве примера можно привести игру Borderlands [1].

Также существует множество аналогов процедурной генерации, которые очень часто используются в настольных играх. К таким аналогам можно отнести составление случайных колод во многих играх, перемешивание между собой составных частей разных колод игровых событий и еще много чего [2]. Однако у таких аналогов есть один значительный недостаток – такая генерация не может быть многоуровневой.

К примеру, нужно составить игровую колоду, в которую будет включен случайный набор карт – это сделать довольно просто. Однако нельзя создать дополнительное условие, при котором в колоду добавится еще один набор карт, если в первый раз была добавлена определенная карта. Если сказать это прямым текстом – игра утратит интригу случайности, ведь игроки уже будут знать, что за карты лежат в колоде. Таким образом, выходит, что в настольных аналогах процедурной генерации нельзя использовать многоуровневые условия, иначе будут утрачены случайность и интрига.

В то же время в компьютерных алгоритмах процедурной генерации игроки не видят, что происходит в алгоритме и выполняются ли какие-то определенные условия. То есть программист, создавший алгоритм, может запрограммировать любое количество условий, с любым количеством уровней этих условий, ведь конечный пользователь в любом случае не будет видеть, какие условия выполняются, а какие нет. Таким образом, получается, что игрок, который у себя на экране видит только конечный результат, не будет знать какие варианты развития алгоритма были выбраны, и к чему это может привести, то есть момент интриги, при котором каждый последующий запуск будет чем-то новым для игрока, сохраняется.

Наглядно увидеть, как происходит многоуровневый выбор в алгоритмах процедурной генерации можно на рис. 1.

Таким образом, при выборе варианта А для дальнейшей генерации уровня в игре, перед алгоритмом появляется выбор – использовать вариант А1 или же А2. В то же время при использовании вариантов Б или В такого выбора не появляется.

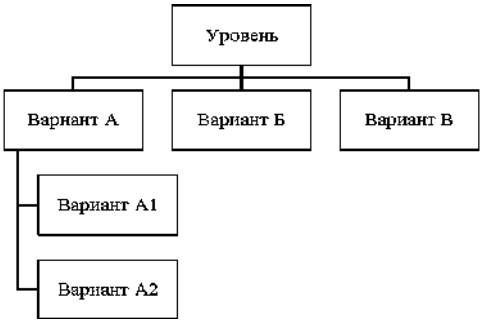


Рис. 1 – Многоуровневый выбор

Однако процедурная генерация полных игровых уровней снискала себе не настолько большую славу. Чаще всего если к алгоритмам процедурной генерации прибегают при создании игровых уровней, то это будут двумерные уровни, которые находятся только в одной плоскости. Генерация трехмерных уровней – это процесс куда более сложный и трудоемкий, чем генерация двумерных уровней, а то и наполнения в игре.

При написании алгоритма процедурной генерации трехмерных виртуальных пространств нужно обратить внимание на очень большое количество факторов. Например, максимально допустимая высота карты, уровень воды на карте, будут ли на карте располагаться поселения людей, выстраивание общей логистики на карты, а также общий вид карты и степень ее реализма.

Цель работы

Главной целью работы является определение алгоритма процедурной генерации, который сможет создавать полноценный игровой уровень, не требующий последующей значительной доработки человеком.

Изложение основного материала

Исследования в области универсальных игровых программ были в первую очередь ориентированы на изучение алгоритмов искусственного интеллекта, и генерация игр в них либо не рассматривалась вовсе, либо была основана на простейших алгоритмах.

Исследованиям процедурной генерации игр посвящены работы К. Брауна [3], В. Хома и Д. Маркса [4], Дж. Тогелиуса [5] и М. Кука [6]. Существующие экспериментальные решения ориентированы на генерацию игр нескольких отдельных жанров.

В работе [7] представлена разработка алгоритма процедурной генерации игрового уровня, который строится из клеток в зависимости от различных вариантов построения уровня – с четкими и свободными границами, формирования самих клеток с возможными вариантами выходов,

ISSN 2079-5459 (print)
ISSN 2413-4295 (online)

построения их набора и реализации проходов между соседними клетками.

Перспективным направлением для дальнейших исследований является улучшение существующих и создание новых алгоритмов генерации с целью повышения качества и разнообразия генерируемых игр.

Для выполнения задачи построения алгоритма процедурной генерации трехмерного игрового уровня можно привести несколько возможных методов [8–11]. Каждый из таких методов имеет свои достоинства и недостатки, руководствуясь которыми программисту следует подбирать нужный метод под каждую ситуацию индивидуально.

Вот пример нескольких алгоритмов процедурной генерации [4–6,11]:

1. Построение карты при помощи наложения текстур на трехмерную модель математической функции.

В данном варианте для создания общего ландшафта игрового уровня создается графическая модель любой трехмерной математической функции. После построения трехмерного графика необходимо нанести текстуры на всю поверхность модели. К несомненным плюсам данного способа можно отнести уникальность каждого уровня, а также максимальную сглаженность отдельных участков. В результате такого метода можно будет получить красивый и плавный ландшафт, однако все еще остается необходимость дальнейшей настройки игрового уровня вручную или с помощью других алгоритмов процедурной генерации.

К плюсам данного метода можно отнести уникальность каждого из построенных уровней, плавность ландшафта и целостность итогового результата.

К минусам же можно отнести обязательную необходимость последующей доработки результата вручную или с помощью других алгоритмов. Также данный метод не подходит для генерации резких ландшафтов, таких как горы или утесы, потому что графики всегда имеют более сглаженные очертания.

Наглядные примеры результатов работы алгоритма с математическими функциями (1) – (2) можно увидеть на рисунках 2-4.

$$f(x,y)=\frac{\sin(x)\sin(y)}{xy}; \qquad (1)$$

$$f(x,y)=xy^3-3y^2+4x^4. \qquad (2)$$

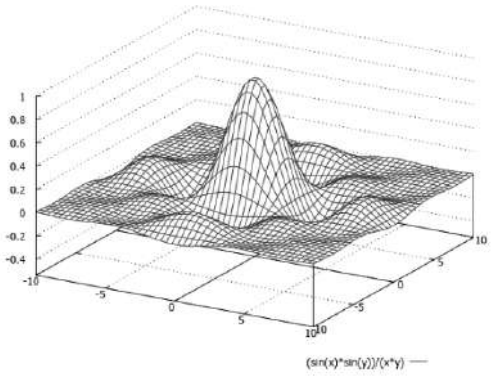


Рис. 2 – Пример формирования рельефа с помощью математических функций (1)

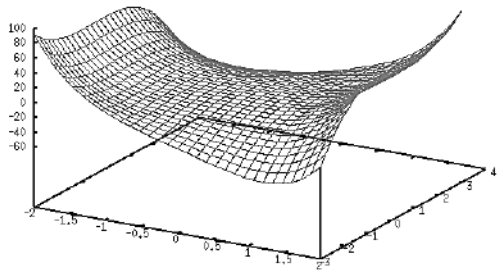


Рис. 3 – Пример формирования рельефа с помощью математических функций (2)

2. Создание набора отдельных взаимозаменяемых участков карты для дальнейшей их комбинации.

Данный метод подразумевает изначальное создание стартового набора разных участков карты. Весь уровень делится на равные сектора (зачастую квадраты), а также вручную создается карта высот уровня. Таким образом появляется набор секторов, карты высот каждого из них, а также их положение на общей карте уровня.

После этого каждый сектор дублируется несколько раз, создавая одинаковое количество копий каждого из участков карты. Затем каждая из копий может быть изменена, при соблюдении карты высот у краев каждого из секторов.

После этого у каждого сектора уровня появляется выборка – набор из разных секторов, с одинаковыми границами, которые являются взаимозаменяемыми. В таком случае очень удобно организовывать многоуровневую генерацию, создавая алгоритм с множеством условий. На каждом из секторов в свою очередь могут быть дополнительные условия, которые будут немного менять их наполнение.

Итогом такого метода является уровень, который при каждой последующей генерации будет создавать выборку из различных секторов, объединяя

их в общую карту уровню. Более наглядно это можно изобразить в виде таблицы:

Таблица 1 – Создание выборки из секторов

Сектор	Значение, которое сектор может принять				
А	1	2	3	4	5
Б	1	2	3	4	5
В	1	2	3	4	5

Таким образом получается, что во время одной генерации алгоритм определит, что сектор А может принять любое значение от 1 до 5, так же, как и сектора Б и В. То есть во время каждой новой генерации алгоритм будет выбирать новое случайное значение, шанс повторения которых крайне мал.

Данный способ позволяет гораздо лучше настроить рамки генерации, чем остальные методы, однако требует большой работы при создании секторов. Помимо этого, он также позволяет использовать многоуровневую генерацию.

К плюсам этого метода можно отнести высокую вариативность сгенерированных уровней, возможность высокого контроля генерации, возможность создания сложных структур и выделяющихся мест, а также ввод нескольких уровней условий.

К однозначным минусам же можно отнести необходимость выполнения большей части работы вручную, а также, при создании маленькой выборки или небольшого количества секторов, относительно небольшую вариативность сгенерированных уровней.

Результат работы такого алгоритма можно увидеть ниже на рис. 4.

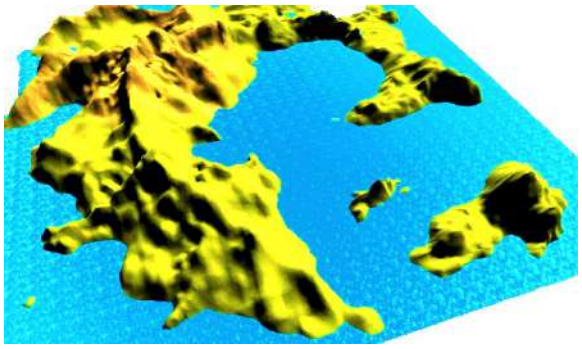


Рис. 4 – Пример сгенерированного уровня путем создания набора отдельных взаимозаменяемых участков карты

3. Разбиение карты на сектора правильной формы для полной генерации карты высот и дальнейшего ее сглаживания.

Данный метод подразумевает изначальное разделение всего игрового уровня на одинаковые сектора правильной формы, такие как квадраты или шестиугольники. Впоследствии эти сектора будут принимать случайное значение в заданном диапазоне, для формирования карты высот данного уровня. Для того, чтобы сгенерированный уровень выглядел правдоподобно, необходимо добавить в алгоритм ряд проверок и зависимостей уровней высот соседних секторов. После генерации карты высот и ее применения необходимо будет сгладить весь уровень, чтобы убрать резкие различия между некоторыми секторами. Лучше всего данный метод используется при генерации относительно ровных местностей, в которых нет необходимости в резкой смене карты высот.

При использовании данного метода можно также увеличить его случайность, используя в качестве карты высот положение не самого сектора, а его вершин. При использовании шестиугольников таких вершин можно использовать 3 или 6.

На рисунке 5 ниже можно увидеть наглядный пример разбиения уровня на сектора из шестиугольников, а также использование в качестве карты высот трех вершин каждого из таких секторов.

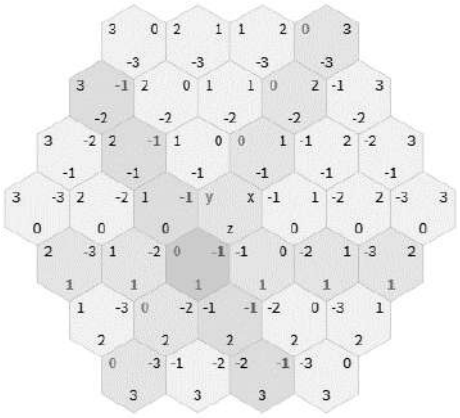


Рис. 5 – Пример разбиения уровня на сектора правильной формы

К плюсам данного метода можно отнести возможность быстрой генерации уровня с относительно гладким ландшафтом, а также приятный общий вид сгенерированного уровня.

В то же время одним из самых значимых минусов является необходимость дальнейшей доработки игрового уровня, ввиду того, что данный метод генерирует только общий ландшафт. Также явным минусом является невозможность генерации резких перепадов высот. Пример такой карты показан ниже на рис. 6.

4. Разбиение карты на сектора случайной неправильной формы для полной генерации карты высот и дальнейшего ее сглаживания.

Данный метод, так же, как и предыдущий, подразумевает изначальное разбиение игрового уровня на сектора. Однако в данном случае эти сектора разбиваются случайным образом на случайные формы. Это позволяет добиться большей хаотичности в методе, а также более резких перепадов и странных форм.

ISSN 2079-5459 (print)
ISSN 2413-4295 (online)

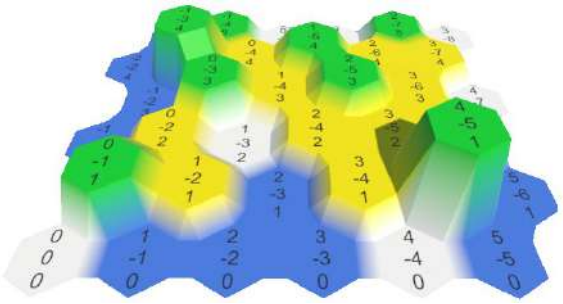


Рис. 6 – Пример сгенерированного уровня с разбиением карты на сектора правильной формы

В отличии от такого же метода с использованием секторов правильной формы, данный метод позволяет полностью сгенерировать уникальный игровой уровень с необычным и местами резким ландшафтом. Такой подход отлично проявляет себя при генерации гористой местности, а также формирования естественной береговой линии.

На рис. 7 можно наглядно увидеть, что подразумевает собой разбиение на случайные сектора.

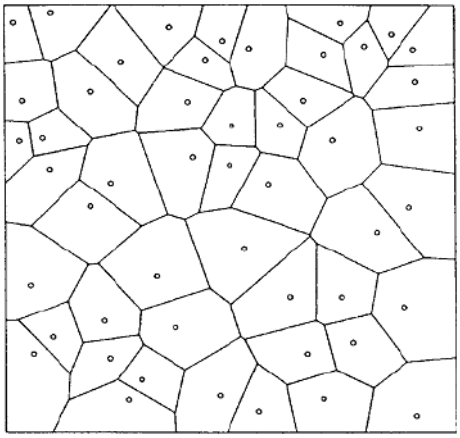


Рис. 7 – Разбиение уровня на сектора случайной неправильной формы

К плюсам данного метода можно отнести более хаотичное разбиение секторов, ведь все сектора получаются неправильных форм, что позволяет добиться уникального построения всего игрового уровня.

Главным минусом этого способа процедурной генерации является необходимость дальнейшей работы с созданным ландшафтом, поэтому больше всего этот метод подходит для генерации двумерных уровней, или карт местности.

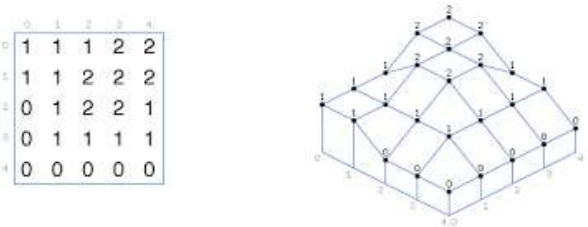
5. Генерация матрицы высот для дальнейшего построения ландшафта.

Данный метод, в первую очередь, подразумевает создание реалистичного ландшафта для игрового уровня. Весь игровой уровень делится на большое количество маленьких квадратных секторов, к каждому из которых в дальнейшем привязывается одна ячейка из массива карты высот.

После разбиения программа генерирует случайную матрицу высот с заданным шагом, что позволяет избежать резких перепадов между соседними секторами. За счет большого количества секторов достигается высокая плавность созданного ландшафта. На рис. 8 можно увидеть наглядное использование матрицы высот.

Такой подход позволяет добиться куда большей вариативности, чем большая часть из описанных ранее, однако так же требует дальнейшей доработки, ввиду генерации только ландшафта игрового уровня.

К плюсам данного метода можно отнести простоту в исполнении, а также большую вариативность и гладкость генерируемого уровня.



1	1	1	2	2
1	1	2	2	2
0	1	2	2	1
0	1	1	1	1
0	0	0	0	0

Рис. 8 – Матрица высот и ее трехмерное изображение

Неоспоримым минусом является необходимость дальнейшей доработки уровня.

Пример работы данного алгоритма можно увидеть ниже, на рис. 9.

Рис. 9 – Пример работы алгоритма с использованием матрицы высот

Обсуждение результатов

Результатом работы каждого из представленных алгоритмов является сгенерированный игровой уровень. Примеры таких уровней можно увидеть на рисунках 4, 6, 9.

После проверки работы каждого из вышеперечисленных алгоритмов процедурной генерации, было определено, что для полноценной генерации трехмерного играбельного уровня лучше

всего использовать метод разделения уровня на сектора и ручной подготовки вариантов каждого из этих секторов.

Данный метод позволяет добиться относительно высокого показателя случайности, позволяя при этом генерировать сложные природные структуры, такие как пещеры или гроты.

Отдельным плюсом такого метода можно назвать возможность создания нескольких условий на разных шагах генерации, что позволяет еще больше повысить показатель случайности и сделать генерацию уровня еще более непредсказуемой для конечного пользователя. Несмотря на необходимость прямого участия разработчика в подготовке стартового набора секторов, этот метод предлагает наибольшую вариацию конечного продукта, а потому на данный момент является фаворитом.

Выводы

Алгоритмы процедурной генерации – это очень вариативный инструмент, который может быть очень полезным в самых разных ситуациях, а особенно при разработке компьютерных игр.

В работе выполнен обзор алгоритмов процедурной генерации, которые могут быть использованы для создания трехмерного игрового уровня. Программные реализации сгенерированных игровых уровней, использующие рассмотренные алгоритмы, позволили выявить их преимущества и недостатки

Был выбран алгоритм процедурной генерации трехмерного игрового уровня – алгоритм разделения уровня на сектора, который позволяет добиться относительно высокого показателя случайности, позволяя при этом генерировать сложные природные структуры, такие как пещеры или гроты с минимальным участием разработчика в конечный процесс формирования ландшафта.

Использование данного алгоритма планируется в первую очередь для разнообразия игрового процесса и создания большего элемента случайности.

Список литературы

1. BoardGameGeek | Gaming Unplugged Since 2000. – URL: https://boardgamegeek.com/17.10.2019. – Заглавие с экрана.
2. Википедия – свободная энциклопедия. – URL: https://ru.wikipedia.org/wiki/Borderlands#Генерация_экипировки 17.10.2019. – Заглавие с экрана.
3. **Browne, C.** Evolutionary Game Design / **C. Browne** – Berlin: Springer, 2011. – 122 p. – doi: 10.1109/TCIAIG.2010.2041928.
4. **Hom, V.** Automatic Design of Balanced Board Games / **V. Hom, J. Marks** // *Proceedings of the Third Artificial Intelligence and Interactive Digital Entertainment Conference*. – 2007. – P. 25-30.
5. **Togelius, J.** An Experiment in Automatic Game Design / **J. Togelius, J. Schmidhuber** // *IEEE Symposium on Computational Intelligence and Games*. – 2008. – P. 111-118.
6. **Cook, M.** Multi-Faceted Evolution Of Simple Arcade Games / **M. Cook, S. Colton** // *IEEE Conference on Computational Intelligence and Games*. – 2011. – P. 289-296. – doi: 10.1109/CIG.2011.6032019.
7. **Kodola, G.N.** About procedural generation of the content and its use at the creation of computer games / **G.N. Kodola, O.R. Denysiuk, M.A. Khrebet** // *Komp'ûterne modelûvannâ: analìz, upravlìnnâ, optimìzaciâ [Computer modeling: analysis, control, optimization]*. – Dnipro: Ukrainian State University of Chemical Technology Publ., 2018. – № 1 (3). – P. 12-17.
8. **Shaker, N.** Procedural Content Generation in Games / **N. Shaker, J. Togelius, J. Nelson Mark** – Springer, 2016. – 218 p. – doi: 10.1007/978-3-319-42716-4.
9. **David, S.** Texturing and Modeling: A Procedural Approach, Third Edition / **David S. Ebert, F. Kenton Musgrave, D. Peachey and etc.** – Morgan Kaufmann, 2002. – 688 p.
10. **Ashlock, D.** Search-Based Procedural Generation of Maze-Like Levels / **D. Ashlock** // *IEEE Transactions on Computational Intelligence and AI in Games*. – 2011. – Vol. 3, No 3. – P. 260–273. – doi: 10.1109/TCIAIG.2011.2138707.
11. **Yannakakis, G. N.** Experience-Driven Procedural Content Generation / **G. N. Yannakakis** // *IEEE Transactions on Affective Computing.* – 2011. – Vol. 2, No 3. – P. 147–161. – doi: 10.1109/T-AFFC.2011.6.

References (transliterated)

1. BoardGameGeek | Gaming Unplugged Since 2000. – Available at: https://boardgamegeek.com/
2. Википедия – свободная энциклопедия. [Wikipedia is a free encyclopedia]. Available at: https://ru.wikipedia.org/wiki/Borderlands#Генерация_экипировки 17.10.2019.
3. **Browne, C.** Evolutionary Game Design Berlin: Springer, 2011, 122, doi: 10.1109/TCIAIG.2010.2041928.
4. **Hom, V., Marks, J.** Automatic Design of Balanced Board Games. *Proceedings of the Third Artificial Intelligence and Interactive Digital Entertainment Conference,* 2007, 25-30.
5. **Togelius, J., Schmidhuber, J.** An Experiment in Automatic Game Design. *IEEE Symposium on Computational Intelligence and Games,* 2008, 111-118.
6. **Cook, M., Colton, S.** Multi-Faceted Evolution Of Simple Arcade Games *IEEE Conference on Computational Intelligence and Games*, 2011, 289-296, doi: 10.1109/CIG.2011.6032019.
7. **Kodola, G. N., Denysiuk, O. R., Khrebet, M. A.** About procedural generation of the content and its use at the creation of computer games. *Komp'ûterne modelûvannâ: analìz, upravlìnnâ, optimìzaciâ [Computer modeling: analysis, control, optimization]*. Dnipro: Ukrainian State University of Chemical Technology Publ., 2018, **1**(3), 12-17.
8. **Shaker, N., Togelius, J., Nelson, Mark J.** Procedural Content Generation in Games. Springer, 2016, 218, doi: 10.1007/978-3-319-42716-4.
9. **Ebert, David S., Kenton, Musgrave, F., Peachey, D. and et al.** Texturing and Modeling: A Procedural Approach, Third Edition. Morgan Kaufmann, 2002, 688.
10. **Ashlock, D.** Search-Based Procedural Generation of Maze-Like Levels. *IEEE Transactions on Computational Intelligence and AI in Games*, 2011, **3** (3), 260-273, doi: 10.1109/TCIAIG.2011.2138707.

ISSN 2079-5459 (print)
ISSN 2413-4295 (online)

11. **Yannakakis, G. N.** Experience-Driven Procedural Content Generation *IEEE Transactions on Affective Computing,* 2011, **2(**3), 147–161, doi: 10.1109/T-AFFC.2011.6.

Сведения об авторах (About authors)

Хребет Михаил Александрович – Государственное высшее учебное заведение «Украинский государственный химико-технологический университет», студент кафедры Информационных систем; г. Днепр, Украина; e-mail: khrebet.michael@gmail.com.

Michael Khrebet – Ukrainian State Chemical-Technological University, Student at the Department of Information Systems, Dnipro, Ukraine; e-mail: khrebet.michael@gmail.com.

Кодола Галина Николаевна – Государственное высшее учебное заведение «Украинский государственный химико-технологический университет», преподаватель кафедры Информационных систем; г. Днепр, Украина; ORCID: 0000-0001-9403-1462; e-mail: galina_kodola@udhtu.edu.ua.

Galyna Kodola – Ukrainian State Chemical-Technological University, Lecturer at the Department of Information Systems, Dnipro, Ukraine; ORCID: 0000-0001-9403-1462; e-mail: galina_kodola@udhtu.edu.ua.

Сербулова Инна Валерьевна – Государственное высшее учебное заведение «Украинский государственный химико-технологический университет», старший преподаватель кафедры Информационных систем; г. Днепр, Украина; e-mail: innasrb20@gmail.com.

Inna Serbulova – Ukrainian State Chemical-Technological University, Senior Lecturer at the Department of Information Systems, Dnipro, Ukraine; e-mail: innasrb20@gmail.com.

Пожалуйста, ссылайтесь на эту статью следующим образом:

Хребет, М. А. О процедурной генерации трехмерных виртуальных пространств / **М. А. Хребет, Г. Н. Кодола, И. В. Сербулова** // *Вестник НТУ «ХПИ», Серия: Новые решения в современных технологиях*. – Харьков: НТУ «ХПИ». – 2019. – № 2. – С. 54-60. – doi:10.20998/2413-4295.2019.02.08.

Please cite this article as:

Khrebet, M., Kodola, G., Serbulova, I. About three-dimensional procedure virtual spaces. *Bulletin of NTU "KhPI".Series: New solutions in modern technologies*. – Kharkiv: NTU "KhPI", 2019, **2**, 54-60, doi:10.20998/2413-4295.2019.02.08.

Будь ласка, посилайтесь на цю статтю наступним чином:

Хребет, М. А. Про процедурну генерацію трьохвимірних віртуальних просторів / **М. А. Хребет, Г. М. Кодола, І. В. Сербулова** // *Вісник НТУ «ХПІ», Серія: Нові рішення в сучасних технологіях*. – Харків: НТУ «ХПІ». – 2019. – № 2. – С. 54-60. – doi:10.20998/2413-4295.2019.02.08.

АНОТАЦІЯ *У сучасних реаліях процедурна генерація контенту є одним з найбільш затребуваних і перспективних напрямків у середовищі розробки ігор. У наш час якісно виконаний та добре продуманий алгоритм процедурної генерації може взяти на себе більшу частину роботи зі створення унікального контенту під час розробки гри, або ж під час самого ігрового процесу. За допомогою процедурної генерації можна у найкоротші терміни створити набір унікальних об'єктів. Існує велика різноманітність усіляких алгоритмів процедурної генерації, кожен з яких має свої плюси і мінуси, й краще за все підходить до певного типу завдань. У даній статті розглянуто алгоритми, які можуть бути використані для створення тривимірного ігрового рівня, що дозволяє значно скоротити час на роботу зі створення таких рівнів, а також дозволить підвищити унікальність кожного згенерованого рівня. Як приклад, для ознайомлення в тексті статті викладено п'ять існуючих алгоритмів процедурної генерації, які так чи інакше дозволяють згенерувати тривимірний ігровий рівень. Кінцевою метою є знаходження оптимального алгоритму процедурної генерації, який буде задовольняти головному критерію відбору – створення повноцінного ігрового рівня з ландшафтом і об'єктами, який не потребує подальшого доопрацювання. На даний момент існує не так багато ігор, які успішно використовують алгоритми процедурної генерації при створенні повноцінного тривимірного ігрового світу – до найбільш відомим можна віднести Minecraft, No Man's Sky і Deep Rock Galactic. Підсумковим результатом дослідження стало знаходження найбільш прийнятного алгоритму процедурної генерації, за допомогою якого можна в реальному часі генерувати унікальні ігрові рівні. В якості основної вимоги до алгоритму виступала відсутність необхідності подальшої роботи зі створеним рівнем, тому що використання даного алгоритму планується у першу чергу для різноманітності ігрового процесу та створення більшого елемента випадковості. У результаті фаворитом виявився алгоритм поділу рівня на сектора, зважаючи на виконання ним основної умови, а також можливості програмування декількох умов.*

Ключові слова: *процедурна генерація контенту; алгоритм; рівень гри; тривимірна генерація; комп'ютерна гра; ігровий ландшафт*

Поступила (received) 29.09.2019