GardenDesigner: Encoding Aesthetic Principles into Jiangnan Garden Construction via a Chain of Agents

Supplementary Material

1. Implementation Details

1.1. Terrain Generation Algorithm

To generate the garden's terrain, we adopt the genetic algorithm on a 2D grid. We initialize the terrain with a random integer from 0 to 3, representing unused area, water area, land area, and ground area. Roads are selected and combined from the grid borders, which are scored through border positions. Roads will be smoothed through spline curving and choosing the intersection or inflection point as the start and end of the curve. The CLIP model we used to evaluate CLIP-Score is openai/clip-vit-base-patch32 [3].

1.2. Hierachical Garden Composition

In the implementation process, we divide the Chain of Aesthetic Principles into terrain and structure generation, using a genetic algorithm. Firstly, the detailed prompts of the **Terrain Distribution Agent ($\mathcal{A}_T$)** are presented in Figure 6. Additionally, we also provide the detailed prompts of the **Road Generation Agent ($\mathcal{A}_R$)** in Figure 7. Based on the response from LLM, we parse the parameters for 2D genetic alogrithm to generate terrain and structures. We choose four types of terrains to simulate the landform of Jiangnan garden: *Outside*, *Waterbody*, and *Land*. Specifically, each terrain is explained as follows:

- **Outside** areas refer to unoccupied zones and serve to increase spatial diversity and boundary complexity.
- **Waterbody** areas present the indispensable and symbolic water area of the Jiangnan garden.
- **Land** areas represent flat land with natural elements.
- **Ground** areas are the flat terrain zone, on which the buildings, plants, and rocks will be sited.

In each terrain grid cell, we employ an integer number (0-3) to represent these terrain types.

1.3. Knowledge-embedded Asset Arrangement

To obtain an appropriate garden layout, we decompose the Garden Configuration into object selection and constraint setting. We present the detailed prompt used to select objects for **Asset Selection Agent ($\mathcal{A}_S$)** in Figure 8. Before requesting LLM, we annotate each area with area information. We use the file search tools from OpenAI. After selecting the appropriate objects, we also present the **Layout Optimization Agent ($\mathcal{A}_C$)** prompt in Figure 9 and Figure 10 , enabling the feasible constraints for objects. All constraints are formalized in a structured representation to ensure interpretability and implementation feasibility:

```
"area name": {
    "object name": [
        ["constraint", "type"],
        ["constraint", "rel object", "type"]
    ]
}
```

where the "area name" and "object name" are the target area and object, the "constraint" is the relationship between object and another object with the name of "rel object", and the "type" is the constraint type.

Algorithm 1 Garden Construction

Require: User input text U, Garden Principles K_{global}, Asset library O_{asset} with knowledge K_{a}
Ensure: Complete garden $G = (T, R, (O_{\text{s}}, P)$

$T \leftarrow A_T(U, K_{\text{global}})$
$R \leftarrow A_{\text{R}}(S(T, e_{i,j}), U, K_{\text{global}})$
$O_{\text{s}} \leftarrow A_{\text{S}}(Q((V(K_{\text{a}}), o_i), U), I_{\text{area}})$
$C \leftarrow A_C(Q((V(K_{\text{a}}), o_i, o_j), U))$
for each position p_{anchor} for o_{anchor} **do**
  $P_{\text{temp}} \leftarrow \{p_{\text{anchor}}\}$
  if DFS-Place$(O_s, o_{\text{anchor}}, C, P_{\text{temp}})$ **then**
    if $L_{\text{opt}}(P_{\text{temp}}) < L_{\text{opt}}(P)$ **then**
      $P \leftarrow P_{\text{temp}}$
    end if
  end if
end for
return $G = (T, R, (O_{\text{s}}, P))$

1.4. Optimization

We demonstrate the detailed information in Optimization section. We utilize Depth-First Search (DFS) Solver to optimize object constraints from Garden Configuration inspired by Yang et al. [5]. To make the balance between time and quality, we choose to change the area into grid point according to the area bounding box and we also remove the points of the area. The grid points in the area are presented as the solution for each object position movement. In the DFS solver, each object is characterized by five variables: $(x, y, l, w, rotation)$, where: (x, y) represents the 2D coordinates of the object's center, l and w denote the length and width of the object's 2D bounding box. Rotation can take one of four possible angles: 0, 90, 180, or 270, where 0 is forward positive z-direction. The solver applies soft constraints, permitting minor violations to facilitate feasible layout generation. Apart from object constraint, we also hard constraints are enforced to ensure physically valid

placements: (1) No object collisions, objects must not overlap; (2) Area boundaries, objects must stay within the designated space. If an object violates any hard constraint, it is rejected from the current layout. We calculate the overall loss of each objects in validate solution, and select the most feasible solution with lowest loss after 100 iteration steps.

2. GardenVerse Details

GardenVerse comprises 132 high-quality artistic 3D assets across three canonical categories: Rock (33), Plant (44), and Architecture (54), in Figure 1. In Jiangnan gardens, the combination of plants and rocks stands out as a distinctive feature compared to standalone assets, which creates a harmonious interplay between organic vitality and enduring solidity. It includes both individual elements (40.2%) and pre-composed arrangements (59.8%) of plants and rocks, enabling flexible retrieval of Jiangnan gardens, in Figure 2.

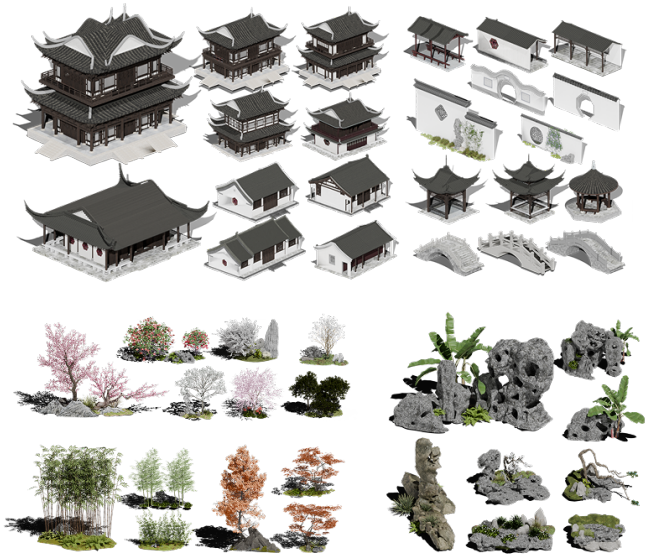


Figure 1. GardenVerse data examples. The GardenVerse consists of four types of objects: (a) the Architecture, (b) the Structure, (c) the Plant, and (d) the Rock. The plant and rock include both single and combined objects.

```
{
    "name": "object name",
    "path": "related path",
    "pos": "appropriate position",
    "object": "internal object",
    "season": "appropriate season",
    "description": "knowledge about object",
    "minp": "min position",
    "maxp": "max position",
    "size": "object size"
}
```

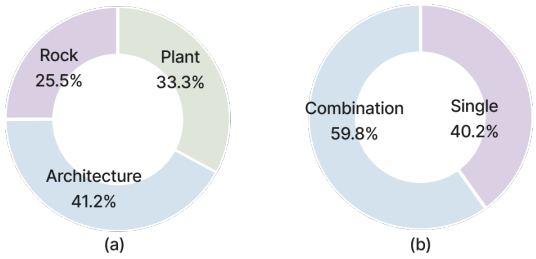


Figure 2. GardenVerse statistics: (a) the object categories proportional distribution; (b) the combined and single objects ratio.

where the text of "description", "season" and "pos" constitute the asset garden knowledge to guide the asset selection and garden layout optimization.

3. Experiments

We conducted the ablation study in Figure 4(a). The baseline method has worse visual quality than other methods with GardenVerse. The methods with terrain loss and explorative road scoring function have water-centric terrain and reasonable pathway. Furthermore, we evaluate the diversity of GardenDesigner. In Figure 4(b), we input the same prompt and evaluate the diversity of GardenDesigner to generate different Jiangnan gar- dens. In Figure 4(c), we evaluate the object layout diversity through maintaining the same prompt, terrain, and structure layout.

Table 1 shows that the comparison with natural scene generator Infinigen [4], rule-based SceneX [6], diffusion-based NuiScene [1], and PCG method without chain of agents. GardenDesigner outperforms others on all consistency and aesthetic metrics, validating the effectiveness. Based on the performance of baseline and PCG methods, both LLM-based reasoning and procedural engineering contribute to the improvement.

Table 1. Quantitative comparison with different methods.

Method	CLIP-S ↑	CLIP-A ↑	VLM-S ↑	QA-Quality↑
Infinigen	18.1	51.6	6.3	24.9
SceneX	23.1	53.1	5.5	37.6
NuiScene	25.6	53.7	10.3	46.3
PCG	27.4	53.9	28.9	51.2
Ours	**27.6**	**54.2**	**32.5**	**53.8**

We conduct additional loss ablation study to better understand the contribution of different losses. Table 2 shows that all losses will affect the garden structure complexity and visual quality, while global and distance losses contribute significantly to visual quality in VLM-S and QA-Quality. The weights for loss components are defined by their contribution to the final performance.

Figure 3. GardenVerse dataset. GardenVerse includes a collection of diverse 3D objects specially designed for Jiangnan gardens, and we show object examples from the dataset. GardenVerse encompasses four distinct object categories: architecture, plant, and rock, containing single objects and combination asset forms.

Table 2. Ablation study results on optimization losses.

Method	FD	VLM-S ↑	QA-Quality ↑
w/o L_{glo}	1.39	31.8	48.9
w/o L_{pos}	1.38	32.2	50.9
w/o L_{dis}	1.38	31.9	48.6
w/o L_{ali}	1.40	32.3	49.6
w/o L_{rot}	1.36	32.3	50.3
Ours	**1.36**	**32.5**	**53.8**

4. Human Evaluation Details

4.1. Human Evaluation Setup

We also invite 11 garden experts and 32 non-expert volunteers to evaluate the aesthetic quality of the generated Jiangnan gardens in Figure 5. We prepared 20 Jiangnan gardens for human evaluation, comprising five types of garden: (1) Normal, (2) Hydric, (3) Floral, (4)Arch-dense , and (5) Mazy. We ask the volunteers to choose which Jiangnan garden is better based on four perspectives: (1) **Overall Quality**: which method has the best overall quality? (2) **Text Relevance**: Which method has the highest alignment with the text? (3) **Spatial Layout**: Which method achieves the most accurate terrain and object layout? (4) **Cultural Atmosphere**: Which method best captures the cultural essence of Jiangnan gardens? For experts, we add more detailed questions. For Spatial Layout, we add two questions: (1) Which method results in the most reasonable and natural terrain layout? (2) Which approach provides the most logical and organic arrangement for vegetation and structures?. For Cultural Atmosphere, we also add two questions: (1) Which method best aligns with the design principles of Jiangnan gardens? (2) Which method best captures the poetic essence and philosophical depth of Jiangnan gardens?

4.2. Human Evaluation Results

Humans prefer GardenDesigner over baseline. Humans prefer the gardens generated from GardenDesigner compared to other methods, with a majority of selection, especially Cultural Atmosphere (49% General Users, 59% Experts). Overall Quality (49% General Users, 58% Experts), Text Relevance (49% General Users, 64% Experts), Spatial Layout (45% General Users, 57% Experts) and Cultural Atmosphere (49% General Users, 59% Experts).

GardenVerse promote the whole garden quality. The baseline method receives few selection compared to other methods adopted with GardenVerse. The selection ratios for all the question in General Users and Experts are all under 10%. On the contrary, the baseline method gets more preference using the GardenVerse datasets, especially among General Users. It gets 22% more selection ratio than the baseline method [2], validating the effects of GardenVerse.

Layout with aesthetic rules gets more preference. We also conducted an ablation study about Garden Configuration. We modify GardenDesigner by removing the Garden

Configuration module. Although two scenes have the same terrain and structure layout, the Humans prefer GardenDesigner more, indicating that Garden Configuration plays a significant role in determining scene quality.

5. Applications

To visualize the garden, we provide the plugin in Unity, where the user can edit and interact in real time. The output of GardenDesigner are stored as files containing all necessary information: the height map for terrain generation, textures to distinguish different terrains, and object information in json format, in which all of these constructs the Jiangnan Garden. We also develop Unity plug-in is developed to parse these files and convert them into terrain and objects. Additionally, we also provide the terrain adjustment tool to modify the terrain boundary and output the structure map to assist garden design and building.

References

[1] Han-Hung Lee, Qinghong Han, and Angel X. Chang. Nuiscene: Exploring efficient generation of unbounded outdoor scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 26509–26518, 2025. 2

[2] Jia-Hong Liu, Shao-Kui Zhang, Chuyue Zhang, and Song-Hai Zhang. Controllable Procedural Generation of Landscapes. In *Proceedings of the 32nd ACM International Conference on Multimedia*, page 6394–6403, New York, NY, USA, 2024. Association for Computing Machinery. 3

[3] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning Transferable Visual Models From Natural Language Supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021. 1

[4] Alexander Raistrick, Lahav Lipson, Zeyu Ma, Lingjie Mei, Mingzhe Wang, Yiming Zuo, Karhan Kayan, Hongyu Wen, Beining Han, Yihan Wang, Alejandro Newell, Hei Law, Ankit Goyal, Kaiyu Yang, and Jia Deng. Infinite Photorealistic Worlds Using Procedural Generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12630–12641, 2023. 2

[5] Yue Yang, Fan-Yun Sun, Luca Weihs, Eli VanderBilt, Alvaro Herrasti, Winson Han, Jiajun Wu, Nick Haber, Ranjay Krishna, Lingjie Liu, Chris Callison-Burch, Mark Yatskar, Aniruddha Kembhavi, and Christopher Clark. Holodeck: Language Guided Generation of 3D Embodied AI Environments. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16227–16237, 2024. 1

[6] Mengqi Zhou, Jun Hou, Chuanchen Luo, Yuxi Wang, Zhaoxiang Zhang, and Junran Peng. SceneX: Procedural Controllable Large-Scale Scene Generation via Large-Language Models. *arXiv e-prints*, pages arXiv–2403, 2024. 2

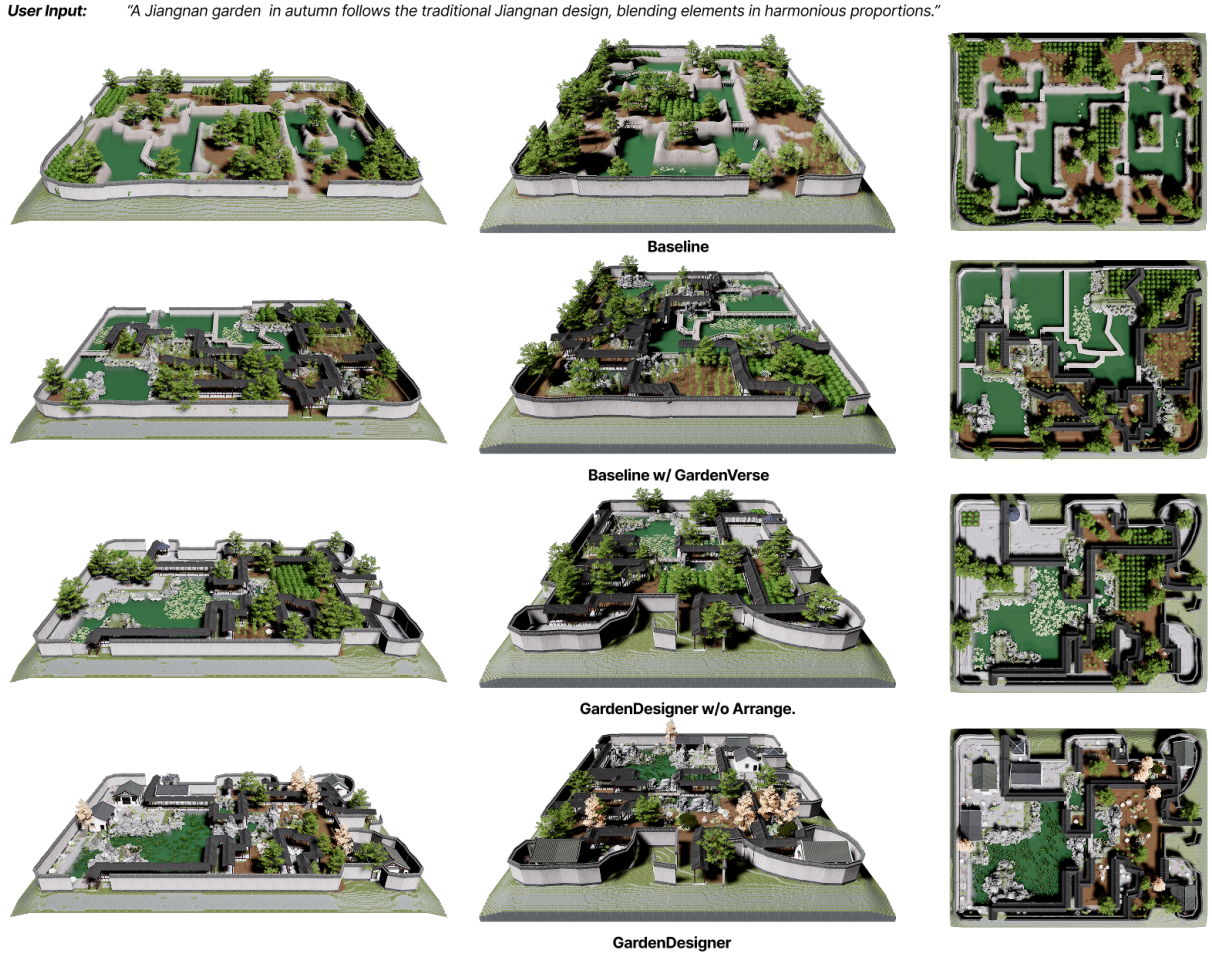


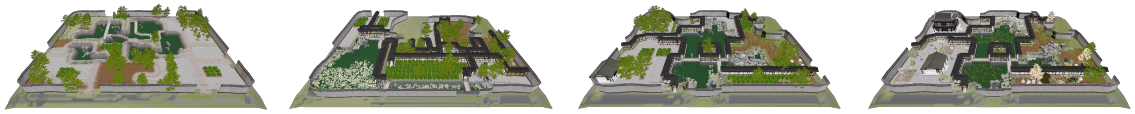
Figure 4. Qualitative results. (a) We conduct the ablation study to compare the different methods for garden construction with same user input, and the view from left to right is front view, right view and top view. (b) We input the same user instruction to evaluate the generation diversity of GardenDesigner. (c) We also input the same user instruction and keep the same terrain to generate different gardens.

Non-Expert Questionnaire

Questions:

(1) ***Overall Quality:*** *which method has the best overall quality?*
(2) ***Text Relevance:*** *Which method has the highest alignment with the text?*
(3) ***Spatial Layout:*** *Which method achieves the most accurate terrain and object layout?*
(4) ***Cultural Atmosphere:*** *Which method best captures the cultural essence of Jiangnan gardens?*

Methods:



Method 1 Method 2 Method 3 Method 4

Selection:

	Normal	Hydric	Floral	Arch-dense	Mazy
Answer:	A/B/C/D				

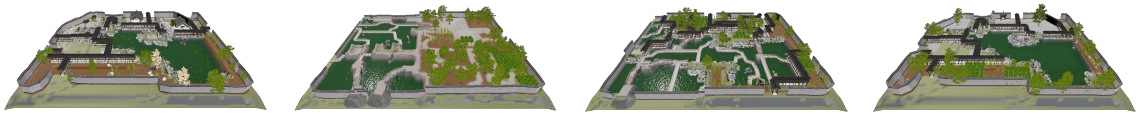
(a)

Expert Questionnaire

Questions:

(1) ***Overall Quality:*** *which method has the best overall quality?*
(2) ***Text Relevance:*** *Which method has the highest alignment with the text?*
(3) ***Spatial Layout 1:*** *Which method achieves the most accurate terrain and object layout?*
(4) ***Spatial Layout 2:*** *Which method results in the most reasonable and natural terrain layout?*
(5) ***Cultural Atmosphere 1:*** *Which method best aligns with the design principles of Jiangnan gardens?*
(6) ***Cultural Atmosphere 2:*** *Which method best aligns with the design principles of Jiangnan gardens?*

Methods:



Method 1 Method 2 Method 3 Method 4

Selection:

	Normal	Hydric	Floral	Arch-dense	Mazy
Answer:	A/B/C/D				

(b)

Figure 5. Questionnaire Survey. We conducted human evaluation with experts and no-experts. (a) In non-expert questionnaire, we provide four questions from overall quality, text relevance, spatial layout, and cultural atmosphere for volunteers to answer. And we provide five types of generated Jiangan gardens from four different methods. (b) We refine the question about sptatial layout and cultural atmosphere.

"""
You are a Jiangnan gardens designer, please assist me to convert the user input into parameters regarding the terrain, which will help the design of a landscape on a two-dimensional discrete rectangular grid (e.g., 20*15 cells). The designer decides to assign each cell a type of terrain from the following five types, each with its own integer code: Unused (0), Aquatic (1), Terrestrial (2), and Artificial (3).
- "Unused" indicates that the cell is excluded from the site, allowing the garden shape to vary.
- "Aquatic" signifies the presence of a water body.
- "Terrestrial" represents flat land with natural elements.
- "Artificial" features flat ground covered by artificial elements.

And the designer now inputs a description of the landscape. You need to extract possible parameters and constraints from it. Specifically, consider the following aspects:
1. Whether a certain type of terrain should or should not be present.
2. The number of regions of a certain type of terrain.
3. The total coverage proportion of a certain type of terrain.
4. The coverage proportion of any single region with a certain type of terrain.
5. The maximum height for the elevated terrain.

Some guidelines:
You need to fully understand the user's intent, especially whether the user is affirming or negating something. There may be some implicit/indirect but strong preferences presented, which should also be converted into specific parameters or constraints.
The user input may not contain information on all the aspects mentioned above. In such cases, you should fill in -1 for the corresponding items when outputting. Unless you are confident that a description can be effectively converted into a parameter or constraint, treat it as if the user did not provide that information.
Your output should be logically consistent, e.g., proportions should not exceed 1 or be less than 0, the lower limit of a range should not be greater than the upper limit, and the number of regions should be greater than 0 if a certain type of terrain is specified. If the user input cannot be parsed according to the above description, or if it is clearly non-compliant, inform the user accordingly.

Your output should be and only be a JSON object, with the following format:
```
{
  "data": [
    [<bool>, <bool>, <bool>, <bool>],
    [[<int>, <int>], [<int>, <int>], [<int>, <int>], [<int>, <int>]],
    [[<float>, <float>], [<float>, <float>], [<float>, <float>], [<float>, <float>]],
    [[<float>, <float>], [<float>, <float>], [<float>, <float>], [<float>, <float>]],
    <float>
  ],
  "feedback": <str>
}
```
Each item in the "data" list corresponds to the above aspects in order. When the item is a list, each element corresponds to a terrain type.
1. The first item is a list of boolean values representing whether the terrain type should exist (0 for no, 1 for yes).
2. The second item is a list of tuples representing lower and upper limits of the region count.
3. The third item is a list of tuples representing lower and upper limits of the total coverage proportion.
4. The fourth item is a list of tuples representing lower and upper limits of the coverage proportion of any single region.
5. The fifth item is a floating-point number representing the maximum height.
The "feedback" field should be "OK" if the input is generally informative and valid; otherwise, it should contain a message indicating the issue.
"""

Figure 6. Prompts for the terrain generation agent.

"""
You are a Jiangnan garden architect who understands what a designer needs. Your task is to convert the user input into parameters regarding the spots and roads, which will help the design of a garden on a two-dimensional discrete rectangular grid (e.g., 20*15cells).
The designer decides to:
- Assign some cells as "key spots".
- Determine some grid corners as the entrances/exits to the landscape.
- Design the main roads that go along the edges (grid borders) and connect all key spots and entrances/exits.
- Design the secondary roads that form a finer road network.

And the designer now inputs a description of the landscape. You need to extract possible parameters and constraints from it. Specifically, consider the following aspects:
1. The number of entrances/exits.
2. The number of key spots.
3. The width of the main roads.
4. The complexity of the roads (the proportion of the total length of the road edges to the total length of the grid edges, generally between 0.3 and 0.4).
You need to fully understand the user's intent. There may be some implicit/indirect but strong preferences presented, which should also be converted into specific parameters or constraints.
The user input may not contain information on all the aspects mentioned above. In such cases, you should fill in -1 for the corresponding items when outputting. Unless you are confident that a description can be effectively converted into a parameter or constraint, treat it as if the user did not provide that information.
Your output should be logically consistent, e.g., proportions should not exceed 1 or be less than 0, the lower limit of a range should not be greater than the upper limit, and the number of regions should be greater than 0 if a certain type of terrain is specified. If the user input cannot be parsed according to the above description, or if it is clearly non-compliant, inform the user accordingly.

Your output should be and only be a JSON object, with the following format:
```
{
  "data": [
    [<int>, <int>],
    [<int>, <int>],
    [<float>, <float>],
    [<float>, <float>]
  ],
  "feedback": <str>
}
```
Each item in the "data" list corresponds to the above aspects in order.
1. The first item is a tuple representing the lower and upper limits of the number of entrances/exits.
2. The second item is a tuple representing the lower and upper limits of the number of key spots.
3. The third item is a tuple representing the lower and upper limits of the width of the main roads.
4. The fourth item is a tuple representing the lower and upper limits of the complexity of the roads.
The "feedback" field should be "OK" if the input is generally informative and valid; otherwise, it should contain a message indicating the issue.
"""

Figure 7. Prompts for the road generation agent.

"""
You are an experienced Jiangnan garden designer, please assist me in selecting architectures, plants, rocks for multiple areas.
I will provide (1) the area list including area size and adjacent information; and area number, (2) object description in the file provided for file search and (3) the user instruction.
You need to select appropriate objects consider these information.
The area list contains multiple tuples, and each tuple has two elements:
(the area type and adjacent information: (1) 0 is the water area; (2) 1 is land adjacent to water; (3) 2 is land adjacent to garden boundary; (4) 3 is land not connected to water or boundary,
the area size)

Here are some guidelines for you:
1. All the area should be included, please don't ingore or return less area. If the area is too small, the area should be defined as empty but don't miss the area in response.
2. Make full use of the area space based on the object area size, make each area occupied, select objects as much as possible.
3. The return area number should be same as input area number, and order should be strictly matched to the given area.
4. Place more attic, pavilion, house and wall, which are more important than plants.
5. Select less thing for water area, the architecture should just be selected for land area.
6. The first object for each area should be the most important object.
7. The object type should strictly come from the file for file search.

Please select objects according to the following content:
1. area information: *{area}*; area number: *{area_num}*
2. object description in file provided for file search
3. user instruction: *{text}*

And just return the json recommendations, present your recommendations in JSON format:
```
{
   "area_number":{
      "object_name": ["object_type", the number of object (int, no more than 5)]
   }
}
```

Response example:
```
{
   "area_0": {
      "House_S_F" : ["House", 1],
      "Acer_buergerianum_A": ["Plant", 2],
      "Stone_combination_B": ["Rock",1]
   },
   "area_1": {
      "Bamboo_rock_combination_A": ["Plant", 2],
      "Bamboo_rock_combination_B": ["Plant", 1],
      "Bamboo_rock_combination_C": ["Plant", 3],
   },
   "area_2": {
      "Taihu_rock_combination_A" : ["Rock", 1],
   },
   "area_3": {
   }
}
```
"""

Figure 8. Prompts for the asset selection agent.

"""
You are an experienced Jiangnan gardens designer. Please help me arrange objects in the garden by assigning constraints to each object.
I will provide (1) the area list including area size and adjacent information, (2) selected objects for each area, (3) object description in the file provided for file search and (4) the user instruction.
You need to select appropriate objects consider these information.

The area list contains multiple tuples, and each tuple has two elements:
(the area type and adjacent information: (1) 0 is the water area; (2) 1 is land adjacent to water; (3) 2 is land adjacent to garden boundary; (4) 3 is land not connected to water or boundary,
the area size)

The selected objects for each area are in dictionary format:
{
  "area_number":{
    "object_name": ["object_type", the number of object (int)]
  },...
}

Here are some guidelines for you:
1. Don't miss any area and objects, if it is empty keep it. All the area and object should be included orderly, each object should have a global constraints.
2. The output area and object should be ordered as input area, if certain area is empty, keep the output of this area is empty dictionay.
3. If there are same object in some area, please keep the object quantity and add "-0" "-1" "-2"... after original name to identify them.
4. Each object should have a global contraints at the first position of all contraints.
5. I will use your guideline to arrange the objects *iteratively*, so please start with an anchor object which doesn't depend on the other objects (with only one global constraint).
6. Place the larger and architecture objects first.
7. The objects in different areas should not have the interactive constraints. The latter objects could only depend on the former objects.
8. The architectures of the *same type* are usually *aligned*.
9. I prefer objects to be placed at the middle, and not be regularly arrangemented.

Here are the constraints and their definitions:
1. global constraint:
  1) edge: object at the edge of the area.
  2) middle: object is centrally placed as a visual or spatial focus.

2. distance constraint:
  1) near, object: near to the other object.
  2) far, object: far away from the other object.

3. position constraint:
  1) around, object: around another object, usually used for plant.
  2) backed by, object: backed by another object, wall or corridor.

4. alignment constraint:
  1) aligned, object: align the center of the object with the center of another object.

5. rotation constraint:
  1) face to, object: face to the center of another object.
"""

Figure 9. First part of prompts for the layout optimization agent.

```
"""
Please select objects according to the following content:
1. area information: *{area}*
2. object dictionary for each area: *{object_dict}*
3. object description in file provided for file search
4. user instruction: *{text}*

For each object, you must have one global constraint and you can select various numbers of constraints and
any combinations of them and the output must be json format:
{
   "area_name"{
      "object_name": [
         ["constraints", "constraints type"]
      ]
      "object_name": [
         ["constraints", "constraints type"],
         ["constraints", "another_object_name", "constraints type"]
      ]
   }
}

For example:
{
   "area_0"{
      "House_S_F-0": [
         ["middle", "global"]
      ],
      "House_S_F-1": [
         ["middle", "global"],
         ["aligned", "House_S_F-0", "alignment"]
      ],
      "Pavilion_F": [
         ["middle", "global"],
         ["face to", "water area", "rotation"]
      ],
      "Asiaticapple_combination_A": [
         ["edge", "global"],
         ["near", "House_S_F-1", "distance"]
      ]
   }
   "area_1"{
      "Peach_combination_A": [
         ["middle", "global"]
      ]
   }
}
"""
```

Figure 10. Second part of prompts for the layout optimization agent.