A hybrid terrain data compression method in unity for deployment on resource-limited devices

Luu Van Sang[1*], Vu Hoang Minh[1], Tran Binh Minh[1],
Nguyen Van Trung[2], Nguyen Anh Tuan[3], Dang Duc Trinh[4]

[1]Institute of Information Technology, Academy of Military Science and Technology, 17 Hoang Sam, Cau Giay, Hanoi, Vietnam;
[2]Military Academy of Logistics, Ngoc Thuy, Long Bien, Hanoi, Vietnam;
[3]Faculty of Basic Science, Air Defence-Air Force Academy, Kim Son, Son Tay, Ha Noi, Vietnam;
[4]Faculty of Maths-Informatics, Military Medical University, 160 Phung Hung, Ha Dong, Hanoi, Vietnam.
[*]Corresponding author: luusangcntt@gmail.com
Received 10 Nov. 2023; Revised 20 Jan. 2024; Accepted 14 Mar. 2024; Published 22 Apr. 2024.
DOI: https://doi.org/10.54939/1859-1043.j.mst.94.2024.130-138

ABSTRACT

With geographic information systems in general and simulation systems in particular, terrain data takes up most of the hard drive space and is often deployed on data servers. The higher the resolution of the terrain data, the more detailed it is, the larger the space occupied on the hard drive. If terrain data needs to be deployed offline on resource-limited devices such as minicomputers, it will face many difficulties due to hard drive space limitations. Terrain data compression is a solution that reduces terrain capacity to overcome that problem. This article presents an efficient hybrid approach based on Brotli and LZ4 compression algorithms to compress terrain data for deployment on resource-limited devices. Experimental results show that the proposed method significantly reduces the volume of terrain data compared to using each component algorithm independently while still ensuring quality.

Keywords: Terrain data; Data compression; Unity; Resource-limited devices.

1. INTRODUCTION

The explosion of digital information leads to the daily growth of data stored in storage devices. As a result, data storage and transmission are likely to increase enormously [1]. According to Parkinson's First Law [2], the necessity of storage and transmission increases at least twice as storage and transmission capacities increase. The data growth rate is much higher than the technology growth rate. So, it is difficult to address the above-mentioned issue of handling extensive data in terms of storage and transmission. To overcome this challenge, an alternative concept called data compression (DC) has been presented in the literature.

DC aims to reduce the data size by transforming the original data into a compact form. In general, data can be compressed by eliminating data redundancy and irrelevancy. Modeling and coding are the two levels to compress data: firstly, the data will be analyzed for any redundant information and extracted to develop a model; secondly, the difference between the modeled and actual data (called residual) is computed and coded by an encoding technique. Many advantages can be gained from compressing data, such as saving data storage and saving bandwidth in the transmission process [3]. Nowadays, DC are essential in most applications such as satellite imagery, geographical information systems (GIS), simulation, etc.

Resource-limited devices (RLDs) have limited processing speed and storage capacity, but their compact size makes them suitable for deployment in highly mobile systems. Typically, when deploying systems that include RLDs, big data is often deployed on servers. However, due to the requirements of practical problems, in some cases, like offline deployment models, RLDs still have to store a large amount of data. Therefore, extensive data is usually compressed before being deployed on RLDs to ensure storage capacity.

In this study, we proposed a hybrid method for compressing and deploying terrain data on RLDs based on a suitable combination of the LZ4 and the Brotli compression algorithms. The rest of this paper is organized as follows: Backgrounds are introduced in section 2. The problem and related works are described in section 3. Section 4 presents experiments, results, and discussion. Finally, the conclusion and future work are given in section 5.

2. BACKGROUNDS

2.1. Data Compression

DC is the process of converting an input data (the original raw data or the source stream) into another data (the compressed data or the compressed stream) that has a smaller size [4]. Because most files have lots of redundancy, they need to be compressed to save space when storing and time when transmitting.

a) Data Compression Classification

Based on the integrity of information, DC methods can be divided into [4]: (i) Lossless method (allows the original data to be rearranged from the compressed data, lossless DC is used in various applications, such as Zip and GZip formats); (ii) Lossy method (the process of DC and then decompression where the resulting data is not the same as the original data, but enough and can still be used as needed). Based on the purpose, DC methods can be divided into [4]: (ii) Backup/archive compression (the entire data must be decompressed before use); (ii) File stream compression (only requires data to be partially decompressed before use).

b) Data Compression Techniques

Some typical DC techniques include [5]:

- Run-Length Encoding (RLE): Replaces runs of two or more of the same character with a number that represents the length of the run, followed by the original character;

- Entropy Coding (RC): Entropy in data compression means the smallest number of bits needed, on average, to represent a symbol or literal. A basic entropy coder combines a statistical model and a coder. The input file is parsed and used to generate a statistical model that consists of the probabilities of a given symbol appearing. Then, the coder will use the model to determine what bit-or-bytecodes to assign to each symbol such that the most common symbols have the shortest codes and vice versa for the least common symbols;

- Shannon-Fano Coding (SFC): Generating a binary tree to represent the probabilities of each symbol occurring. The symbols are ordered such that the most frequent symbols appear at the top of the tree and the least likely symbols appear at the bottom;

- Huffman Coding: A variant of entropy coding that works very similarly to SFC, but the binary tree is built from the top down to generate an optimal result;

- Arithmetic Coding (AC): Transforming the input data into a single rational number between 0 and 1 by changing the base and assigning a single value to each unique symbol from 0 up to the base. Then, it is further transformed into a fixed-point binary number, which is the encoded result. The value can be decoded by changing the base from binary back to the original base and replacing the values with the symbols they correspond to.

2.2. Typical Data Compression Algorithms

This subsection presents typical DC algorithms directly related to the paper.

a) LZ77 compression algorithm [6]

The LZ77 (also known as LZ1) is a lossless compression algorithm developed by Abraham Lempel and Jacob Ziv in 1977. It is used to determine how to reduce the size of data by replacing redundant information with metadata. Sections of the identical data are replaced by a small amount

of metadata that indicates how to expand those sections again. The encoding algorithm is used to take that combination of data and metadata and serialize it into a stream of bytes that can later be decompressed. In LZ77, the sliding window is usually divided into a search buffer (a dictionary) and a look-ahead buffer. The dictionary includes the characters that have been encoded before. When the window moves from left to right, the contents of the dictionary and the input characters that the pattern looks for will be changed. The search buffer will have thousands of bytes, and the look-ahead buffer has tens of bytes.

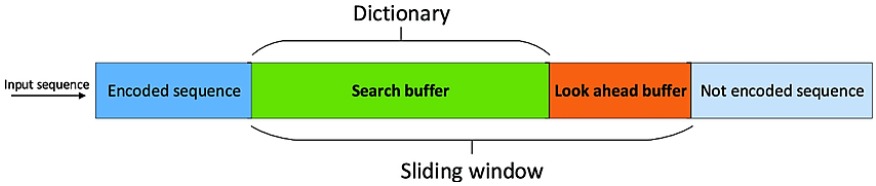

Figure 1. *LZ77 sliding window.*

b) LZ4 compression algorithm [7]

LZ4 is a dictionary-based algorithm proposed by Yann Collet in 2011. That was developed for high-speed compression and decompression and does not require the entire terrain data to be decompressed before use. LZ4 only uses a dictionary-matching stage and does not combine it with an entropy coding stage. The LZ4 algorithm represents the data as a series of sequences. Each sequence begins with a one-byte token that is broken into two 4-bit fields. The first field represents the number of literal bytes that are to be copied to the output. The second field represents the number of bytes to copy from the already decoded output buffer (with 0 representing the minimum match length of 4 bytes). A value of 15 in either of the bitfields indicates that the length is larger and there is an extra byte of data that is to be added to the length. A value of 255 in these extra bytes indicates that yet another byte is to be added. Hence, arbitrary lengths are represented by a series of extra bytes with a value of 255. The string of literals comes after the token, and any extra bytes are needed to indicate string length. This is followed by an offset that indicates how far back in the output buffer to begin copying. The extra bytes (if any) of the match-length come at the end of the sequence.

c) Brotli compression algorithm [8]

Brotli is a general-purpose data compressor introduced by Google in 2013 and is a lossless DC algorithm based on the Lempel-Ziv compression scheme, and its data format was submitted as RFC 7932 in 2016. The main goal of Brotli's design was to achieve maximal compression density, based on a pseudo-optimal entropy encoding of LZ77-phrases. The Brotli combines the LZ77 algorithm, the Huffman code [9], and the second modeling context. Compressed data consists of a header and a meta-blocks sequence. The header has the sliding window size parameter used during the compression process. Each meta-block is compressed using a combination of LZ77 and Huffman coding algorithms. Huffman coding is used as a prefix code. The prefix codes for each meta-block are independent of the meta-blocks before and after. In the Brotli, a reference can be used or fixed to a fixed dictionary (static dictionary entry). The compressed data consists of command lines and each command consists of two parts: a sequence of bit literals (a string that is not detected as a duplicate in the sliding window) and a pointer to a duplicate string.

2.3. Data Compression Evaluation

Some typical indicators used to evaluate the efficiency of DC are [10]:

a) Compression ratio (CR), also known as compression power, is a measurement of the relative reduction in size of data representation produced by a DC algorithm. It is typically expressed as the division of compressed size by uncompressed size. The lower the value of the compression ratio, the more satisfying the compression results are.

$$CR = \frac{Compressed\ Size}{Uncompressed\ Size} \quad (1)$$

b) Space saving (SS), also known as percentage of savings, is the ratio percentage between the difference in the size of the data from before it was compressed and after being compressed to the size of the data before being compressed. The greater the percentage of savings, the more satisfying the compression results.

$$SS = 1 - \frac{Compressed\ Size}{Uncompressed\ Size}\ x\ 100\% = (1 - CR)\ x\ 100\% \quad (2)$$

3. PROBLEM AND RELATED WORKS

3.1. Unity Terrain Data

a) Terrain data components

In Unity, the terrain is represented by the TerrainData class, which allows the user to build, edit, and display. All terrain components affect the amount of storage the terrain needs, however, the most affectionate factors in Unity are [11]:

- Heightmap: Data about the digital evaluation model (DEM), is a grayscale image and allows the creation of complex and varied terrains with ravines, hills, and valleys. The higher the resolution of DEM, the larger the terrain capacity;

- Texture: Represent different types of soil on the surface, such as grass, rock, mud, water, and snow. The texture is usually formatted as images. The larger the number and size of textures, the larger the terrain capacity;

- Objects: Including the main object (trees, constructions) and other detailed objects (grass clumps and rocks). The main objects are often 3D models. The larger the number of objects and the more detailed object quality, the larger the terrain capacity.

b) Unity Terrain Data Management

Two methods of managing terrain data in Unity are encapsulating data with the application and separating the data from the application. The latter method is widely used because terrain data can be updated independently of the application. To manage terrain data, Unity provides a technique that allows the packaging of extensive terrain data into AssetBundle (an archive container that holds additional files inside it that Unity can load at run time).

c) Loading Unity Terrain Data in Runtime

In the runtime process, Unity loads terrain data partially from AssetBundle to Random Access Memory (RAM) depending on the observation position. When the observation position changes, new data will be loaded, and old data will be deleted.

d) Factors affecting Unity terrain data capacity

3.2. Problem

Information integration system [12], minicomputers take care of personal-level data storage and processing. They not only have to store many terrain areas but also have to store other data, such as multimedia data. Therefore, there are often cases of overload in storage capacity. To solve this problem, we can reduce the terrain quality (level of detail and accuracy) to decrease storage capacity. However, to ensure the terrain is as realistic as required, it can only decrease the detail and accuracy to the allowable level, so the problem still is not solved. Based on the survey results during the editing of terrain data, we found that with Unity's terrain data types, there is certain repeatability in areas in the terrain that lead to redundancy. Therefore, it is possible to apply DC techniques to attack the existing problem. It should be noted that the Unity terrain data loading

mechanism is a file stream, so it cannot use DC algorithms specialized for backup/archive to compress terrain data.

Although lossy compression techniques have higher compression ratios than lossless compression methods. However, terrain data in the information integration system [12] is used for reconnaissance forces, so high accuracy is required. That is why we choose lossless compression algorithms such as the LZ4 and the Brotli.

3.3. Related works

According to the survey, relatively few publications exist on terrain DC techniques, especially for terrain in Unity. Some case studies include:

- Olanda Ricardo et al. [13] proposed a novel wavelet-tiled pyramid for compressing terrain data that replaces the traditional multiresolution pyramid usually used in wavelet compression schemes. The new wavelet-tiled pyramid modifies the wavelet analysis and synthesis processes, allowing an efficient transmission and reconstruction of terrain data in those applications based on multiresolution tiled pyramids;

- Guo Hao-Ran et al. [14] presented a high-performance terrain DC method based on lifting wavelet transform and parallel hybrid entropy codec, and combined with graphics process unit (GPU) Ray-casting to achieve large-scale 3D terrain visualization. First, the multi-resolution wavelet transform model of terrain tile is constructed to map the refinement and simplification operation. Then, the multi-resolution quadtree of DEM and terrain texture is built separately based on lifting wavelet transform, the sparse wavelet coefficient generated from quantization is compressed by a hybrid entropy codec, which is combined with parallel run-length coding and variable-length Huffman coding. The compressed data are organized into progressive streams to do real-time decoding and rendering;

- Ying Zhoua et al. [15] suggested a method to compress 3D models using the combination of the Quadric Error Metrics (QEM) algorithm and the enhanced edge collapse algorithm to solve the problem of undetected model structure and over-simplified model details. Their method can preserve the model's integrity while maintaining a high compression efficiency. This method works well for the compression of large-scale scene models, which includes the compression ratio and the structural preservation in the model.

- Maleika W. et al. [16] proposed a lossless compression method specifically for Digital Terrain Models (DTM) data. The method involves discarding redundant data, performing differential coding, variable length value coding, and finally, compression data using the LZ77 or the PPM algorithm;

- Maria Mrówczyńska et al. [17] combined the PCA transformation and neural network to compress data obtained from geodetic measurements. The results of vertical displacement measurements of a building have exemplified the applicability of these methods. The results of calculations carried out using artificial intelligence assisted and the PCA indicate that the approach can be effectively used to compress geodetic measurement results and reproduce them without losing the accuracy of displacement identification.

4. METHODOLOGY

4.1. Analysis

Unity offers two options to compress data in AssetBundle based on LZMA and LZ4 data compression algorithms. LZMA needs compressed data to be uncompressed entirely before use, so it cannot attack the problem of saving space capacity when terrain data is deployed on RLDs. LZ4 is a good algorithm for compressing terrain data, however, it aims to provide a good trade-off between speed and compression ratio. When conducting preliminary experiments, we found that LZ4 is effective at compressing 3D models such as trees and constructions in Unity 3D models but

is less efficient than other algorithms when compressing heightmap and texture data. From another perspective, Brotli is an effective compression algorithm for compressing text and images, so it is helpful in compressing heightmap and texture. However, it is still limited when compressing other file types like 3D model data. For information integration systems such as in [12], the balance between compression efficiency and decompression speed is also a concern because RLDs must simultaneously process other data types besides processing and displaying terrain data. To solve the above problems simultaneously, we suggest a suitable combination of LZ4 and Brotli to compress Unity terrain.

4.2. Hybrid data compression process

The hybrid process of compression and decompression is illustrated in figure 2.

a) Compression phase

Each part of the original terrain data (Unity AssetBundle) will be compacted by a suitable compressor, in which heightmap and texture will be condensed by the Brotli compressor first and then by the LZ4 compressor; objects will be compressed by the LZ4 compressor. After that, compressed terrain data is then deployed on RLDs.

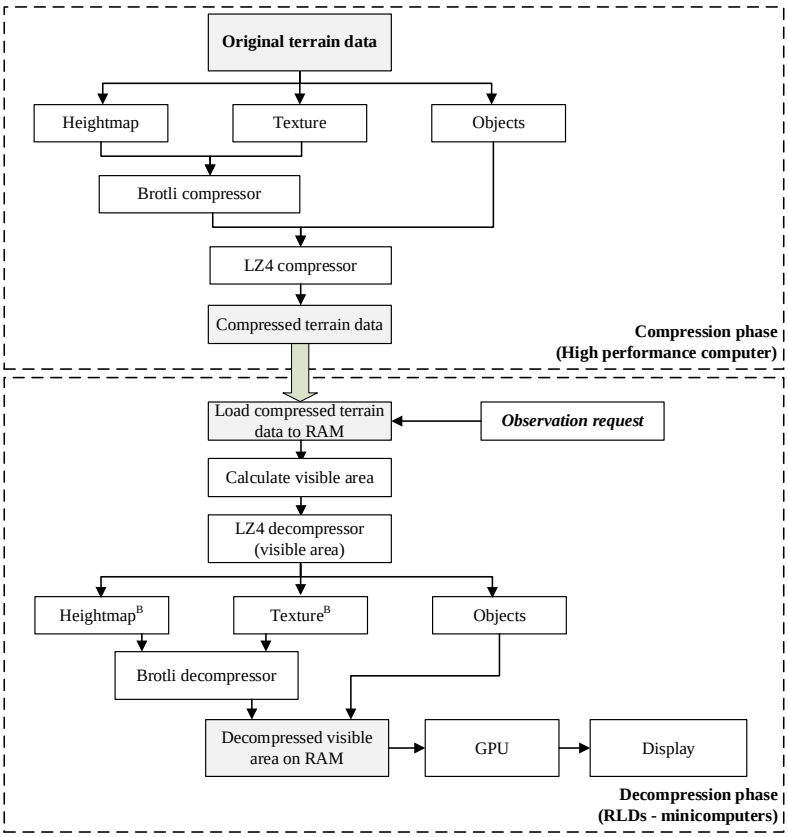

Figure 2. *Data compression and decompression process of Unity terrain data.*

b) Decompression phase

When having a user observation request:

- First, the compressed terrain data will be loaded into RAM;
- Secondly, a calculation will be implemented to determine the visible area corresponding to the observation request;
- Finally, the LZ4 and the Brotli decompressor will decompress the visible area data into RAM

for processing and transfer to the GPU for display. In this case, heightmap and texture are decompressed by the LZ4 first, and then the Brotli; objects are decompressed by the LZ4).

The visible area chosen to decompress includes only the area that can be observed at the camera location, so it is small. When changing the camera position, a new visible area will be decompressed, and the Unity cache management mechanisms will delete the old data. The decompressed data is only stored in RAM and does not increase the data stored in the hard drive during the decompression process. Combining the LZ4 and the Brotli compression method takes advantage of both methods. In this case, the LZ4 is effective when compressing object data and Brotli is effective when compressing heightmap and texture data.

5. EXPERIMENTS, RESULTS AND DISCUSSION

5.1. Experiments

To evaluate the performance of the proposed method, experiments were conducted with the following conditions:

- Hardware: Compression computer (Intel Core i7 7800X CPU processor, 16 GB of RAM); Minicomputer (Ryzen 5 5600G 3.9 GHz processor, 16 GB of RAM, 256 GB HDD).
- Unity framework: Version 2022.2.1;
- Terrain data: This includes three types, as shown in T. For each terrain type, the experiment was executed with each algorithm separately and the hybrid method.
- Accuracy: Maximize by default of Unity (Heightmap was 16 bits, Alphamap was 8 bits).

Table 1. *Experimental terrain data.*

Index	Name	Square	Components of the terrain
1	TD 1	20.25 km^2 (4.5 x 4.5 km)	- Heightmap (DEM): 30 m resolution - Texture: Satellite image, level 19 of Google
2	TD 2	20.25 km^2 (4.5 x 4.5 km)	- Heightmap (DEM): 30 m resolution - Texture: Satellite image, level 19 of Google - Objects: Trees (30,000 trees from 10 different catergories, random planted)
3	TD 3	20.25 km^2 (4.5 x 4.5 km)	- Heightmap (DEM): 30 m resolution - Texture: Satellite image, level 19 of Google - Objects: Trees (30,000 trees from 10 different catergories, random planted) and constructions (100 constructions from 20 different building catergories)

5.2. Results and Discussion

In order to illustrate the performance of our proposal, we compressed three types of terrain data from low to high storage capacity corresponding to the level of detail from low to high. The experimental results are shown in table 2 - table 4, in which: Terrain data storage is rounded to MB; Compression ratio is calculated according to the formula (1); Space saving is calculated according to the formula (2). With terrain data storage and compression ratio, the smaller the value, the better. On the contrary, with space-saving, the larger the value, the better. Values shown in bold font are best, and those shown in bold italic font are better than those shown in standard font.

Table 2. *Storage (less is better, bold is the best).*

Index	Name	Storage (rounded to MB)			
		Uncompression	***LZ4***	***Brotli***	***Hybrid***
1	TD1	1517	155	113	**103**
2	TD2	1792	362	382	**308**
3	TD3	1878	397	407	**334**

Table 3. *Compression ratio (less is better).*

Index	Name	Compression ratio		
		LZ4	***Brotli***	***Hybrid***
1	TD1	0.102175	0.074489	**0.067897**
2	TD2	0.202009	0.213170	**0.171875**
3	TD3	0.211395	0.216720	**0.177849**

Table 4. *Space saving (more is better).*

Index	Name	Saving space (%)		
		LZ4	***Brotli***	***Hybrid***
1	TD1	89.782%	92.551%	**93.210%**
2	TD2	79.799%	78.683%	**82.813%**
3	TD3	78.860%	78.328%	**82.215%**

Based on the experimental results, the following conclusions can be made:

- The Brotli method is better than the LZ4 when compressing terrain data that does not contain objects: Experimental results on TD1 terrain data (only includes heightmap and texture data) show that CR using Brotli is significantly higher than LZ4.

- The LZ4 method is better than the Brotli when compressing terrain data containing objects: Experimental results on TD2 and TD3 terrain data (including heightmap, texture, and object data) show that CR using the LZ4 is slightly better than the Brotli.

- The proposed method is always better than using each method individually. Although the LZ4 is more effective in compressing object data, it still has a particular effectiveness in compressing heightmap and texture data. Therefore, combining the LZ4 with the Brotli still results in higher compression of heightmap and texture data than using the Brotli alone.

6. CONCLUSIONS

In this paper, we combined LZ4 and Brotli DC methods to compress terrain data in the Unity framework's application deployed on RLDs. The experiment results have shown that the proposed method has good performance and efficiency on DC compared to applying each algorithm independently, and the proposed research hypothesis was proved. We have applied the research results in compressing terrain data to deploy on minicomputers of the information integration system [12]. In the future, we will survey the combination of other compression algorithms to improve the performance of terrain data compression.

REFERENCES

[1]. J. Uthayakumar, T. Vengattaraman, P. Dhavachelvan. *"A survey on data compression techniques: From the perspective of data quality, coding schemes, data type and applications",* Journal of King Saud University - Computer and Information Sciences, (2018) https://doi.org/10.1016/j.jksuci.2018.05.006

[2]. Parkinson, C.N. *''Work expands so as to fill the time available."* In: Parkinson's Law and Other Studies in Administration. Ballantine Books, New York, (1957).

[3]. P. Lancett, *"The advantages of file compression"*, Techwalla, (2018), https://www.techwalla.com/articles/the-advantages-of-file-compression.

[4]. Er. Mangi Lal , Er. Sammah Rasheed, *"A Review on data compression techniques",* IJARIIE-ISSN(O)-2395-4396, Vol-6 Issue-1, pp.590-597, (2020).

[5]. https://ethw.org/History_of_Lossless_Data_Compression_Algorithms.

[6]. H. Dheemanth, *"LZW data compression"*, American Journal of Engineering Research (AJER), vol.3, no.2, pp.22-26, (2018).

[7]. Euronext, *"Introduction to LZ4 compression"*, Ruronext optiq technical notes, (2016).

[8]. Alakuijala, Jyrki & Farruggia, Andrea & Ferragina, Paolo & Kliuchnikov, Eugene & Obryk, Robert & Szabadka, Zoltan & Vandevenne, Lode. *"Brotli: A General-Purpose Data Compressor",* ACM

Transactions on Information Systems. 37. 1-30. 10.1145/3231935, (2018).
[9]. Techie Delight, *"Huffman Coding Compression Algorithm - Techie Delight",* (2018), http://www.techiedelight.com/huffman-coding.
[10]. Vicky Reynaldo, Arya Wicaksana and Seng Hansun. *"Brotli data compression on moodle-based e-learning server"*, ICIC International 2019 ISSN 2185-2766, (2019).
[11]. https://docs.unity3d.com/ScriptReference/Terrain.html.
[12]. Minh Tran Binh, The research topic "ĐTVCN.01.22/CNTT", Military Information Technology Institute, (2021).
[13]. Olanda Ricardo, Pérez Mariano, Orduña Juan, Rueda Silvia, *"Terrain data compression using wavelet-tiled pyramids for online 3D terrain visualization",* International Journal of Geographical Information Science, 28, pp.407-425, (2014).
[14]. Guo Hao-Ran, Pang Jian-Min, *"Terrain Data Hybrid Entropy Coding Compression Based on Lifting Wavelet and Real-time Rendering",* Journal of Electronics & Information Technology, 34(12), pp.3013-3020, (2012).
[15]. Ying Zhoua, Lingling Wanga, Lieyun Dinga, Cheng Zhoua, *"A 3D model Compression Method for Large Scenes",* 35th International Symposium on Automation and Robotics in Construction (2018).
[16]. Maleika W., Forczmański P. *"Lossless Compression Method for Digital Terrain Model of Seabed Shape".* In: Choraś, R. (eds) Image Processing and Communications Challenges 8. Advances in Intelligent Systems and Computing, vol 525. Springer, Cham, (2017). https://doi.org/10.1007/978-3-319-47274-4_18
[17]. Maria Mrówczyńska, Jacek Sztubecki, Andrzej Greinert. *"Compression of results of geodetic displacement measurements using the PCA method and neural networks",* (2020). https://doi.org/10.1016/j.measurement.2020.107693.

TÓM TẮT

Phương pháp nén dữ liệu địa hình trên Unity phục vụ triển khai trên các thiết bị có tài nguyên hạn chế

Trong các hệ thông tin địa lý nói chung và hệ thống mô phỏng nói riêng, dữ liệu địa hình chiếm phần lớn dung lượng ổ cứng máy tính và thường được triển khai trên máy chủ dữ liệu. Dữ liệu địa hình có độ phân giải càng cao, càng chi tiết thì dung lượng chiếm dụng trên ổ cứng càng lớn. Trường hợp dữ liệu cần triển khai trực tiếp trên các thiết bị có tài nguyên hạn chế như máy tính mini thì sẽ gặp nhiều khó khăn do dung lượng ổ cứng có hạn. Nén dữ liệu địa hình là một giải pháp làm giảm dung lượng địa hình để khắc phục tồn tại đó. Nghiên cứu này đề xuất một phương pháp nén dữ liệu địa hình của Unity sử dụng dựa trên việc kết hợp các thuật toán Brotli và LZ4 để triển khai trên các thiết bị có tài nguyên hạn chế. Kết quả nghiên cứu cho thấy, dung lượng dữ liệu địa hình sau khi nén giảm đáng kể so với khi sử dụng độc lập từng thuật toán thành phần trong khi vẫn đảm bảo chất lượng.

Từ khóa: Dữ liệu địa hình; Nén dữ liệu; Unity; Thiết bị có tài nguyên hạn chế.