TTG: Text-to-Terrain Generation

Final Research Project Report of Computer Graphics, COMP4610-2024

Oucheng Liu	**Tianchen Guo**	**Xinni Song**
U7439247	U7439173	U7439250

The Australian National University

Abstract

Terrain modeling represents a critical area of research in computer graphics, with applications spanning from gaming to simulation. Traditional methods, such as Perlin noise and value noise, have long been utilized to generate terrains. While effective, these techniques inherently produce random terrains that do not take into account user-specific requirements or detailed specifications. To address this limitation, we introduce a novel approach known as **T**ext-to-**T**errain **G**eneration (**TTG**) which leverages textual descriptions to create corresponding terrains. Due to the absence of pre-existing terrain-descriptive datasets, two bespoke datasets are developed: Distribution-to-Terrain and Description-to-Terrain. Initially, we leverage the Distribution-to-Terrain dataset to train a classifier that can categorize elevation distributions into five basic terrains. Next, the generator, trained with the Description-to-Terrain dataset, translates textual descriptions into specific elevation distributions. Finally, these distributions are converted into comprehensive regional elevation data using the value noise method, and visualized using Terragen 4. Our exploration led us to adopt the Word2Vec model for text vectorization coupled with a Deep Neural Network (DNN) for the generator. The effectiveness of our method was rigorously evaluated through trained classifier, manual assessments, and visualizations, demonstrating that our approach offers an efficient and user-friendly solution for terrain generation.

1 Introduction

For over three decades, the modeling and synthesis of terrain have been pivotal areas of investigation within computer graphics research. Due to the complexity of geographic variations, influenced by human activity, rain erosion, and tectonic movements, the precise synthesis of realistic terrains presents significant challenges[1]. Additionally, in the context of gaming, accurately generating terrain to meet players' needs holds substantial practical value.

This work explores the application of value noise, a variant of Perlin noise, in generating terrains for computer games and terrain simulators. Our goal is to create terrains that are more user-friendly and tailored to players' needs, that is, to generate terrains corresponding to specific geographic features described in textual inputs. A major challenge of this is the lack of datasets for training text to generate distributions. Another significant challenge is the current absence of existing models for generating desired terrains from textual descriptions, requiring us to build the architecture from scratch. To achieve this, we propose a framework named **T**ext-to-**T**errain **G**eneration (**TTG**).

Specifically, our method is as follows: First, we established a Distribution-to-Terrain dataset to train a distribution classifier that discerns which of the five fundamental terrains (plains, plateaus, basins, hills, mountains) [2] a given distribution belongs to. Next, we set up a Description-to-Terrain

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

dataset to train a distribution generator. We combined the trained distribution classifier with the distribution generator to classify the terrain distribution generated from a terrain description prompt into corresponding labels as a classification task. For example, if a user inputs "generate a flat terrain," we generate a elevation distribution that matches a plain and classify it under the plains label. Building on the work of [3], we utilize the elevation distribution derived from text and apply value noise to produce an 8000×8000 elevation distribution data. Finally, after beautification and rendering in Terragen 4, we visualize the terrain that matches the user's textual description.

We conducted extensive experiments on our constructed datasets, exploring the performance of different text representation/text vectorization methods (TF-IDF, Word2Vec, RoBERTa tokenization) and different networks as terrain generators (DNN, SVM, textCNN, RoBERTa). We found that the combination of Word2Vec and DNN significantly outperformed other methods. Additionally, we analyzed the discrepancies between the quality of the generated distributions as judged manually and the results determined by the trained classifiers, demonstrating the accurate performance of our method, except in some special categories. We visualized the generated terrains to further showcase the effective generation capabilities of our method. Based on the experimental results and findings, we summarize the main contributions of our work as follows:

- Developed a Text-to-Terrain Generation framework. To our knowledge, this is the first model that generates desired terrains based on textual descriptions.
- Created two new datasets that can be used for classifying terrains from elevation distributions and generating distributions from textual descriptions, filling a gap in training data for this domain.
- We built a framework based on DNN and Word2Vec and conducted extensive experiments to validate it performance, providing a benchmark for future work in text-driven terrain generation.

2 Related Work

Given the infinite nature of the explorable virtual world, imposing resilient boundaries such as a valley surrounded by mountains or an island enclosed by the ocean is undesirable. Instead, it is preferable to allow users to explore freely without encountering obstacles[4]. Terrain generation can be classified into three primary approaches: procedural, simulation-based, and example or learning-based methods. In general, terrain generation involves the creation of surface meshes that accurately describe terrain shape and elevation[5].

The development of procedural algorithms, such as noise generation, has enabled the creation of infinite terrains. Procedural generation utilizes algorithmic methods to create landscapes by exploiting the self-similarity or fractal structure inherent in terrain. This includes the use of noise functions like Perlin Noise. For instance, Li et al. proposed a parallel algorithm for terrain generation based on CUDA architecture to address the high computational load and low efficiency associated with generating large-scale terrains using the Perlin noise superposition method[6]. Similarly, Parberry et al. used value noise, a variant of Perlin noise, to generate terrain based on real-world USGS elevation data[3]. Value noise and deep learning are used in our paper's approach, which is comparable to Parberry's approach, to build terrain.

Simulation-based methods employ tools to replicate the natural processes of erosion over a given terrain, often used to simulate specific landforms[7]. Cordonnier et al. introduced a novel framework for interactive landscape authoring that supports bi-directional feedback between erosion and vegetation simulation. Schott et al[8].[9] shifted from modeling in the elevation domain to the uplift domain, simulating stream power erosion to create hydrologically consistent reliefs without sacrificing user control.

Example or learning-based methods generate new terrains from examples or sketches of terrains. Hnaidi et al. presented a diffusion method for generating terrains from parameterized curves that characterize landform features such as ridge lines, riverbeds, or cliffs. Additionally, deep-learning-based solutions, such as Generative Adversarial Networks (GANs), have been utilized for realistic and rapid terrain generation[10]. Zhang et al. developed a conditional GAN that encourages maximum-distance embedding of acquired styles in the latent space, enabling the rapid synthesis of multi-style terrains directly learned from real terrain data[11].

3 Methodology

In this section, we introduce TTG, a novel method for generating desired terrains through textual descriptions. We begin by explaining how we utilize the Distribution-to-Terrain dataset to train the distribution classifier. Next, we describe the process of training the distribution generator using the Description-to-Terrain dataset. Finally, we demonstrate how to generate and visualize the desired terrain based on the provided descriptions. We give an overview of the TTG in Figure 1.

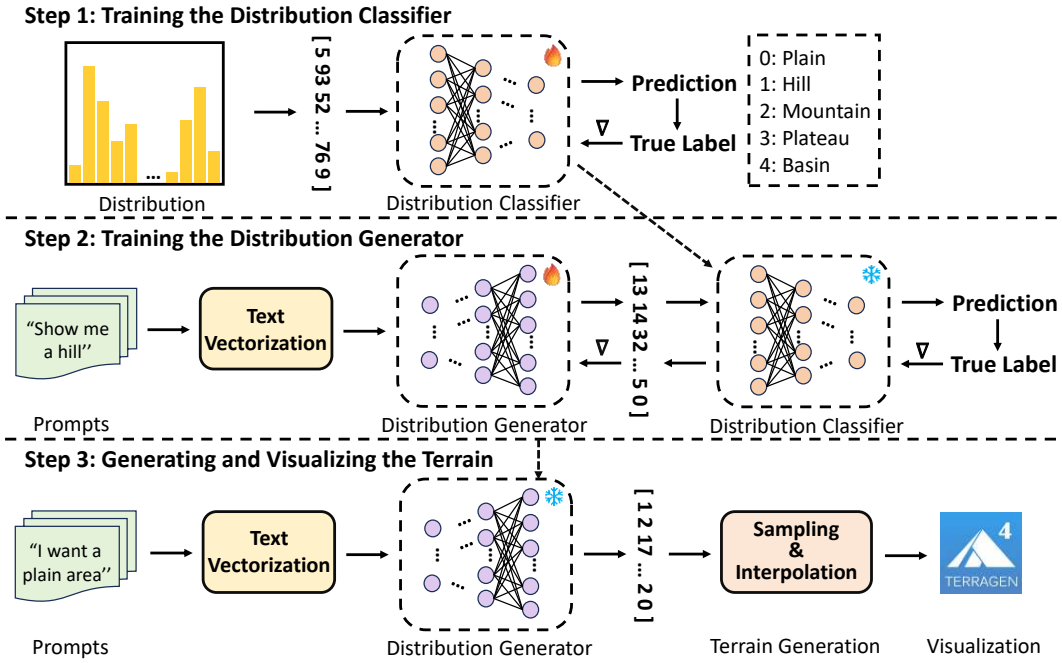


Figure 1: Overview of Text-to-Terrain Generation's pipeline.

3.1 Terrain Distribution Classifier

3.1.1 Distribution-to-Terrain Dataset

Based on the differences in appearance and morphology, terrain is primarily categorized into five types: plains, hills, mountains, plateaus, and basins[2]. From observations of past models[3] of random terrain generation, we found that the types of terrain generated are closely related to the distribution of randomly sampled altitudes provided, which aligns with the intuitive understanding of these five terrain types:

- **Plain:** Flat with low to medium elevation and minor fluctuations, hence the elevation distribution is concentrated within a narrow range of lower elevations.
- **Hill:** Situated between plain and mountain, with slight elevation variations, thus the elevation distribution is primarily focused within a moderate range of mid-elevations.
- **Mountain:** Characterized by large elevation changes and steep terrains, thus the elevation distribution is spread across a broader range.
- **Plateau:** Have the highest elevations but relatively flat surfaces, therefore, the elevation distribution is concentrated within a narrow range of higher elevations.
- **Basin:** Relatively the lowest elevations with flat interiors, hence the elevation distribution is concentrated within a narrow range of lower elevations.

Based on these characteristics, we constructed a dataset called the Distribution-to-Terrain Dataset. In line with previous work[3], we uniformly divided the elevation range of [-4000, 4000] into 31 intervals, assigning a weight to each interval to represent the proportion of elevation sampling, with

the total sum of weights being 256. We consider these 31 weights as the distribution of terrain elevations, which serve as features in our dataset. The terrain type generated corresponding to this distribution is used as the label. We set the labels for the five types of terrain as follows:

$$\{\text{Plain: } 0;\ \text{Hill: } 1;\ \text{Mountain: } 2;\ \text{Plateau: } 3;\ \text{Basin: } 4\}$$

We ensure that it is balanced with each terrain type represented by 100 different distributions. Considering that the primary use of this dataset is to train a classifier that can categorize elevation distributions into terrain types, an intermediate step rather than for final terrain generation, we just divide it into train and test sets in a 7:3 ratio.

3.1.2 Distribution Classifier

For the Distribution-to-Terrain dataset, we utilize a Deep Neural Network (DNN) with three hidden layers as the classifier. The network accepts a 31-dimensional vector $\mathbf{x}$ as input, which representing the elevation distribution. The DNN can be formulated as follows:

$$\mathbf{h}_i = \text{ReLU}(\mathbf{W}_i\mathbf{h}_{i-1} + \mathbf{b}_i) \quad i = 1, 2, 3 \tag{1}$$

where $\mathbf{h}_0 = \mathbf{x}$. The final output layer produces a 5-dimensional vector $\mathbf{y} = \text{softmax}(\mathbf{W}_4\mathbf{h}_3 + \mathbf{b}_4)$, using the softmax function to classify into one of five terrain categories. The dimensions of the hidden layers are empirically set to [64, 32, 16], this configuration achieved a 100% accuracy rate, which meets the demands of subsequent tasks where the classifier's reliability is crucial for generating realistic terrain distributions.

3.2 Terrain Distribution Generator

3.2.1 Description-to-Terrain Dataset

To construct a more user-friendly model, we aimed to enable terrain generation through natural language inputs. Thus, we created the Description-to-Terrain dataset, with features as textual prompts and labels corresponding to specific terrain types. Each terrain category has 50 unique prompts, and the dataset is divided into training, validation, and test sets in a 7:1:2 ratio. We employed GPT-4o to generate prompts, ensuring a diversity in tone, complexity, and vocabulary (using synonyms) for describing terrains. We give an example below:

$$\text{Variations in } \begin{cases} \text{tones} & : \text{“\{Give me a basin landscape\}” and “\{Generate a flat highland\}”} \\ \text{complexity:} & \text{“\{Small hill\}” and “\{Give me a mountainous landscape\}”} \\ \text{vocabulary:} & \text{“\{I need a flat area\}” and “\{Show me a plain area\}”} \end{cases}$$

3.2.2 Distribution Generator

Training a elevation distribution generator is the core of our task. We combine the distribution generator with the distribution classifier to train on the task of classifying generated terrain distributions. The process begins by vectorizing the textual prompts from the Description-to-Terrain Dataset using Word2Vec technology. The text vectors are then input into the distribution generator, which outputs a 31-dimensional elevation distribution. The previously trained distribution classifier is responsible for predicting the terrain type from these distributions. We compute the loss by comparing these predictions with the true labels, employing a cross-entropy loss function:

$$\mathcal{L} = -\sum_{i=1}^{C} y_i \log(\hat{y}_i) \tag{2}$$

where C is the number of categories, y_i is the integer label of the true category, and $\hat{y}_i$ is the predicted probability of the model for the i-th category. During backpropagation, we consider the classifier as a form of prior knowledge with fixed parameters and update only the parameters of the distribution generator.

Specifically, we utilize the `word2vec-google-news-300` model for text vectorization, which has been pretrained on the Google News dataset using the skip-gram algorithm[12]. Its proficiency in handling synonyms aids in addressing terrain expressions that were not encountered during training[12]. Additionally, our distribution generator is a simple Deep Neural Network (DNN).

Empirical evidence suggests that DNNs with 2-3 hidden layers achieve optimal performance, as having too few or too many layers can lead to underfitting or overfitting respectively. In Section 4, we further compare this setup with alternatives where text vectorization and distribution generation are done using RoBERTa[13], or the text vectorization is carried out using TF-IDF[14], or the distribution generator is SVM[15] or textCNN[16].

3.3 Generation and Visualization

In this step, we fix the parameters of the distribution generator, allowing the vectorized text of a given input prompt to be processed by the generator to yield a distribution consistent with the expected terrain described by the prompt. Following previous work[3], we achieve terrain elevation data for an area of 8000×8000 by appling value noise method: create a lattice of dots with random values sampled from the distribution, then returns linearly interpolated numbers based on the values of the surrounding lattice points. Finally, the terrain is visualized using the Terragen 4 software.

4 Experiments and Discussion

4.1 Settings

We configure the hidden layers of the DNN distribution generator to [128, 64, 32]. Beside, we set kernel sizes to [3, 4, 5] and filters' number to 10 for textCNN. The SVM is employed with an RBF kernel. The distribution generator is trained with a learning rate of 0.001 using the Adam optimizer. Depending on the actual situations, we train for 20 to 50 epochs; for instance, TF-IDF typically requires more than 40 epochs, whereas 20 epochs are usually sufficient for Word2Vec to avoid overfitting. We conduct 3 random trials for each model to ensure reliability of the results.

4.2 Performance

4.2.1 Strategy Comparison

In Table 1, we present a comparison of Accuracy and Precision for distributions generated using different models as generators and various text vectorization methods. It is evident that using a DNN as the classifier significantly outperforms other models. Additionally, employing Word2Vec[12] for text vectorization further enhances model performance, an improvement we attribute to Word2Vec's ability to handle synonyms effectively, allowing it to manage terrain expressions not encountered during the training phase. We also observed that the performance of more complex models was only slightly better than random guessing, particularly with RoBERTa[13], one of the mainstream language models, which achieved only a 0.20 accuracy rate, categorizing all samples into one class. We speculate this may be due to the lack of pre-training on our specific task and an unsuitable sample size for fine-tuning.

Table 1: Experiment Results for Different Text Features and Classifiers.

Classifier	Accuracy	Precision
TF-IDF		
textCNN	0.20 ± 0.02	0.10 ± 0.01
SVM	0.28 ± 0.06	0.12 ± 0.02
DNN	0.78 ± 0.02	0.81 ± 0.00
Word2Vec		
textCNN	0.20 ± 0.00	0.04 ± 0.00
SVM	0.25 ± 0.05	0.14 ± 0.10
DNN	$\mathbf{0.84 \pm 0.04}$	$\mathbf{0.85 \pm 0.05}$
RoBERTa		
RoBERTa	0.20 ± 0.00	0.04 ± 0.00

4.2.2 Generated Distribution Manually Examination

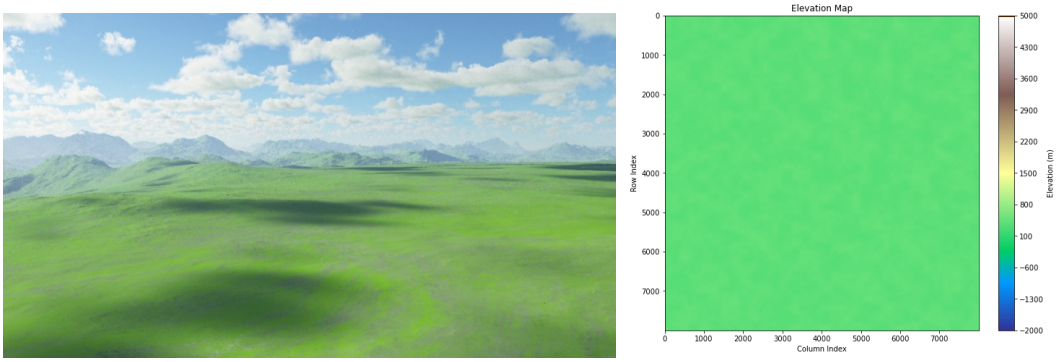
Our distribution classifier is trained based on existing data and may not have encountered some possible distributions. Therefore, when the generator produces some new distributions, the classifier might erroneously assess whether these distributions are correct. To more accurately evaluate the effectiveness of the distribution generator, we select the results that achieved the highest accuracy on the test set and manually checked whether these distributions are correctly generated. The precision of classifier and manual judgments for five terrains is presented in Table 2. The results indicate that the classifier and manual judgments are generally consistent for Plain, Hill, Mountain, and Plateau. However, for the Basin category, the generated distribution performed poorly, with a precision of only 0.20, and most cases were misclassified as Plains or Plateaus. This counterintuitive result might be due to the fact that basins also tend to have low relief variations and are sometimes described with terms like "flat" in text prompts, similar to plateaus and plains. This discovery suggests the need for more robust modeling methods, which represents a potential direction for future improvements.

Table 2: Comparison of Classifier and Manual Precision Across Different Terrain Types

Terrain Type	Classifier	Manual
Plain	0.80	0.70
Hill	1.00	1.00
Mountain	0.80	1.00
Plateau	1.00	1.00
Basin	0.70	0.20

4.3 Terrain Visualization

To visualize the terrain, since Terragen 4 does not accept `.asc` file types, we initially generate the elevation data in the `.asc` format. We then use QGIS, a free and open-source geographic information system, to convert this data from the `.asc` format to a `.tif` file. Finally, the `.tif` file is imported into Terragen 4 for rendering the terrain visually.



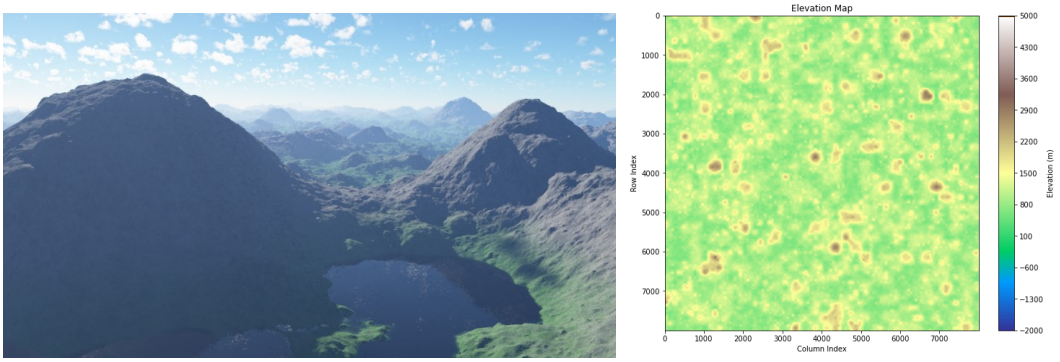
(a) **Plain** (b) Elevation Map

Figure 2: The rendered visualization results and elevation map of **Plain** landform.

Plain: To generate a Plain, we gave our network a prompt **"Generate a steppe"**, and then got a result shown in Figure 2. It can be seen that the generated terrain conforms to the terrain characteristics of the plain: very flat with little elevation variation. It should be noted that, to enhance the visual effect, we added green to represent vegetation in the rendered model. The elevation map shows that, despite the color span being set from -2000 to 5000, the image primarily displays green, indicating relatively low or medium elevation. The entire region exhibits minimal elevation variation, suggesting that the elevation range is very small and concentrated within a low elevation range.

Hill: To generate a hill, we gave our network the prompt **"Generate a mound"**. Based on the provided hill image and its corresponding elevation map (Figure 3), we can conclude the following: the hill's terrain is characterized by gentle undulations, with smaller elevation variations compared to mountains, and relatively smooth terrain. The image shows some scattered highlands, indicating

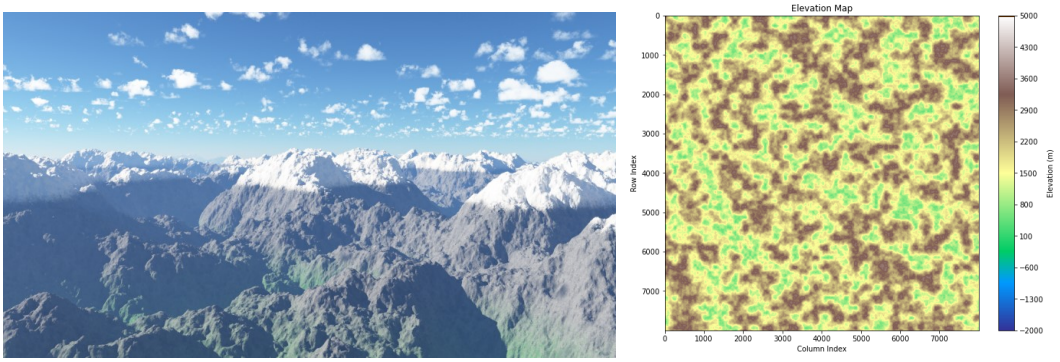
that the terrain is not completely flat. To enhance the visual effect, we added vegetation and water. The elevation map shows moderate elevation changes, primarily displaying green and yellow areas, indicating small elevation variations. Most of the region's elevation ranges from 0 to 1500 meters, highlighting the low elevation characteristics.



(a) **Hill** (b) Elevation Map

Figure 3: The rendered visualization results and elevation map of **Hill** landform.

Mountain: To generate a mountain, we gave our network the prompt **"Generate a towering terrain"**, and then obtained the result in Figure 4. The generated mountain terrain is complex and has many mountains, the significant change in elevation makes the terrain appear very rugged. To enhance the visualization, we incorporated elements of snow to depict the low temperatures typical of high altitudes. In elevation map, yellow and brown areas represent higher elevations, while green and blue areas indicate lower elevations. Despite the color span being set from -2000 to 5000, the elevation map reveals a mix of various colors, indicating significant terrain variation.

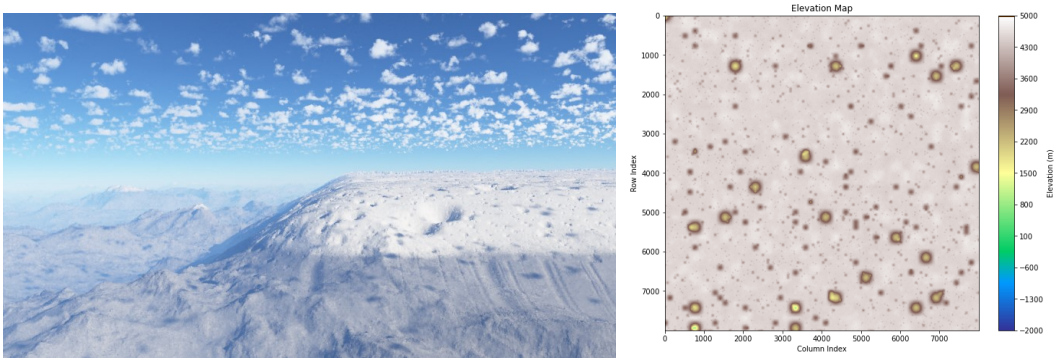


(a) **Mountain** (b) Elevation Map

Figure 4: The rendered visualization results and elevation map of **Mountain** landscape.

Plateau: For the creation of a plateau, we gave our network a prompt **"Plateau landscape"**, and then got a result as follow. Based on the provided plateau image and its corresponding elevation map (see Figure 5), we can conclude the following: the plateau's terrain is characterized by its vast expanse and high elevation, with some localized variations in elevation. The elevation map shows that the generated high-altitude areas are almost all above 3000 meters. Despite the color span being set from -2000 to 5000, the different colored spots in the map reflect local variations in elevation. Yellow and brown spots represent high-altitude areas, while green and blue indicate relatively lower elevations.

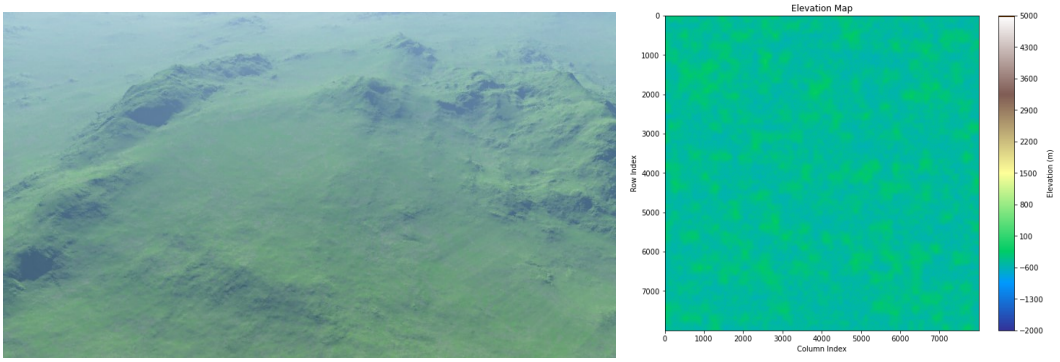
Basin: To generate a basin, we gave our network the prompt **"Show me a basin area"**, and then obtained the following result. Based on the provided basin image and its corresponding elevation map (Figure 6), we can conclude the following: the basin's terrain is characterized by low-lying areas, possibly surrounded by higher terrain, with lower elevations in the center. Although the basin is generally flat, there are still some subtle undulations and variations. To make the visualization more clear, we chose to render the image from an overhead perspective, highlighting the low-lying nature



(a) **Plateau** (b) Elevation Map

Figure 5: The rendered visualization results and elevation map of **Plateau** landform.

of the area. Additionally, the elevation map shows that the basin area has relatively small elevation changes, primarily concentrated in a low elevation range, with blue and green areas indicating low elevation characteristics.



(a) **Basin** (b) Elevation Map

Figure 6: The rendered visualization results and elevation map of **Basin** landform.

5 Conclusion and Future Work

5.1 Conclusion

In summary, our project has successfully developed the TTG framework, which transforms textual descriptions into realistic terrain visualizations. Our methodology, supported by specifically built datasets, has proven effective in generating accurate terrains. The performance of our models, particularly the combination of DNNs and Word2Vec, has been promising, suggesting potential avenues for further enhancements in procedural terrain generation.

5.2 Future Work

In further research, we can first build a larger Distribution-to-Terrain and Description-to-Terrain datasets, allowing the model to more generically capture terrain features and improve terrain generation robustness. Additionally, we can explore other model architectures, including RNN and LSTM. For RoBERTa and other BERT-based models [13], we can further explore fine-tuning strategy to adapt them to this task. We can also investigate how to represent distributions more effectively, thus the model generates terrain more accurately. Moreover, we use shorter prompts as input in this work, but we can explore using more detailed instructions to guide the model to generate terrain more precisely as required, such as creating canyons on the sides of the region.

References

[1] Shunji Ouchi. Response of alluvial rivers to slow active tectonic movement. *Geological Society of America Bulletin*, 96(4):504–515, 1985.

[2] Junko Iwahashi, Izumi Kamiya, Masashi Matsuoka, and Dai Yamazaki. Global terrain classification using 280 m dems: segmentation, clustering, and reclassification. *Progress in Earth and Planetary Science*, 5:1–31, 2018.

[3] Ian Parberry. Designer worlds: Procedural generation of infinite terrain from real-world elevation data. *Journal of Computer Graphics Techniques*, 3(1), 2014.

[4] Kazimierz Choroś and Jacek Topolski. A method of the dynamic generation of an infinite terrain in a virtual 3d space. In *Intelligent Information and Database Systems: 7th Asian Conference, ACIIDS 2015, Bali, Indonesia, March 23-25, 2015, Proceedings, Part II 7*, pages 377–387. Springer, 2015.

[5] Aryamaan Jain, Avinash Sharma, and KS Rajan. Learning based infinite terrain generation with level of detailing. In *International Conference on 3D Vision (3DV) 2024*, 2024.

[6] Huailiang Li, Xianguo Tuo, Yao Liu, and Xin Jiang. A parallel algorithm using perlin noise superposition method for terrain generation based on cuda architecture. In *International Conference on Materials Engineering and Information Technology Applications (MEITA 2015)*, pages 967–974. Atlantis Press, 2015.

[7] Norishige Chiba, Kazunobu Muraoka, and Kunihiko Fujita. An erosion model based on velocity fields for the visual simulation of mountain scenery. *The Journal of Visualization and Computer Animation*, 9(4):185–194, 1998.

[8] Guillaume Cordonnier, Eric Galin, James Gain, Bedrich Benes, Eric Guérin, Adrien Peytavie, and Marie-Paule Cani. Authoring landscapes by combining ecosystem and terrain erosion simulation. *ACM Transactions on Graphics (TOG)*, 36(4):1–12, 2017.

[9] Hugo Schott, Axel Paris, Lucie Fournier, Eric Guérin, and Eric Galin. Large-scale terrain authoring through interactive erosion simulation. *ACM Transactions on Graphics*, 42(5):1–15, 2023.

[10] Houssam Hnaidi, Eric Guérin, Samir Akkouche, Adrien Peytavie, and Eric Galin. Feature based terrain generation using diffusion equation. In *Computer Graphics Forum*, volume 29, pages 2179–2186. Wiley Online Library, 2010.

[11] Jian Zhang, Chen Li, Peichi Zhou, Changbo Wang, Gaoqi He, and Hong Qin. Authoring multi-style terrain with global-to-local control. *Graphical Models*, 119:101122, 2022.

[12] Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In Yoshua Bengio and Yann LeCun, editors, *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013.

[13] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019.

[14] Claude Sammut and Geoffrey I. Webb, editors. *TF–IDF*, pages 986–987. Springer US, Boston, MA, 2010.

[15] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[16] Yuebing Zhang, Zhifei Zhang, Duoqian Miao, and Jiaqi Wang. Three-way enhanced convolutional neural networks for sentence-level sentiment classification. *Inf. Sci.*, 477:55–64, 2019.