УДК 004.9

ПРОЦЕДУРНА ГЕНЕРАЦІЯ ВОКСЕЛЬНИХ ЛАНДШАФТІВ НА ОСНОВІ ІЗОПОВЕРХОНЬ ІЗ ЗАСТОСУВАННЯМ БАГАТОПОТОКОВОСТІ

Аушева Н.М., д.т.н.,
nataauscheva@gmail.com, ORCID: 0000-0003-0816-2971
Чорний В. О.,
vladchornuy1@gmail.com , ORCID: 0009-0006-4078-2838
Кардашов О.В. , аспірант*,
alexanderkardashov3@gmail.com, ORCID: 0000-0003-1767-7846
Онисько А.І., к.в.н.,
kw_fedun@ukr.net, ORCID: 0000-0001-7178-1471
Тарнавський Ю.А., к.т.н.,
tarnavski.yu@gmail.com , ORCID: 0000-0003-3226-3107
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського» (Україна)

Швидкий розвиток технологій вимагає вдосконалювати підходи до розробки програмного забезпечення. При створенні реального середовища у симуляторах і тренажерах важливою частиною є моделювання рельєфу місцевості або ландшафту. Технологія «unreal» дозволяє створювати власні рушії для розв'язання прикладних завдань, тому дослідження останніх розробок, а саме Unreal Engine 5, для генерації тривимірних моделей ландшафту в режимі реального часу є актуальною задачею.

В статті наводиться аналіз існуючих проблем при моделюванні рельєфів місцевості в режимі реального часу. Ландшафт будується у вигляді воксельної моделі, яка формується на основі значень ізоповерхні. Значення ізоповерхні визначаються скалярним полем. Для візуалізації моделі застосовується кубічне відображення. Для позбавлення від надлишку інформації пропонується скористатися жадібним алгоритмом, який призводить до генерації полігональної сітки зі зменшеною кількістю геометрії. Перехід від воксельного ландшафту до полігонів здійснюється двома шляхами на основі алгоритмів поверхневих сіток та маршируючих кубів. Наводяться псевдокоди алгоритмів. Для позбавлення від затримки головного потоку пропонується розділити завдання по візуалізації за різними потоками. Визначено задачі, які можна виконувати на головному і на додатковому потоках. Воксельний ландшафт будується на основі шматків поверхні, що зменшує навантаженість на відеокарту.

Розроблено програмне забезпечення та проведено його тестування з однаковими параметрами налаштування для різних алгоритмів. Наведено

* Науковий керівник – д.т.н., професор Аушева Н.М.

результати тестування та показано зменшення затримки головного потоку при застосуванні принципу багатопотоковості. Для процедурної генерації ландшафту застосована відкрита бібліотека FastNoiseLite, для створення зручного інтерфейсу налаштувань використовується система графічного скриптингу Visual Scripting Blueprint, для розробки плагіну C++ та Visual Studio 2019.

Ключові слова: тривимірна модель, воксельний рушій, технології Unreal Engine, багатопотоковість, алгоритм маршируючих кубів, алгоритм поверхневих сіток.

Постановка проблеми. Однією із сфер застосування комп'ютерної графіки є візуалізація зображень у симуляторах і тренажерах. За допомогою комп'ютерно-механічних пристроїв моделюються ситуації які наближені до реального середовища та дозволяють одержувати навички для керування складними технічними пристроями [1]. Важливою частиною будь-якого тренажера є створення тривимірної моделі навколишнього середовища, основною частиною якої є ландшафт. Ефективних методів та алгоритмів візуалізації ландшафтів у реальному часі наразі існує багато [2,3,4], але у зв'язку з швидким розвитком нових технологій виникає потреба у адаптації та вдосконалення підходів до нових розробок, створення інструментів для розробки та налаштування моделей, оптимізації процесів візуалізації для виведення об'єктів в режимі реального часу. Зокрема, застосувати технологію «unreal», яка надає досить вагомі переваги, а саме, кросплатформність, спрощений та швидкий процес розробки, зниження вартості розробки на всіх етапах. Для створення складних рельєфів, інтерактивної та динамічної їх зміни у реальному часі доцільно застосовувати воксельний рушій [5], який можна використовувати для побудови, дослідження та аналізу скалярних середовищ чи областей значень, що призначені для моделювання дискретних об'єктів. Тому вирішення задачі створення рельєфу місцевості на основі нових технологій Unreal Engine зі створенням зручного інтерфейсу користувача для генерації тривимірних моделей в режимі реального часу є актуальною проблемою.

Аналіз останніх досліджень і публікацій. Тетчером Ульріхом [2] запропоновано представляти великі обсяги даних про місцевість на основі фрагментованих LOD рівнів, що дозволяє відображати їх в режимі реального часу. Авторами роботи [3] проводяться дослідження стосовно ефективного введення та виведення даних, надається спосіб розміщення даних для прискорення візуалізації. Дослідниками з НТУ «ХПІ» [4] пропонується побудова моделі ландшафту за супутниковими знімками і полем висот. Запропонований метод дозволяє отримувати різні типи ландшафтів у будь-яких природно-кліматичних умовах та надає можливість для синтезу рослинного покрову. Для візуалізації природних рельєфів в реальному часі в роботі [6] надається структура GPU, а також

інформація стосовно побудови екосистеми з мінімальною кількістю попередньо збережених даних. Веб-візуалізацію в режимі реального часу повного природного середовища пропонують автори публікації [7]. Модель рослинності вдосконалює процедурне створення екосистем та моделювання ландшафту.

Формулювання цілей статті. Метою роботи є створення програмного забезпечення для процедурного формування воксельних ландшафтів на основі ізоповерхонь з застосуванням принципу багатопотоковості на основі технології UE5.

Основна частина. Рушій Unreal Engine 5 (UE5) було представлено нещодавно з новими поліпшеннями анімації, звуку, технічної та графічної складової [8, 9]. Технології «unreal» дозволяють використовувати їх як основу для створення власного рушія для розв'язання прикладних завдань. Наразі кількість рішень для нової версії є дуже обмеженим (офіційний магазин компанії Epic Games). Для створення зручного інтерфейсу налаштувань використовується система графічного скриптингу блюпринт Visual Scripting Blueprint, а для розробки плагіну C++ та Visual Studio 2019.

Будемо моделювати воксельний ландшафт на основі значень ізоповерхні, яка задається скалярним полем, в кожній точці цієї поверхні задається найменша відстань від цієї точки до поверхні, що проходить через всю тривимірну область. На рис. 1. показана область щільності поверхні у тривимірному просторі.

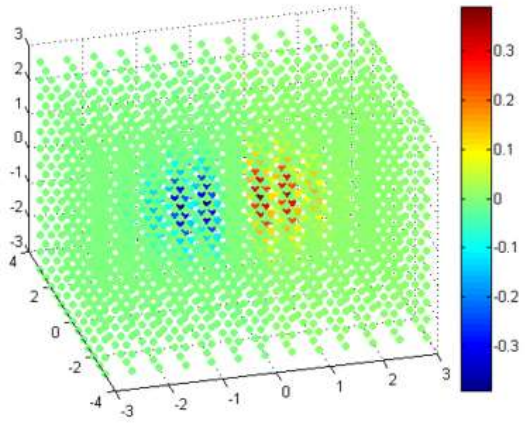

Рис.1. Область щільності поверхні у тривимірному просторі

Така ізоповерхня задається масивом, який зберігає значення відстані в діапазоні [-1,1], якщо значення «-1» - точка находиться в середині поверхні, «1» - точка за межами поверхні. При об'єднанні 8 сусідніх точок утворюються комірки сітки. Для побудови та відображення полігональної поверхні алгоритми використовують інформацію про перетин ізоповерхні ребрами кубів.

Для побудови воксельної моделі відбувається прохід по кожній комірці сітки та визначається значення ізоповерхні в кожній із 8 вершин, якщо вершина має від’ємне значення, будується куб з 12 трикутників. Побудова вокселів в середині моделі не доцільна, тому відображаються лише вокселі, що знаходяться на поверхні. Для відображення лише тих граней, які видимі користувачеві застосовується метод оптимізації Face culling. На рис.2. наведено приклад побудованого ландшафту за допомогою кубічного відображення.

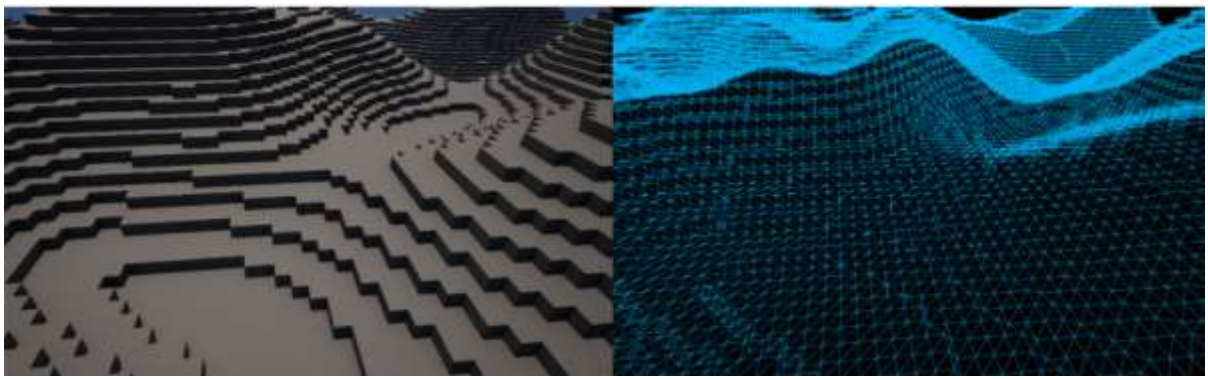

Рис.2. Воксельний ландшафт на основі кубічного відображення

Кубічна воксельна модель має надлишок інформації, тобто багато спільних рис мають грані моделі. Для оптимізації цього процесу можна скористатися жадібним алгоритмом [10], який дозволяє об’єднати вокселі зі спільними ознаками в один чотирикутник. Задача зводиться до генерації полігональної сітки для двовимірного поперечного перерізу, тобто виконується декомпозиція задачі та перехід із дво- до тривимірного простору. На рис.3. наведено псевдокод жадібного алгоритму.

```
Пройти в напрямку кожної із граней куба
    Визначити таблицю злиття;
    Пройти по кожному вокселю двовимірного перерізу
        Знайти початковий воксель;
        Пройти в напрямку X поки не знайдеться воксель
        відмінний від початкового;
        Пройти в напрямку Y, з підтримкою знайденої
        ширини по X, поки не знайдеться воксель
        відмінний від початкового;
        Побудувати грань на основі початкового
        значення та кінцевого;
        Проініціалізувати таблицю злиття;
```

Рис.3. Псевдокод жадібного алгоритму у воксельній моделі

На рис. 4 наведено воксельний ландшафт, який було побудовано на основі розробленого алгоритму, за допомогою якого значно зменшилась кількість геометрії.

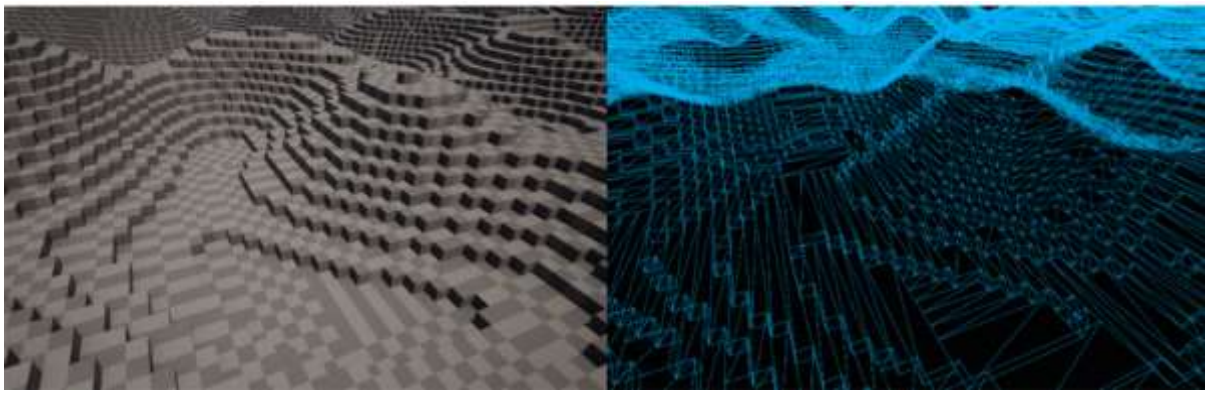

Рис.3. Оптимізований воксельний ландшафт на основі жадібного алгоритму

Приклад побудованого шматка воксельного ландшафту на площині на основі жадібного алгоритму наведено на рис.4.

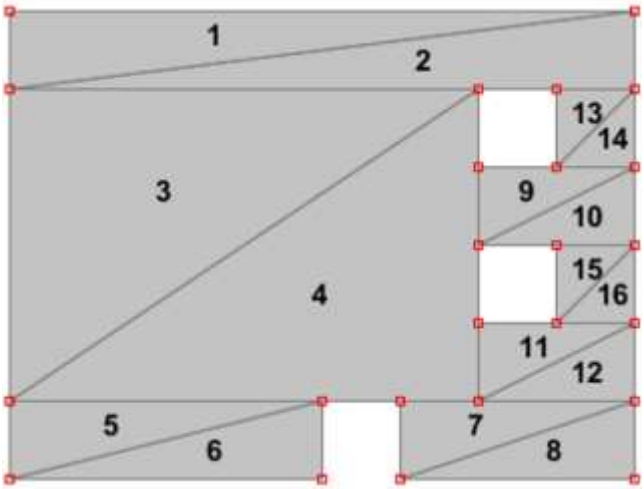

Рис.4. Шматок воксельного ландшафту на основі жадібного алгоритму

Для переходу від воксельного ландшафту до полігональної сітки застосовується алгоритм поверхневих сіток [11]. При використанні класичного алгоритму визначаються поверхневі вокселі на основі інтерполяції центрів шести сусідніх вокселів. Найбільш ефективним є процес інтерполяції на основі точок перетину граней з ізоповерхнею. На рис.5. наведено псевдокод переходу від вокселів до полігонів. Воксельний ландшафт, що побудований на основі методу поверхневих сіток представлено на рис.6. Одним з розповсюджених алгоритмів переходу до воксельних гладких поверхонь є алгоритм кубів, що марширують [12]. Цей алгоритм є стандартним підходом при переході від вокселів (рис.7).

Для процедурної генерації ландшафту та отримання значень ізоповерхні використана відкрита бібліотека генерації шуму FastNoise Lite з відкритим кодом, широким спектром налаштувань та алгоритмів, яка дозволяє досягати високої продуктивності.

```
Пройти по кожній точці тривимірного масиву
    Визначити 7 сусідніх вершин;
    Якщо вершини вокселя мають змінні значення;
        Записати координати вокселя;
Пройти по всіх знайдених вокселях
    Пройти по всім ребрам
        Якщо ребро перетинає поверхню
            Визначити точку перетину;
    Інтерполювати цент вокселя відповідно до точок
    перетину ребер ізоповерхнею;
Пройти по всіх знайдених вокселях
    Пройти по трьом ребрам
        Якщо ребро перетинає поверхню
            Знайти центральні точки сусідніх вокселів
            для яких ребро є спільним;
            Побудувати грань на основі чотирьох
            центральних точок;
```

Рис.5. Псевдокод алгоритму переходу від воксельної моделі до полігональної

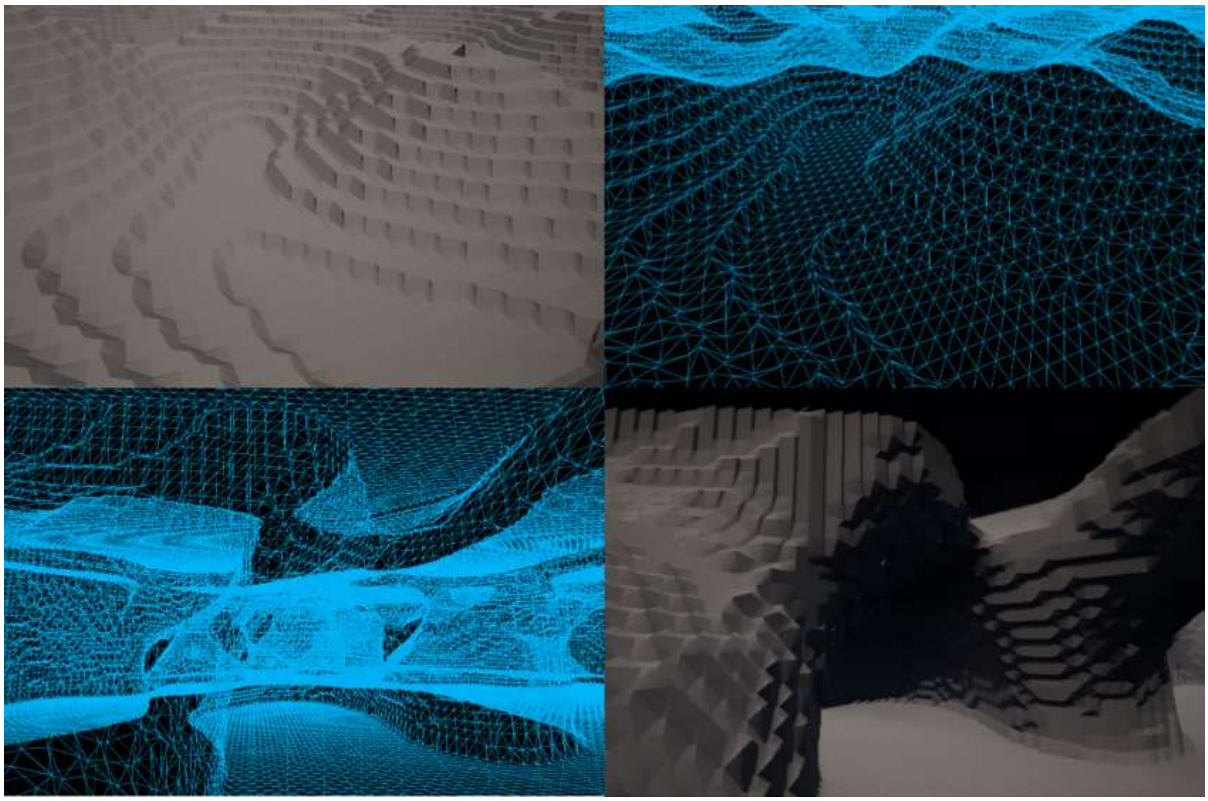

Рис.6. Воксельний ландшафт, що побудований на основі методу поверхневих сіток

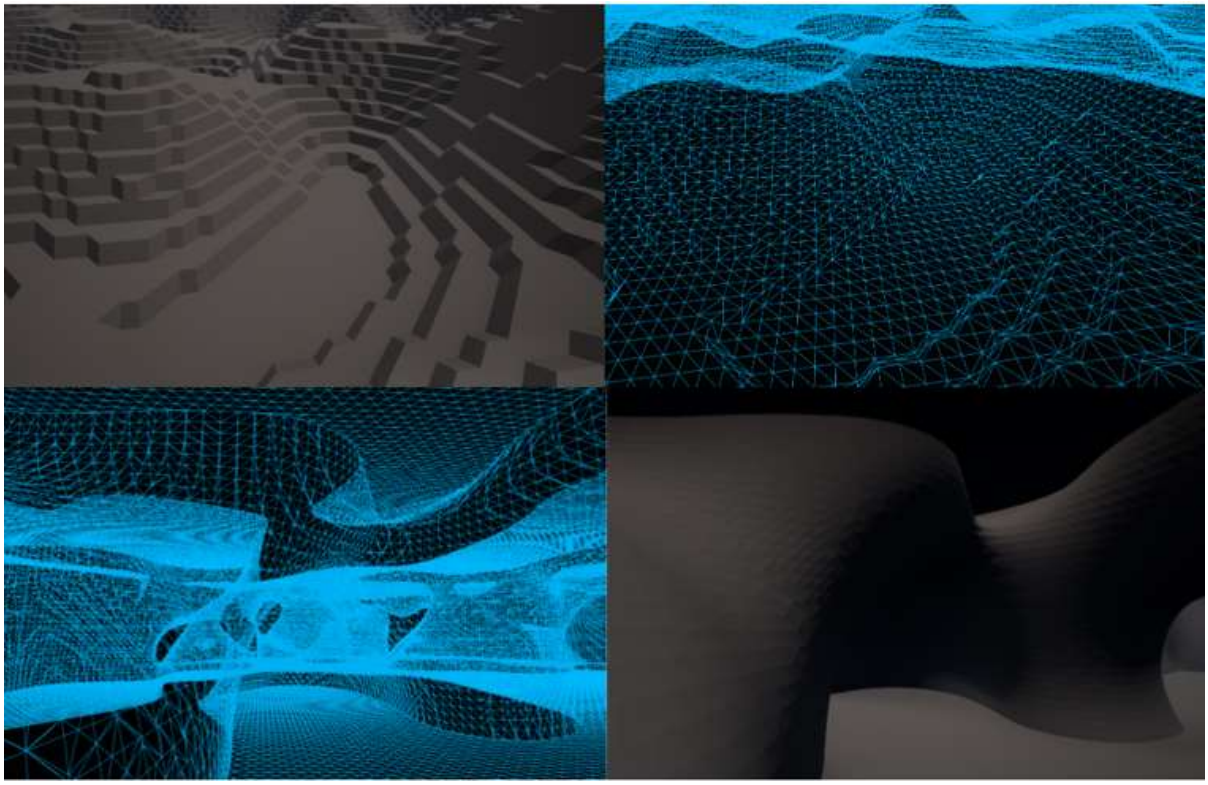

Рис.7. Воксельний ландшафт, що побудований на основі методу кубів, що марширують

Для відображення рельєфу у режимі реального часу застосуємо підхід багатопотоковості, а саме систему асинхронних завдань [13]. Розділимо завдання візуалізації поверхні по різним потокам: перенесемо визначення значень ізоповерхні (дані про вертекси, нормалі, трикутники) та побудову полігональної сітки у додатковий потік. В цьому випадку збільшується частота кадрів та не виконуються затримки головного потоку та стрибки продуктивності (рис.8).

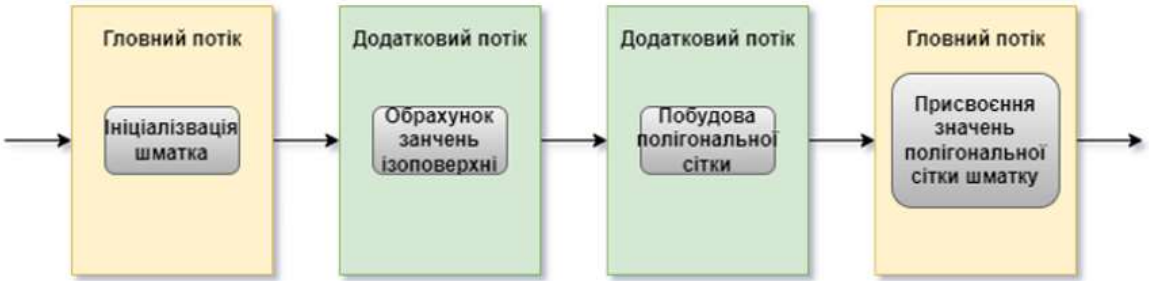

Рис.8. Створення багатопотокового режиму для шматка поверхні

Воксельний рушій було розроблено в середовищі Visual Studio 2019 на основі C++ з застосуванням бібліотеки шумів FastNoiseLite як окрему підсистему UE5. Для тестування були зроблені наступні налаштування: розмір шматка поверхні – 32 вокселя; встановлено тривимірний стандартний шум Перліна, початкове значення для шуму – 1337, кількість

октав -1, частота шуму – 0.03. Результати тестування наведено у таблиці 1. Для порівняння методів побудови воксельних моделей було застосовано 64 шматка поверхні. Результати тестування наведено у таблиці 2.

Таблиця 1.

	Кубічне відображення	Кубічне відображення поверхневих вокселів	Використання оптимізації Face Culling
Кількість трикутників	915588	117828	18492
Кількість вертексів	610392	78552	12328
Кількість нормалей	610392	78552	12328
Кількість UV	610392	78552	12328
Максимальна затримка головного потоку (мс)	10869	1191	558
Максимальна затримка головного потоку з багатопотоковістю (мс)	8843	931	228

Таблиця 2.

	Кубічне відображення з Face Culling	Жадібний алгоритм	Алгоритм поверхневих сіток	Алгоритм кубів, що марширують
Кількість трикутників	18492	6690	18492	17808
Кількість вертексів	12328	4460	12328	17808
Кількість нормалей	12328	4460	12328	17808
Кількість UV	12328	4460	12328	17808
Максимальна затримка головного потоку (мс)	1842	2357	2050	1824
Максимальна затримка головного потоку з багатопотоковістю (мс)	715	461	811	1080

Висновки. На основі аналізу існуючих технологій та методів створення інтерактивних рельєфів місцевості було обґрунтовано використання технології «unreal» для реалізації воксельного рушія, алгоритмів поверхневих сіток та алгоритму кубів, що марширують. Для використання рушія у режимі реального часу запропоновано застосувати принцип розділення завдань на головний і додатковий потоки. Проведено тестування створеного програмного забезпечення та показано зменшення затримки головного потоку при застосуванні принципу багатопотоковості.

Література

1. Sokolowski John A., Banks Catherine M. Principles of modeling and simulation: a multidisciplinary approach. Hoboken, N.J.: John Wiley, 2009. 260 p.

2. Thatcher U. Rendering Massive Terrains using Chunked Level of Detail Control. *ACM SIGGRAPH 2002*, volume Course 35, 2002.
3. Lindstrom P. Visualization of Large Terrains Made Easy/ P. Lindstrom, P. V. Pascucci. *IEEE Visualization*, October 2001. P. 363 – 370.
4. Качанов П.А., Зуев А.А. Автоматизированный синтез модели поверхности ландшафта по спутниковым снимкам. *Восточно-европейский журнал передовых технологий.* 2014. Том. 6, № 3 (72). С.10-15.
5. Cohen-Or D., Kaufman F.Fundamentals of Surface Voxelization. *Graphical Models and Image Processing.*Volume 57, Issue 6, November 1995, P. 453-461.
6. Amara Y., Meunier S., Marsault X. A GPU Framework for the Visualization and On-the-Fly Amplification of Real Terrains. *International Symposium on Visual Computing*, 2007. P 586-597.
7. Onrust B., Bidarra R., Rooseboom R., Koppel J. Procedural generation and interactive web visualization of natural environments. *Web3D '15: Proceedings of the 20th International Conference on 3D Web Technology*. 2015.P. 133–141.
8. Epic Games announces Unreal Engine 5 with first PS5 footage. *GamesIndustry.biz*. URL: https://www.gamesindustry.biz/epic-games-announces-unreal-engine-5-with-first-ps5-footage (дата звернення 26.05.2023).
9. O'Connor A. Unreal Engine 5 revealed with a Tomb Raider-y tech demo. *Rock, Paper, Shotgun*. URL: https://www.rockpapershotgun.com/unreal-engine-5-revealed-with-a-tomb-raider-y-tech-demo (дата звернення 26.05.2023).
10. Consume everything - how greedy meshing works : веб-сайт. URL: https://devforum.roblox.com/t/consume-everything-how-greedy-meshing-works/452717 (дата звернення 26.05.2023).
11. Understanding Surface Nets: веб-сайт. URL: https://cerbion.net/blog/understanding-surface-nets/ (дата звернення 26.05.2023).
12. Lorensen W.E., Cline H.E. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. *ACM SIGGRAPH Computer Graphics.* № 21 (4). P.163–169.
13. AsyncTask Function: веб-сайт. URL: https://unrealcommunity.wiki/asynctask-function-mfdnmfca (дата звернення 26.05.2023).

PROCEDURAL GENERATION OF VOXEL LANDSCAPES BASED ON ISOSURFACES USING MULTITHREADING

Natalia Ausheva, Vladyslav Chornyi, Oleksandr Kardashov, Andrii Onysko, Yuri Tarnavski

The rapid development of technologies requires improving approaches to software development. When creating a real environment in simulators and trainers, an important part is modeling the topography of the terrain or landscape. The "unreal" technology allows to create custom engines for solving applied problems, so the study of the latest developments, namely Unreal Engine 5, for the generation of three-dimensional models of the landscape in real time is an urgent task.

The article provides an analysis of existing problems in real-time terrain relief modeling. The landscape is built in the form of a voxel model, which is formed on the basis of isosurface values. The values of the isosurface are determined by the scalar field. Cubic mapping is used to visualize the model. To get rid of excess information, it is suggested to use a greedy algorithm, which leads to the generation of a polygonal mesh with a reduced amount of geometry. The transition from the voxel landscape to the polygons is carried out in two ways based on the algorithms of surface meshes and marching cubes. The pseudocodes of the algorithms are given. In order to get rid of the main thread delay, it is suggested to divide the rendering tasks into different threads. Tasks that can be performed on the main and on additional streams have been defined. The voxel landscape is built on the basis of pieces of the surface, which reduces the load on the video card.

The software was developed and tested with the same settings for different algorithms. The test results are presented and the delay reduction of the main stream when applying the principle of multi-threading is shown. The FastNoiseLite open library is used for the procedural generation of the landscape, the Visual Scripting Blueprint graphic scripting system is used for creating a convenient settings interface, C++ and Visual Studio 2019 are used for the development of the plugin.

Keywords: three-dimensional model, voxel engine, Unreal Engine technologies, multithreading, marching cube algorithm, surface mesh algorithm.

References

1. Sokolowski John A., Banks Catherine M. Principles of modeling and

simulation: a multidisciplinary approach. Hoboken, N.J.: John Wiley, 2009. 260 p.
2. Thatcher U. Rendering Massive Terrains using Chunked Level of Detail Control. ACM SIGGRAPH 2002, volume Course 35, 2002.
3. Lindstrom P. Visualization of Large Terrains Made Easy/ P. Lindstrom, P. V. Pascucci. IEEE Visualization, October 2001. P. 363 – 370.
4. Kachanov P.A. Zuev A.A. (2014) Automated synthesis of a landscape surface model from satellite images. Eastern-European Journal of Enterprise Technologies. Том. 6, № 3 (72).10-15 [in Ukrainian]
5. Cohen-Or D., Kaufman F. (1995) Fundamentals of Surface Voxelization. *Graphical Models and Image Processing.*Volume 57, Issue 6, November 1995, 453-461.
6. Amara Y., Meunier S., Marsault X. A. (2007) GPU Framework for the Visualization and On-the-Fly Amplification of Real Terrains. *International Symposium on Visual Computing*, 586-597pp.
7. Onrust B., Bidarra R., Rooseboom R., Koppel J. Procedural generation and interactive web visualization of natural environments. *Web3D '15: Proceedings of the 20th International Conference on 3D Web Technology*. 2015.133–141pp.
8. Epic Games announces Unreal Engine 5 with first PS5 footage. *GamesIndustry.biz*. Retrieved from: https://www.gamesindustry.biz/epic-games-announces-unreal-engine-5-with-first-ps5-footage.
9. O'Connor A. Unreal Engine 5 revealed with a Tomb Raider-y tech demo. *Rock, Paper, Shotgun*. Retrieved from: https://www.rockpapershotgun.com/unreal-engine-5-revealed-with-a-tomb-raider-y-tech-demo.
10. Consume everything – how greedy meshing works : веб-сайт. URL: https://devforum.roblox.com/t/consume-everything-how-greedy-meshing-works/452717 .
11. Understanding Surface Nets: веб-сайт. Retrieved from: https://cerbion.net/blog/understanding-surface-nets/ .
12. Lorensen W.E., Cline H.E. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. *ACM SIGGRAPH Computer Graphics.* № 21 (4). 163–169.
13. AsyncTask Function: веб-сайт. Retrieved from: https://unrealcommunity.wiki/asynctask-function-mfdnmfca .