

Supplementary Materials: Controllable Procedural Generation of Landscapes

Anonymous Authors

A SUPPLEMENTS FOR THE METHOD

A.1 Parameterization from the Input Text

In the parameterization step, the system prompt m_s is divided into four parts: The first part establishes the assistant's identity and offers a brief background introduction, and the second part provides a detailed problem explanation and clarifies vital concepts. The third one outlines the task, instructing the assistant to generate specific parameters based on user input. The last part precisely specifies the output text format for parsing. Given that user input might not cover all relevant factors considered in the prompt, we allow the assistant to provide "default" results when specific parameters are not explicitly mentioned. A simplified version of the system prompt is illustrated in Figure 1. Note that defining the context is a once-and-for-all process. The same context can be combined with various inputs and used for numerous queries.

A.2 Specifications

A grid with $w \times h$ square cells (with w columns and h rows) is used, with a total of $(w+1) \times (h+1)$ corners and $h \times (w+1) + (h+1) \times w$ edges in the grid. In the subsequent descriptions, the length of an edge is considered as 1, which serves as the unit distance for the grid. Similarly, two adjacent cells (left and right or front and back) are considered to have a unit distance. In most calculations, we compute the distance between cells, edges, or corners using the Euclidean metric. It is also important to recognize that a unit distance corresponds to D meters in the real-world context.

In this paper, an edge e is defined as the border between two cells or the connected part between two corners v_1 and v_2 , denoted as $e = (v_1, v_2)$. A road R comprises $k \geq 1$ connected edges linking two different corners v_A and v_B , i.e., $R = \{(v_A, v_1), (v_1, v_2), \dots, (v_{k-2}, v_{k-1}), (v_{k-1}, v_B)\}$.

A.3 Genetic Algorithm on a Grid

In this section, we present the key concepts in the adapted genetic algorithm as follows:

Encoding and Basics We use a $w \times h$ array matching the grid's size to store integers ranging from 0 to $K-1$ for any individual. A cell u positioned at column x and row y is denoted as $u(x, y)$. In any individual, two cells are considered "connected" only if they have a unit distance between them, so diagonal cells are not considered connected. A "region" $\Omega = \{u_1, u_2, \dots, u_k\}$ is defined as a set of cells of the same type connected as a whole. The term "fragment" usually denotes a small undesirable region. A "block" $B = \{u_1, u_2, \dots, u_{w_1 \times h_1}\}$ represents a rectangular area within the grid, encompassing $w_1 \times h_1$ ($1 \leq w_1 \leq w, 1 \leq h_1 \leq h$) cells. The illustrations are as Figure 2a.

Crossover Crossover is the operation where specific components of two chosen individuals are exchanged to generate two

Background

*You are a landscape architect who helps the user. The user is going to design the landscapes on a grid (for example, with 20*20 cells)...*

Problem Explanation

There are expected to be several types of terrain... These terrains has the following characteristics... For example, a lake refers to a body of water where bridges can be built and...

If a series of connected cells on the grid are all of the same type, they will be considered a region. The entire terrain may have multiple regions...

Task Specification

The user will next input a description of the landscape, and you need to extract certain parameters and restrictions regarding these aspects:

- 1. The user specifies which types of terrain must be present or must not be present.*
- 2. The user specifies the total coverage percentage for certain types of terrain.*

...

It is likely that the user's input does not cover some of the information above. In such a case, you... Your output needs to follow logical consistency, such as... If the user's input cannot be resolved, then...

Output Format

Your output must strictly follow the format described below:

The first 5 lines contain lists corresponding to ground cover types (0 to 4)...

The first line contains boolean values (0 for no, 1 for yes).

The second line contains tuples representing lower and upper limits...

The last line provides feedback to the user. If the input cannot be resolved...

Otherwise, output "OK".

Figure 1: A simplified version of the system prompt for parameterization. To enhance the LLM's comprehension and task completion, we segment the prompt into four parts.

new individuals. If the selected individuals are the same, no crossover is needed. Otherwise, we begin by randomly selecting a position (x, y) , where the two cells $u_1(x, y)$ and $u_2(x, y)$ at that position from different individuals are of different types, denoted as t_1 and t_2 , respectively. We then compute the regions Ω_1 and Ω_2 containing the two cells, respectively. By overlapping the two regions, we obtain the intersection part $\Omega_c = \Omega_1 \cap \Omega_2$. If the intersection part is sufficiently large, i.e., $\frac{|\Omega_c|}{|\Omega_1| + |\Omega_2| - |\Omega_c|} \geq \eta_\Omega$ (where η_Ω is a controllable threshold), then the regions Ω_1 and Ω_2 are considered "similar" to each other. In such a case, the contents of the two regions are exchanged, implying that all cells in Ω_1 are assigned type t_2 , while those in Ω_2 are assigned type t_1 . Otherwise, only the intersection part is exchanged. Figure 2c shows the crossover operation for the two cases. Generally, the adapted approach preserves the spatial continuity when exchanging the components.

Mutation Mutation is the operation wherein an individual undergoes specific changes in its content. We design two approaches to address this operation, as shown in Figure 2d. The first approach is to mutate a region. A region Ω is randomly selected, and all its cells are altered to another single type, preserving the structure of all existing regions. The second one is to mutate a block. A small block B is randomly selected at a random location, and all its cells are assigned a random single type. Although easy to execute, the second approach can potentially disrupt spatial continuity and result

ALGORITHM 1: The Genetic Algorithm

Output: The individual with the highest fitness value.

- 1 Initialize the population of N_p individuals;
- 2 $Generation \leftarrow 1$;
- 3 **repeat**
- 4 Evaluate all individuals using the fitness function;
- 5 Select N_p individuals as the new generation of population;
- 6 Perform **Crossover** operations with a rate of η_c ;
- 7 Perform **Mutation** operations with a rate of η_m ;
- 8 Perform **Evolution** operations with a rate of η_e ;
- 9 $Generation \leftarrow Generation + 1$;
- 10 **until** $Generation > N_g$;
- 11 Choose the individual with the highest fitness value throughout the entire process as the output;

in more fragments. However, introducing such variations to the regions can pave the way for better solutions.

Evolution Evolution, a novel operation proposed, actively modifies an individual to increase its likelihood of being selected into the next generation. Specifically, an individual after evolution is better evaluated by the fitness function, as manifested in Figure 2e. In essence, evolution serves as a positive form of mutation. Since most random mutations are detrimental to individuals, an evolution operation assists the population in fixing defects and cultivating improved solutions. The implementation is always based on the fitness function. In this paper, this operation prioritizes reducing fragmentation and altering undesirable cells or regions in most instances.

The procedure of the adapted genetic algorithm is outlined as Algorithm 1. The algorithm's parameters, including N_p , N_g , η_c , η_m , and η_e , are all adjustable.

A.4 Optimization of the Terrain

For a controllable and effective generation, the genetic algorithm's fitness function should adhere to both the parameters and general landscape design principles. Therefore, any violation of the parameters and the principles incurs penalties. To compute the fitness value $f(I)$ for any individual I , we begin by summing up a "total penalty value", denoted as $\rho(I)$, which comprises several components as listed below:

- (1) **The existence penalty**, denoted as $\rho_e(I)$. Aligning with the parameter P_e , the value depends on each region type's correct or incorrect existence in I . For instance, if the user specifies no lakes (i.e., $P_e(Lake) = False$), but "aquatic" cells are present in I , then $\rho_e(I)$ is significantly increased.
- (2) **The numeracy penalty** $\rho_N(I)$, which is controlled by P_N . When the count of regions in I for a specific type does not fall within the range decided by the parameter, the penalty accumulates. The deviation from the standard range influences the penalty value.
- (3) **The area penalty** $\rho_A(I)$, aligned with both P_A and P_a . Like $\rho_N(I)$, the penalty accumulates whenever the area coverage

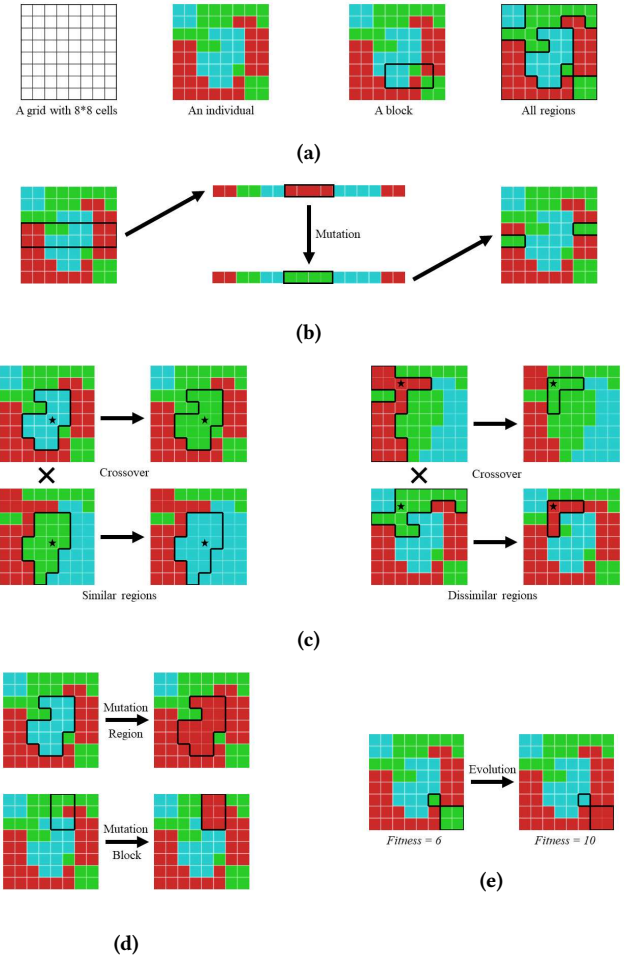


Figure 2: Examples and illustrations for the genetic algorithm. (a) The basic concepts. (b) The grid cells can be flattened using a row-wise representation. However, when operating on one-dimensional continuous cells, more fragments may be created in the grid due to discontinuity in a two-dimensional representation. (c) The crossover operation on two cases: When the selected regions are similar, the contents are exchanged; otherwise, only the contents of the intersection part are exchanged. (d) The two approaches for the mutation operation. Contents either in a region or a block are switched as a whole. (e) The novel "evolution" operation that increases the fitness value. The implementation depends on the fitness function.

rates deviate outside the specified range. Both the total coverage rate for all regions and the coverage rate for any single region contribute to the penalty.

- (4) **The location penalty** $\rho_L(I)$, influenced by P_L . Each region is assigned one or more enumerable labels representing its approximate locations (e.g., *Center* and *BottomLeft*), and the penalty accumulates if none of these labels match the parameter specifications. "Aquatic" and "elevated" cells incur

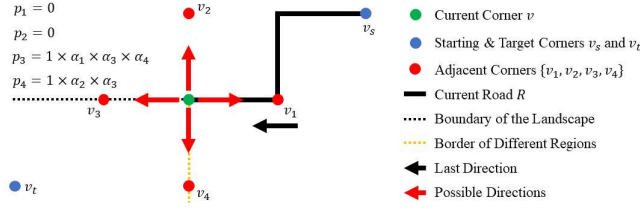


Figure 3: An example of deciding the probabilities (before normalization) in a single iteration of Algorithm 2. The probabilities are adjusted based on the principles described in Section 4.2.4. An edge that follows the principles is more likely to be selected and added to the road.

higher penalties due to their distinctiveness. Additionally, landscape principles are incorporated into the evaluation. For instance, an "aquatic" region too close to the landscape boundary is penalized.

- (5) **The compatibility penalty** $\rho_C(I)$ that is unrelated to any parameter. Governed by the landscape design principles, it evaluates the type compatibility of adjacent regions. For instance, an "elevated" region adjacent to an "aquatic" region is considered appropriate. However, if a lake surrounds the highland, the penalty value will accumulate due to incompatibility.

In the evaluation, all the above components yield non-negative values. Upon computing the penalties, the total penalty value $\rho(I)$ and the fitness value $f(I)$ are given by Equation 1 and Equation 2. In each iteration of the algorithm, the N_p individuals for the next generation are selected using a classic Roulette wheel selection approach [1]. For a population $\{I_1, I_2, \dots, I_{N_p}\}$ with corresponding fitness values $\{f(I_1), f(I_2), \dots, f(I_{N_p})\}$, the probability of selecting any individual I_i is $\frac{f(I_i)}{\sum_{k=1}^{N_p} f(I_k)}$. The selection process involves randomly choosing the individuals one by one (with repeated selection allowed) until the desired N_p individuals are selected.

$$\rho(I) = \rho_\epsilon(I) + \rho_N(I) + \rho_A(I) + \rho_L(I) + \rho_C(I) \quad (1)$$

$$f(I) = \frac{100}{\rho(I) + 1} \quad (2)$$

A.5 Optimization of Spots and Roads

The randomized heuristic approach to generate primary roads is outlined in Algorithm 2. The algorithm operates by repeatedly determining the next corner. In each iteration, an equal probability of being selected is initially assigned to each corner among the four possible directions. However, the probabilities are multiplied by several hyperparameters or set to zero based on different cases. Eventually, edges adhering to the principles (described in Section 4.2.4 of the paper) are more likely to be selected as the road. As an example, we present Figure 3 to illustrate the key ideas in the algorithm.

This road generation approach balances exploration and exploitation, potentially resulting in satisfying roads. However, as a randomized approach, it may also lead to poor outcomes. Therefore, we

ALGORITHM 2: The Randomized Heuristic Path-Finding Approach

Input: The starting corner v_s and the target corner v_t .

Output: The road R connecting v_s and v_t .

```

1  $R \leftarrow \{\}$ ;
2  $V \leftarrow \{v_s\}$  // Visited corners;
3  $v \leftarrow v_s$  // Current corner;
4  $d \leftarrow -1$  // Last direction;
5 repeat
6   Determine the set of four adjacent corners
7    $V_a \leftarrow \{v_1, v_2, v_3, v_4\}$ ;
8   for  $i \in \{1, 2, 3, 4\}$  do // Handle each adjacent corner
9      $p_i \leftarrow 1$  // Probability of being selected;
10    if  $v_i \in V$  then // Already visited
11       $p_i \leftarrow 0$ ;
12    end
13    if  $v_i$  is outside the landscape then
14       $p_i \leftarrow 0$ ;
15    end
16    if  $(v, v_i)$  goes along the boundary of the landscape then
17       $p_i \leftarrow \alpha_1 p_i$ ;
18    end
19    if  $(v, v_i)$  goes along the border of two regions then
20       $p_i \leftarrow \alpha_2 p_i$ ;
21    end
22    if  $(v, v_i)$  goes towards  $v_t$  then
23       $p_i \leftarrow \alpha_3 p_i$ ;
24    end
25    if  $d == i$  then // Go straight
26       $p_i \leftarrow \alpha_4 p_i$ ;
27    end
28     $s \leftarrow p_1 + p_2 + p_3 + p_4$ ;
29    for  $i \in \{1, 2, 3, 4\}$  do // Normalize
30       $p_i \leftarrow \frac{p_i}{s}$ ;
31    end
32    Randomly select a number  $k$  from the set  $\{1, 2, 3, 4\}$ , where
33      each number  $i$  has a probability of  $p_i$  of being chosen;
34     $d \leftarrow k$ ;
35     $R \leftarrow R \cup \{(v, v_d)\}$ ;
36     $V \leftarrow V \cup \{v_d\}$ ;
37     $v \leftarrow v_d$ ;
38  until  $v = v_t$ ;

```

employ the select-best structure. In doing so, we generate multiple solutions and select the best one. The evaluation score $E(R)$ for road R is formalized as Equation 3, where $|R|$ is the length of the road and $T(R)$ is the number of left/right turns in the road. Each edge e is assigned a value $E(e)$, specifically, β_1 for boundary edges of the landscape, β_2 for borders of regions, and 0 otherwise. The solution with the highest $E(R)$ is selected as the final result.

$$E(R) = -1.5(|R| + T(R)) + \sum_{e \in R} E(e) \quad (3)$$

The roads to connect all POIs and entrances are generated repeatedly using Algorithm 2. In practice, already-generated roads can impact the process of connecting more corners. We use a union-find

set to record the connected corners to address this. Corners connected by existing roads are placed in the same union. Therefore, in the iteration process of Algorithm 2, the termination condition can be modified to " v and v_t are in the same union", indicating that v and v_t are already connected by existing roads.

With only the primary roads, the tourists may only have access to a few regions and scenic spots. Secondary roads, serving as supplements to the primary roads, are crucial for enhancing the touring experience. Unlike addressing primary roads, we use the following process to determine secondary roads: First, we designate all non-primary-road region borders as secondary roads. Subsequently, we repeatedly divide the largest region into two parts and add the dividing edges to secondary roads until the number of edges reaches the complexity specified by parameter $P_E(\text{RoadComplexity})$. Each region division employs a growth-based approach, prioritizing relative balance while incorporating controlled randomness. Any value (extent) of parameter $P_E(\text{RoadComplexity})$ is mapped to a rate r ($0 < r < 1$), which represents the number of road edges divided by the total number of available edges in the grid. For example, a $P_E(\text{RoadComplexity}) = \text{Extent.Low}$ is mapped to $r = 0.3$, and Extent.High is mapped to $r = 0.4$.

A.6 Optimization of Attributes

The genetic algorithm in this step uses similar configurations to that in Section A.4. We apply the same approach to calculate the fitness value for each individual, with modifications to the criteria for the penalty value $\rho'(I)$ (some of which operate similarly, thus descriptions are omitted):

- (1) **The existence penalty** $\rho'_E(I)$.
- (2) **The numeracy penalty** $\rho'_N(I)$.
- (3) **The area penalty** $\rho'_A(I)$.
- (4) **The location penalty** $\rho'_L(I)$. Same as the one in Section A.4, the value is influenced by the parameter P_L , but no additional restrictions for the location of specific types are applied here.
- (5) **The compatibility penalty** $\rho'_C(I)$, which serves as a crucial component guided by landscape design principles. Firstly, we assess the attribute's compatibility with the terrain, where every cell with inappropriate combinations is penalized. For example, "tall-growing" and "aquatic" are incompatible types, but "tall-growing" and "elevated" match perfectly. Next, we assess the attribute's compatibility with the key spots, where we only consider cells near the entrances and POIs. For example, among the cells with an Euclidean distance of less than 3 to a POI, an "architectural" cell is required. Another case is that both "tall-growing" and "architectural" cells are not recommended near an entrance due to the reservation of open space.

After evaluating all the abovementioned components, we compute $\rho'(I)$ and the fitness value $f'(I)$ similar to Equation 1 and Equation 2. Other components of the genetic algorithm, including the selection policy, remain unchanged.

A.7 Smoothing

For the vertical aspect of the smoothing, we generate a continuous height map that specifies the physical height of the ground at any location (with coordinates not necessarily being integers). Firstly,

every cell u is assigned a relative height $h(u)$ ranging from 0 to 1 according to its terrain type:

- Every "aquatic" cell is assigned $h(u) = 0$. To simplify, we stipulate that all water surfaces have a unified height of $H_W > 0$, so that they can cover all "aquatic" cells.
- "Terrestrial" and "artificial" cells are both assigned approximate heights slightly higher than H_W . Random values are introduced for a little variance in the heights, formally, $h(u) = H_G + \mu_T \text{RandomFloat}(0, 1)$ for terrestrial cells and $h(u) = H_G + \mu_A \text{RandomFloat}(0, 1)$ for artificial cells. Here, H_G represents the basic height of the ground, and μ_T and μ_A are small threshold ratios for the variation.
- "Elevated" cells are classified into three layers. A cell in the i th ($i \in \{1, 2, 3\}$) layer is given a relative ratio $t = \frac{i-1+\text{RandomFloat}(0,1)}{3}$, then $h(u) = (1-t)H_G + t$, indicating an interpolation of H_G and 1 with ratio t . In a region full of "elevated" cells, interior cells are of higher layers. Specifically, cells adjacent to the boundary are of layer 1, cells adjacent to the layer 1 cells are of layer 2, and the remaining cells are of layer 3. This layering process ensures a smooth climb from the bottom to the top and allows for elevations among different parts of the highland.

When computing heights, each cell can be represented by its center point, as shown in Figure 4. The simplest way to obtain a continuous height map is to perform bilinear interpolation on the two-dimensional grid. However, as the gradients among adjacent cells may vary significantly, there can be obvious artifacts at the positions of the edges and corners in the original grid. Another approach is to apply bicubic interpolation [2], which introduces curves of higher degrees and results in a much smoother map. However, we observe that applying bicubic interpolation directly can lead to changes in height values at the cell center points, especially near the water and the highlands, e.g., an aquatic cell is lifted weirdly. Therefore, before interpolation, we perform an upscaling operation by computing the heights of corners and edge midpoints. As illustrated in Figure 4, the height of a corner equals the mean of four adjacent cells' heights, which is a special case of bilinear interpolation. The heights of edge midpoints are computed in a similar manner. Given a grid with $w \times h$ cells, the upscaling operation results in a new grid comprising $(2w+1) \times (2h+1)$ corners. Performing bicubic interpolation on the new grid allows the original assigned heights to be better preserved while achieving sufficient smoothing. For a more natural terrain elevation, we add Perlin Noise to the resulting height map scaled by μ_P , i.e., $\Delta h = \mu_P(\text{PerlinNoise}(0, 1) - 0.5)$ [5]. The resulting height map is truncated to ensure all values range from 0 to 1, represented by a grayscale image. When eventually generating the landscape scene, the relative height is mapped linearly to a range from 0m to $P_S(\text{MaxHeight})$, where $P_S(\text{MaxHeight})$, corresponding to $h(u) = 1$, is a parameter specified by the user.

Horizontally, the smoothing is applied to each region border, which comprise a set of edges stretching along the axes' directions and connecting the corners. The idea is to introduce curves and insert more points. Firstly, we add a slight Gaussian displacement to each corner. Then we identify the control corners, which are corners connected to more than two edges. Each segment, defined as the portion between two control corners, constitutes a sequence of

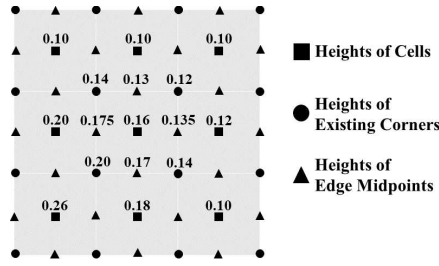


Figure 4: The illustration of the upscaling operation for computing heights. Each cell is represented by its center point (rectangle), with the height already assigned. The heights of corners and edge midpoints are computed through bilinear interpolation. These points are then regarded as the corners of a new grid for subsequent interpolation. If the original grid contains $w \times h$ cells, the new grid will have $(2w+1) \times (2h+1)$ corners.

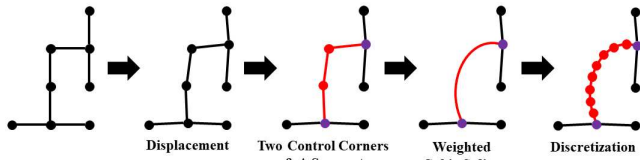


Figure 5: The process of smoothing the regions' borders. Given a set of corners and edges, we start by adding slight displacements. Then we identify the control corners and segments. Lastly, we apply a weighted cubic spline for smoothing and perform the discretization by inserting points.

adjacent corners. For each segment, we employ a cubic spline to fit all corners. The starting and ending corners are given significantly higher weights than others to ensure the spline passes through these two corners. Finally, we insert new points along the spline and connect them to form the new segment. Figure 5 illustrates the above process.

A.8 Procedural Arrangement of Landscape Elements

Here, we list the cases for instantiating the zone borders:

- **Non-road borders:** No instantiation is required for borders that are not roads.
- **Site boundaries:** Borders serving as the boundary for the entire site are instantiated with walls. Entrances are reserved for clear pathways.
- **Roads across lakes:** Borders functioning as roads across lakes are treated as bridges. When the bridges are too short to connect the lands, the terrain along the border is uplifted to connect the remaining parts.
- **Ground roads:** Borders functioning as roads on the ground are instantiated as road bricks. Widths for primary roads and secondary roads are both controlled by $P_S(\text{RoadWidth})$.

Figure 6 shows the ideas of the two kinds of rules and three kinds of patterns. To provide further details:

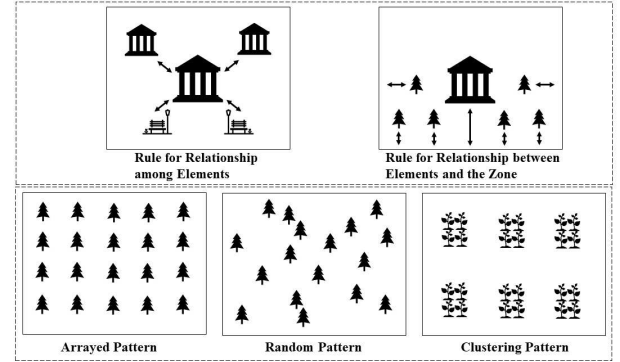


Figure 6: Rules and patterns for the procedural arrangement of elements. The rules concentrate on the spatial relationship (both positional and orientational) among distinct models and between models and the zone. Meanwhile, the patterns emphasize various structures for arranging a multitude of replicated models.

- The rule for **relationship among elements**: Arrange two or more models as a group, specifying the positional or orientational differences between any two models based on recognized landscape practices or standards. For example, a chamber may be accompanied by two statues at the front with specific positions and orientations.
- The rule for **relationship between elements and the zone**: Place a model in a specific location within the zone (e.g., central) or arrange it to have a specific spatial relationship (positional and orientational) with the zone's boundary. For example, a hall should be positioned neither too far nor too close to the main road [3].
- The **arrayed pattern**: Arrange a sequence of models regularly in an array-like structure within the zone. The density is affected by both the parameter P_E (e.g., $P_E(\text{TreeDensity})$) and the model type (e.g., trees occupy more space than shrubs). However, strict regularity in the arrangement is not mandatory. An alternative approach is to introduce random displacement (e.g., Gaussian) in the positions. Another method is to establish a grid within the zone and position each model randomly within each cell.
- The **random pattern**: Randomly distribute some models within the zone, with the density influenced by both the parameter P_E and the model type. In certain instances, the models are permitted to overlap, especially for basic and low-growing elements such as grass, flowers, and rocks. However, overlaps are strictly prohibited in significant architectural elements like buildings.
- The **clustering pattern**: Organize some models into clusters within the zone. In most cases, the density of clusters is controlled by the parameter P_E , and their properties (size, number of models, etc.) are determined by the model type. This pattern is beneficial for creating scenic spots by grouping attractive elements. Rocks, in particular, are suitable to be arranged by piling and clustering [4].

Based on the rules and patterns, we further present ideas for organizing the models as below:

- **Trees:** As crucial components in a landscape, trees play a vital role in many combined types. In most cases, we apply both the **random pattern** and the **clustering pattern** for the arrangement. However, specific **relationships** with some artificial elements such as pavilions may be added for additional constraint.
- **Shrubs:** As common elements in a landscape, these models can be arranged using all three patterns depending on their functionalities. For example, in small zones, they are often grouped with **arrayed** or **clustering** patterns. In large zones, however, they can be **randomly** distributed alongside trees.
- **Rocks:** As mentioned earlier, these models are preferably placed in **clusters**. An exception is that in some lakes, a few rocks are positioned at **specific locations within the zone** to serve as attraction spots [4].
- **Buildings** (e.g., chambers and pavilions): In most cases, the arrangement of such models adheres to certain **rules**. However, we permit a **random** distribution of a few smaller-scale buildings.
- **Lotus and various flowers:** The arrangement of these models depends on the environment. When occurring naturally, they are **randomly** placed. Otherwise, they can be arranged in an **array** or **clusters**.
- **Grass and other low-growing plants:** As they typically grow naturally in random places, we simply utilize the **random pattern** to arrange them.

In the arrangement, relevance among various elements within each zone is ensured by the proposed rules, such as spatial relations for multiple elements. With the arrangements completed, the landscape generation is concluded, and the result is outputted as files containing all necessary information for constructing the landscape scene. Specifically, two images are generated, one for the height map and the other for the texture of the terrains. Additionally, a text file is provided, specifying information about all models, including their position, orientation, and scale.

B SUPPLEMENTS FOR EXPERIMENTS

B.1 Results and Setup

We implement our framework using Python 3.8. Since the generation processes of different landscapes are independent, we leverage 20-core CPU parallel computing for acceleration. The hyperparameters used in Section A are detailed in Table 2. The results of the landscape generation framework are presented using Unity version 2022.3.14f1c1. A total of 100 rendered images are presented in Figure 7 and Figure 8.

B.2 Influence of the User Input

We test whether the LLM can determine the existence of the lake (i.e., $P_e(Lake)$) correctly. The inputs and corresponding results, after a hundred repeated queries each, are presented in Table 1. The second column in the table indicates the percentage of $P_e(Lake)$

explicitly set to *True*, whereas the third column shows the percentage of $P_e(Lake) = False$. The percentage of the default value used can be computed by subtracting these two percentages from 1.

Although the first three inputs all indicate the presence of lakes, only the first input achieves rather satisfying results. For the second input, the LLM may be disturbed by additional information, including nouns such as "hills" and "forests", and adjectives/adverbs such as "dense" and "partially". For the third input, the LLM may fail to comprehend that an "island" implies the presence of a lake. The fourth input does not include decisive information about a lake but suggests that a lake is more likely not to exist. In this case, the LLM comprehends it well and provides accurate responses. The last input explicitly states that no lakes can be present. However, for more than 30% of all trials, the LLM is wrong or not sure about the answer. This may be attributed to the instability of the LLM itself and the neglect of negative wording. Overall, the model's performance can be improved. Factors such as the design of the context and the capabilities of the LLM may both be accounted for.

B.3 Effectiveness of the Genetic Algorithm

The classic 1D implementation used for comparison is described as follows: Each individual is encoded by a string of integers, which is the row-wise flattened representation of the grid. After randomly selecting a position, the crossover operation swaps the subsequent components between the two parents to create the offspring. The mutation operation randomly selects a short segment in the string and assigns the same new value to override the existing values. The evolution operation does not exist in this implementation.

B.4 Effectiveness of the Optimization of Roads

This part compares the proposed road generation algorithm with a baseline approach. The evaluation is based on three metrics: the **number of terrain regions** (split by roads), the **relative standard deviation of the region areas** (i.e., standard deviation divided by mean), and **road complexity** (i.e., the number of road edges divided by the total number of edges). Five hundred trials are tested for both approaches and the means are computed.

The results are summarized in Table 3. Compared to the baseline, our heuristic approach leads to over 20% fewer number of regions, significantly reducing the disruption to the structure of regions. Additionally, our approach results in a slightly smaller relative standard deviation concerning the areas of all regions, indicating a more balanced separation of areas. However, since the baseline typically generates the shortest roads among the corners, the roads produced by our approach are slightly longer. We conclude that our approach effectively contributes to a more balanced region structure while generating roads of acceptable lengths.

B.5 User Study

Here, we present the explanations to all evaluation criteria:

- **Degree of Ecological Diversity (D):** Abundance of the variety of plant species tailored for ecological stability, and adherence to the ecological characteristics of natural vegetation; Ambiance and aesthetic representation of nature in the landscape, aligned with the principles of situational

Table 1: Inputs and results for testing the LLM’s interpretation, focusing on the scenario related to the existence of lakes. After a hundred queries, we calculate the percentages of different types of responses. The responses are considered appropriate only for the first and the fourth input. In other cases, the LLM may fail to accurately comprehend the inputs’ intentions. In general, there is room for improvement in performance.

Input Text	Percentage of $P_e(Lake) = True$	Percentage of $P_e(Lake) = False$
"The landscape has two lakes."	86%	2%
"The site has a lake partially surrounded by hills and dense forests."	59%	9%
"The landscape features an island and two valleys."	43%	8%
"The site is mostly covered by trees."	15%	29%
"The landscape has no lakes."	13%	69%

Table 2: The hyperparameters used for implementing the method.

Name	Value	Brief Description
D	20	Real length (m) for a unit distance in the grid
η_Ω	0.6	Threshold for judging similarity in crossover
N_p	100	Population size in the genetic algorithm
N_g	100	Maximum generation in the genetic algorithm
η_c	0.9	Crossover rate in the genetic algorithm
η_m	0.7	Mutation rate in the genetic algorithm
η_e	0.1	Evolution rate in the genetic algorithm
α_1	$\frac{1}{5}$	One of the multiplication rates in path-finding
α_2	2	One of the multiplication rates in path-finding
α_3	3	One of the multiplication rates in path-finding
α_4	3	One of the multiplication rates in path-finding
β_1	-5	Edge value for landscape boundaries
β_2	2	Edge value for region borders
H_W	0.08	Relative height of water surfaces
H_G	0.1	Relative height of grounds
μ_T	0.025	Scale of randomness for "terrestrial" cells
μ_A	0.01	Scale of randomness for "artificial" cells
μ_P	0.02	Scale for Perlin Noise

Table 3: The results of the ablation study for generating the roads. Although our method produces roads that are slightly longer, it significantly reduces the disruption to the region structure and results in more balanced region areas.

Metric	Baseline	Ours
Number of Regions	20.8	16.3(↓ 21.6%)
Relative Standard Deviation of Areas	0.958	0.891(↓ 7.0%)
Road Complexity	0.0926	0.0998(↑ 7.8%)

landscaping; Rational arrangement of waterscapes, considering slope and spatial proportions in relation to the overall landscape.

- **Adaptability Based on Local Conditions (A):** Seamless integration of buildings and plants with the topography, emphasizing harmony and adaptability to local conditions; Consideration of the relationship among water sources, mountainous terrain, and buildings to seamlessly blend architecture with the natural surroundings.
- **Management of Spatial Sequences (S):** Design of the overall landscape layout that ensures a cohesive flow of elements, complementing other attractions and creating a harmonious unity; Emphasis on the coherence of spatial sequences, creating depth through vivid layouts and elevation changes; Use of miniature landscapes to express panoramic views, combining majestic features and intricate details for picturesque scopes.
- **Presentation of Visual Richness (R):** Design of paths with rich access to landscapes, ensuring a visually engaging experience for visitors; Use of appropriate curvature in path design to enable frequent changes in perspective, achieving a dynamic visual experience with every step.
- **Application of Landscaping Techniques (T):** Enhancement of complementarity between indoor and outdoor views, effectively expanding the overall viewing visibility. Emphasis on hierarchy organizing to create harmony between distant and near views.
- **Minimization of Artificial Traces (N):** Conveying natural-ity through appropriate orientation and natural lighting of buildings, along with the creation of openness through a suitable density of buildings. Adherence to natural topography in the layout of scenic spots to minimize artificial traces.

REFERENCES

- [1] Tobias Blickle and Lothar Thiele. 1996. A comparison of selection schemes used in evolutionary algorithms. *Evolutionary Computation* 4, 4 (1996), 361–394.
- [2] Robert Keys. 1981. Cubic convolution interpolation for digital image processing. *IEEE transactions on acoustics, speech, and signal processing* 29, 6 (1981), 1153–1160.
- [3] Dunzhen Liu. 1979. *Classical Gardens of Suzhou*. China Architecture & Building Press.
- [4] Liqiong Ouyang, Bo Zhang, and Fan Fu. 2015. The Differences in Choosing Stones Among <Yuan Ye> <Superfluous Things> and <Xian Qing Ou Ji>. *Huazhong Architecture* 33, 9 (2015), 155–158.
- [5] Ken Perlin. 1985. An image synthesizer. *ACM Siggraph Computer Graphics* 19, 3 (1985), 287–296.

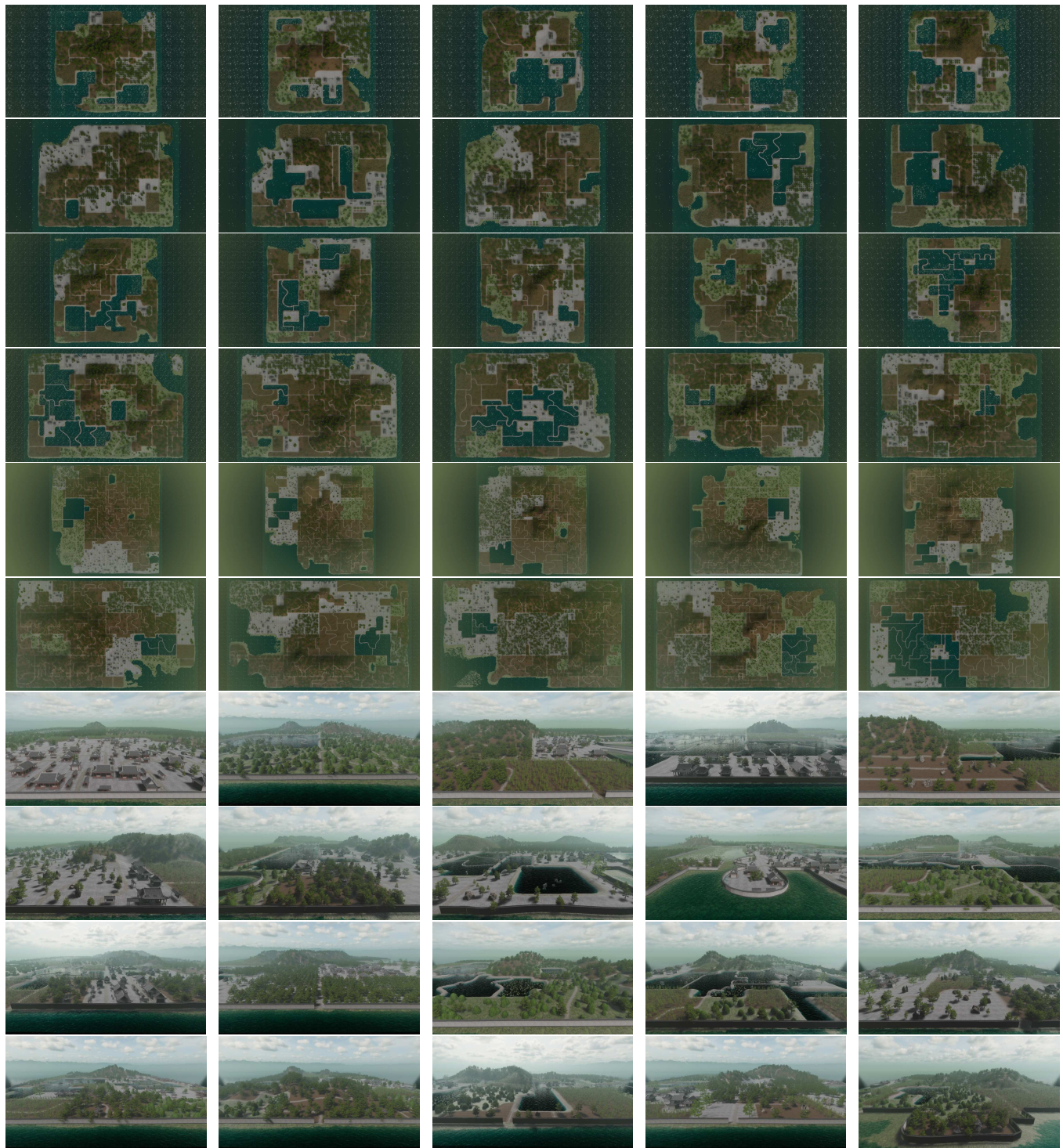


Figure 7



Figure 8