

References

- [1] Ata Çelen, Guo Han, Konrad Schindler, Luc Van Gool, Iro Armeni, Anton Obukhov, and Xi Wang. I-design: Personalized llm interior designer. *arXiv preprint arXiv:2404.02838*, 2024. [1](#)
- [2] Gabrielle Littlefair, Niladri Shekhar Dutt, and Niloy J. Mitra. Flairgpt: Repurposing llms for interior designs. *Computer Graphics Forum*, 44(2):e70036, 2025. [1](#), [4](#), [5](#)
- [3] Fan-Yun Sun, Weiyu Liu, Siyi Gu, Dylan Lim, Goutam Bhat, Federico Tombari, Manling Li, Nick Haber, and Jiajun Wu. Layoutvlm: Differentiable optimization of 3d layout via vision-language models. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 29469–29478, 2025. [1](#), [4](#), [5](#)
- [4] Hou In Ivan Tam, Hou In Derek Pun, Austin T. Wang, Angel X. Chang, and Manolis Savva. SceneMotifCoder: Example-driven Visual Program Learning for Generating 3D Object Arrangements. 2024. [5](#), [8](#)
- [5] Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. Executable code actions elicit better LLM agents. In *Forty-first International Conference on Machine Learning*, 2024. [1](#)
- [6] Yue Yang, Fan-Yun Sun, Luca Weihs, Eli Vanderbilt, Alvaro Herrasti, Winson Han, Jiajun Wu, Nick Haber, Ranjay Krishna, Lingjie Liu, et al. Holodeck: Language guided generation of 3d embodied ai environments. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2024)*, pages 20–25. IEEE/CVF, 2024. [1](#)

INTERIORAGENT: LLM Agent for Interior Design-Aware 3D Layout Generation

Supplementary Material

1. Acknowledgments

This work was supported in part by NSF grant IIS 2402583, a Qualcomm Innovation Fellowship, ONR grant N00014-23-1-2526, gifts from Adobe, Rembrand, and the Ronald L. Graham Chair.

2. Reproducibility

All code, data, prompts and experimental details will be publicly released.

Please see the attached `constraints.py` file in the folder for reference.

3. Full prompts for Visualizations

MIT Scenes

Bakery := *“The bakery’s layout includes refrigerated display cases along the left and back wall. In the center, a long wooden table displays freshly baked loaves and pastries. A few tables and chairs are placed in the front for customers.”*

Kitchen := *“The kitchen features a large island in the center of the back wall with a marble countertop. Along one wall, there is a row of cabinets and there is a coffee table with two armchairs in the corner.”*

Classroom := *“The classroom features rows of individual desks with attached chairs arranged in neat columns. A large whiteboard spans the front wall, with a teacher’s desk placed in one corner.”*

Gym := *“The center of the gym has several stationary bikes arranged in a grid pattern with a TV mounted on their front wall. Behind them, a row of treadmills are placed along the wall.”*

Buffet := *“A buffet scene with a row of tables placed along the left, each having a variety of food items placed on top. In the center, four round tables are placed for seating”*

Flower Shop := *“The flower shop features a central display table with flower bouquets on top, and floor lamps illuminating the arrangements. Along the walls, there are additional tables with potted plants”*

4. MIT Scenes Dataset

Comparison to LayoutVLM[3] and Holodeck[6] Please see Figs. 1, 2 for more illustrations of scenes generated via INTERIORAGENT, LayoutVLM[3] and Holodeck[6]. We

note the ability of INTERIORAGENT to more accurately follow descriptive prompts while maintaining better aesthetic quality. We observe that INTERIORAGENT is able to utilize tools in its toolkit to generate a diverse set of realistic scenes that are not only more visually pleasing, but also respect human-centric factors for better interior design. Further, we note that INTERIORAGENT is able to effectively use both 3D-FRONT as well as other datasets such as Objaverse to meet its asset requirements. This is a key advantage over previous methods which do not have a *tool* based approach and therefore, are not easily extensible beyond a fixed dataset.

Comparison to FlairGPT[2] We compare INTERIORAGENT with FlairGPT [2], another state-of-the-art method for indoor scene generation also inspired by interior design principles. FlairGPT consists of three stages: a *Language Phase*, where an LLM is prompted via chain-of-thought to generate an initial layout by identifying zones, adding anchor and secondary assets, and specifying constraints; a *Translation Phase*, which converts these natural-language layouts and constraints into programs; and an *Optimization Phase*, which executes the program to produce the final scene.

Despite these apparent similarities, INTERIORAGENT differs in several key ways. First, it adopts a direct program synthesis approach to specify layouts and constraints, avoiding the limitations of natural language representations used in FlairGPT, Holodeck [6], I-Design [1], and LayoutGPT [3]. Prior work shows program synthesis provides stronger planning capabilities through programming structures and external tools [5]. This is reflected in reliability: FlairGPT fails to translate to an executable program roughly 25% of the time (tested on 20 MIT Scenes prompts, issue also noted in official code release), whereas INTERIORAGENT, supported by code debuggers, achieves a near 99% execution rate. Runtime also differs substantially: FlairGPT requires 20 minutes per scene versus 3 minutes for INTERIORAGENT.

Second, while both methods reference functional zones, INTERIORAGENT formalizes them via dedicated group *templates* that support diverse zone types with isolated object registration, placement, and optimization. This hierarchical design more faithfully captures functional groupings than FlairGPT’s chain-of-thought prompting.

Third, IDSDDL equips INTERIORAGENT with a richer suite of placement and optimization tools, including constraints unavailable in FlairGPT such as Clearance, Visibility, and VLM-based constraints (ObjectProportions, RoomProportion, Conversation, Balance). These contribute to more coherent and functional layouts (Figures 3, 4).

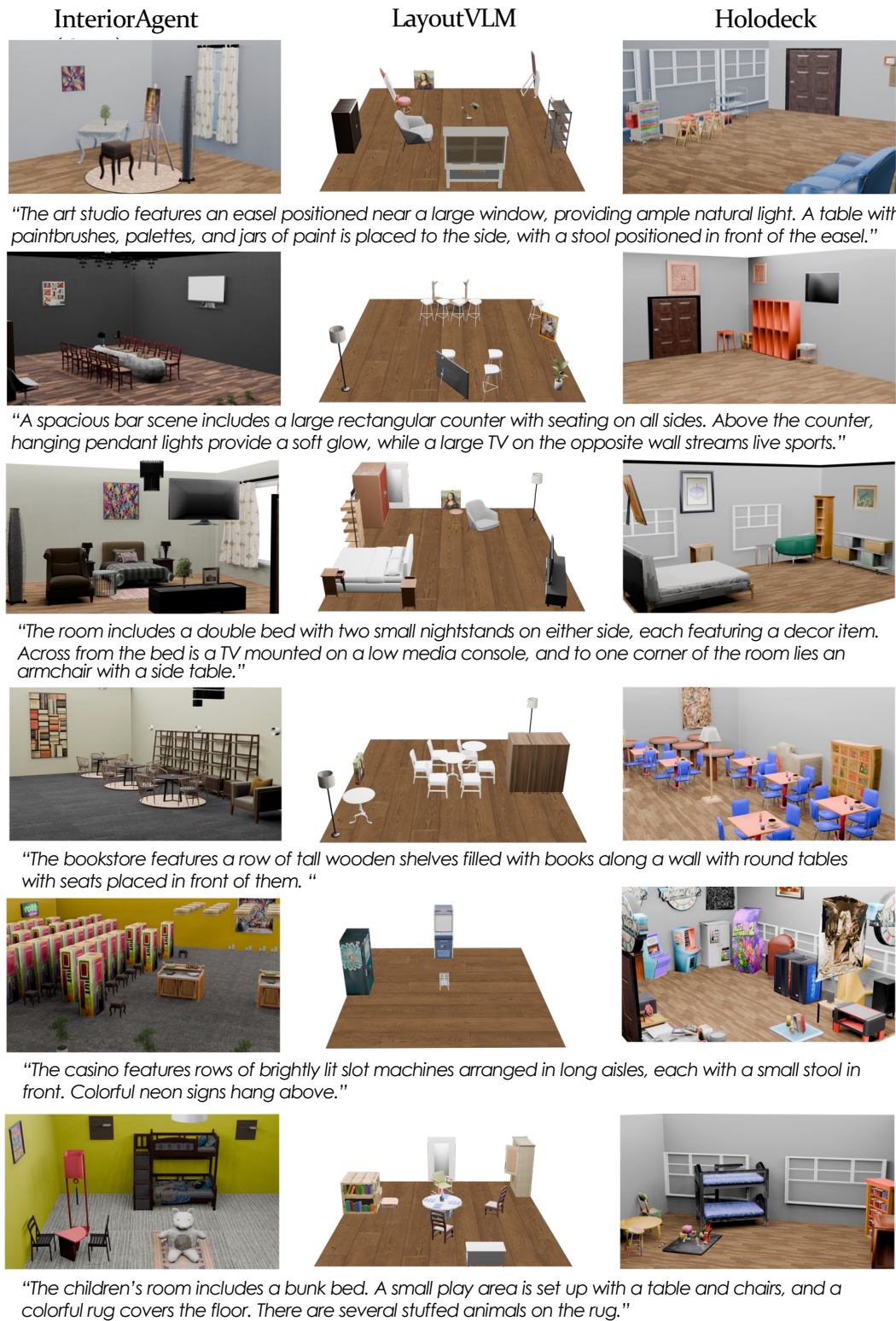
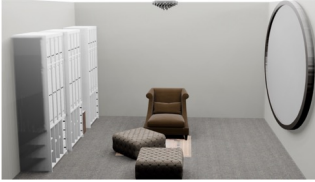


Figure 1. Example renders of scenes generated with prompts from six MIT Scenes categories.

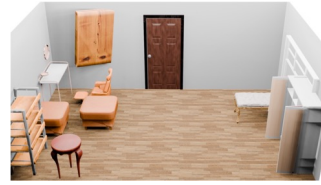
InteriorAgent



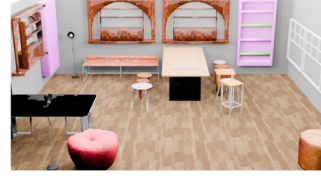
LayoutVLM



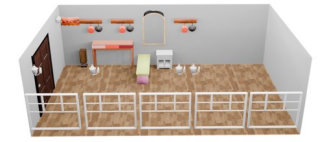
Holodeck



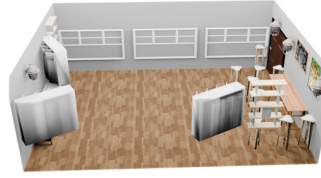
"The walk-in closet has an entire wall covered in mirror panels. The opposite wall features wardrobes. In the center, there are two ottomans and an armchair. It also has overhead lighting to illuminate the space."



"The computer room features a row of computer desks with desktop monitors along one wall. A large whiteboard is mounted on the opposite wall, and a large table is placed in the center with stools around it."



"The corridor features a door each along the left and right walls. A runner rug lines the center of the floor, several paintings are placed the front which are visible to couches placed in the back."



"The deli features several refrigerated units along the back wall, with a row of long glass display counter filled with fresh sandwiches and salads, positioned at some distance in front of them. A row of bar stools lines a counter along the wall with a huge window for casual seating."



"The dental office features a reception desk, a waiting area with a few chairs in the middle. Along the back wall, there are three dental chairs with a tray of instruments placed next to each. There are plants, floor lamps placed near the reception desk."



"The museum hall features a central exhibit showcasing a large dragon on a raised platform. Spotlights from the ceiling illuminate the artifact, and benches are positioned along the walls for visitors."

Figure 2. Example renders of scenes generated with prompts from six MIT Scenes categories.

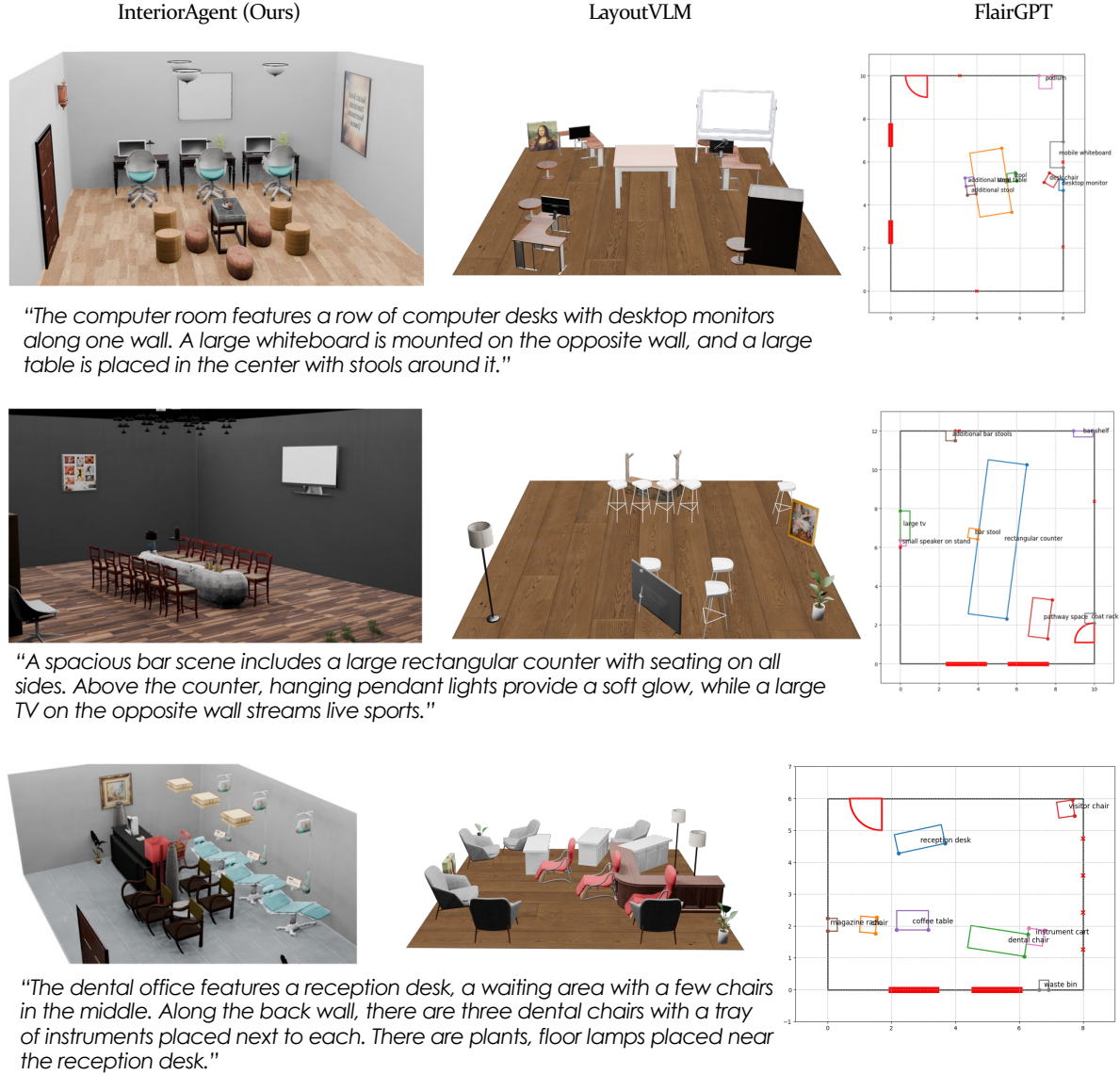


Figure 3. **Comparison to FlairGPT.** MIT Scenes renders for INTERIORAGENT, LayoutVLM [3], and 2D Layouts from FlairGPT [2]. *Computer room:* FlairGPT fails to place desks in a row with monitors; LayoutVLM produces multiple desks but not aligned. *Bar:* FlairGPT inserts a central table but only a single stool, missing seating on all sides. *Dental office:* FlairGPT adds just one dental chair instead of three, and omits plants and lamps near the reception desk. In contrast, INTERIORAGENT generates scenes that are prompt-faithful, aesthetically pleasing, and functionally coherent.

Finally, FlairGPT’s placement and optimization capabilities are fixed, while INTERIORAGENT offers an extensible framework, allowing designers to easily add new templates for asset retrieval, placement, and optimization.

For comparison, we use the officially released FlairGPT code. However, it only produces 2D layouts without object orientations, preventing generation of 3D scenes and direct rendering comparisons. We therefore conduct a qualitative comparison on six MIT Scenes prompts (Figures 3, 4). Across all cases, INTERIORAGENT consistently outperforms

FlairGPT [2] in object retrieval (FlairGPT often omits key objects specified in the prompt), placement (it frequently fails to capture explicit inter-object relations), and optimization (its scenes are less functional).

5. Placement Groups

See Figure 8

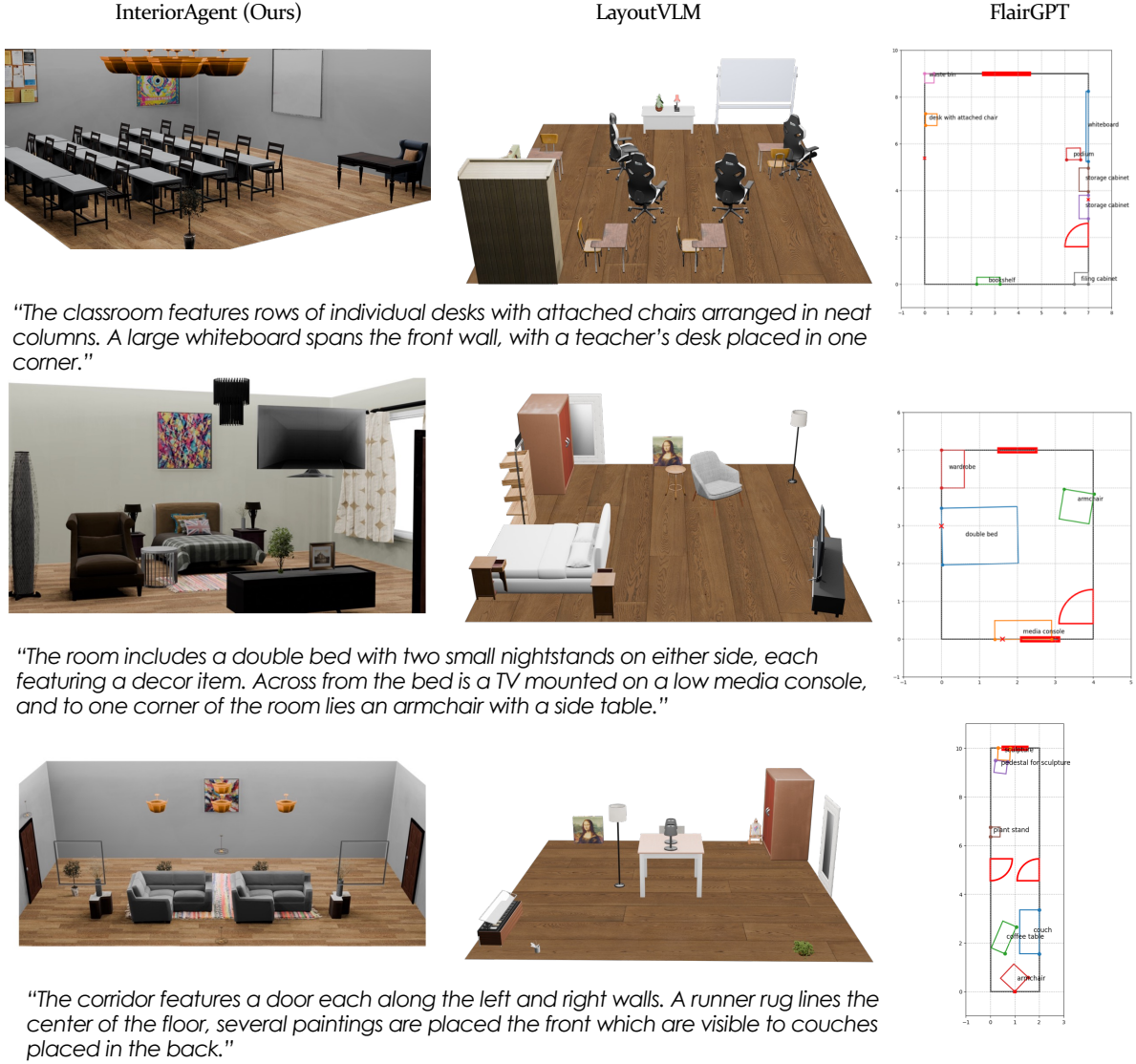


Figure 4. **Comparison to FlairGPT.** MIT Scenes renders for INTERIORAGENT, LayoutVLM [3], and 2D layouts from FlairGPT [2]. *Classroom:* FlairGPT places only a single desk–chair despite the prompt asking for a row of desks with attached chairs, and omits the teacher’s desk–chair. *Bedroom:* Nightstands and a side table for the armchair are missing. *Corridor:* Entry–exit doors are misplaced, breaking the layout’s intent of having seating in between. In contrast, INTERIORAGENT produces scenes that are prompt-faithful, aesthetically coherent, and functionally sound.

6. Tool Usage

INTERIORAGENT is highly extensible and as such its capabilities can be easily enhanced by providing additional tools. Here we show one example each for object retrieval, placement and optimization.

Object retrieval SceneMotifCoder [4] generates 3D object arrangements through visual program learning. A key feature of SceneMotifCoder is its ability to create *stacks* or *grid-like* arrangements of objects. Since objects like “a

stack of 7 books” aren’t readily available in popular datasets, we enable this functionality in INTERIORAGENT by adding SceneMotifCoder as one of our retrieval tools. To achieve this we write a template as shown in Figure 9, that gives relevant information to INTERIORAGENT about using this tool. Additionally, we follow the official documentation to correctly use SceneMotifCoder and use that to write the `_call_()` function that implements the functionality. Figure 10 shows that with such a minimal input, INTERIORAGENT is able to easily leverage the SceneMotifCoder tool to generate *stacked* objects for its larger scene synthesis

```

1 class InteriorAgentAssetRetriever:
2     def __init__(self, scene):
3         self.name = "Some name like Front3DAssetRetriever"
4         self.description = """A brief description of its use case,
5             like 3DFront is great for indoor furniture!"""
6         self.example = """A few examples of this retriever in use
7             1. A wooden cabinet 2. A modern sofa"""
8         def build(self): ## (Optional) some retrievers may require caching
9             ## or pre-computation such as loading models to GPU
10            ## Example for building embeddings for 3DFront/RS2D retriever.
11            ## Other datasets may use different embedding methods.
12            all_descriptions = [self.VLM(f"""Give a brief description of the asset,
13                including its type, material, and any notable features. Can use this metadata: {metadata}""",
14                images=renders) for metadata, renders in self.all_models]
15            scales = [self.VLM(f"""What is the width of this asset? {desc}""", images=renders)
16                for desc, renders in zip(all_descriptions, self.all_models.renders())]
17            self.embd = [OpenAIEmbeddings().embed_query(desc), scale
18                for desc, scale in zip(all_descriptions, scales)]
19
20        def __call__(self, description: str) -> tuple(str, float):
21            ## Logic to retrieve an asset based on the description
22            embd = OpenAIEmbeddings().embed_query(description)
23            similarity = self.embd.similarity_search(embd, k=1)
24            path, scale = similarity[0].path, similarity[0].scale
25            ## Can also use text-to-3D models to generate 3D assets. Result section show more
26            ## sophisticated retrievers including StableDiffusion for generating paintings and
27            ## 3D generation repository SceneNetCoder for stacked assets.
28            return path, scale # asset path and width dimensions

```

Figure 5. Template for defining an asset retriever in IDSDL, with an example 3DFront retriever implementation.

```

1 class InteriorAgentGroup:
2     def __init__(self, scene, *args, **kwargs):
3         self.name = "Some name like RelativeGroup"
4         ## Some group specific parameters
5         self.SIDE_GAP = 0.1 # Gap between objects placed on the sides
6
7         @placemethod ## Decorator to define a placement method
8         def place_on_left(self, obj, *args, **kwargs):
9             # A brief description of its use case and parameters
10            self.description = f"""
11            Place an object on the left side of the anchor object
12            Inputs: - obj: The object to be placed on the left side of the anchor object.
13            Outputs: - None, the object is placed in the scene.
14            """
15            # A few examples of this placement methods in use
16            self.example = f"""
17            with scene.RelativeGroup() as group:
18                group.set_anchor(sofa) ## Set sofa (initialized earlier) as the anchor object
19            end_table = scene.AddAssets("A modern end table")
20            group.place_on_left(end_table) ## places the end table on the left side of the sofa
21            """
22            ## Logic to compute the object poses x, y, z, theta
23            ## For relative placement, compute cardinal directions, center point and dimensions of anchor object
24            dira, center, dims = self.get_anchor_data()
25            width, height, depth = dims
26            front_dir, back_dir, left_dir, right_dir = dirs
27
28            left = center + left_dir * (width / 2 + obj.get_width() / 2 + self.SIDE_GAP)
29            x, y, z, theta = left[0], obj.get_height() / 2, left[2], 0 # Place on floor
30            obj.set_location(x, y, z)
31            obj.set_rotation(theta)
32            self.add_child(obj) ## Adds the object to the group
33
34            ## Optionally, control the placement and optimisation flow of the group.
35            def compile(self):
36                for op in self.operations: ## All placements are stored in self.operations
37                    if self.operations[op] is not None:
38                        self.operations[op].execute()
39            self.grad_optimize() ## Optimize gradient-based constraints
40            self.vlm_optimize() ## Compute feedback for VLM-based constraints

```

Figure 6. Template for defining groups in IDSDL, with an example place_on_left () method in RelativeGroup ().

objective. See Figure 5 for object registration template.

Object placement The teaser figure shows a large scale scene consisting of a cherry blossom tree forest. Usually, such scenes are extremely challenging for an LLM to create given the limited 3D understanding capacity of LLMs. However, when the task is subdivided at a word and later at alphabet level, we note that LLMs can be reliably prompted to achieve desired results. Therefore, we use a prompt engineered LLM (GPT-4o) to serve as a tool for creating their own ascii art. Figure 11 shows the entire code used for this implementation. See Figure 6 for object placement template.

Object arrangement Another interesting use case can be enabled by allowing rendering based losses to work in tandem with gradient and VLM based losses which is made possible because IDSDL seamlessly unifies all the various constraint and types. We show an application where wall art needs to be arrangement by comparing them to a target

```

1 class InteriorAgentConstraint:
2     def __init__(self, scene, group, *args, **kwargs):
3         self.name = "Some name like OverlapConstraint"
4         self.description = f""" ## A brief description of its use case and parameters
5         Ensures that no two objects overlap with each other
6         No inputs or outputs, this is a constraint that is applied to the group.
7         """
8         self.example = f""" ## A few examples of this constraint in use
9         with scene.<Group() as group:
10            ...
11            group.OverlapConstraint() ## Applies the overlap constraint to the group
12            """
13        def compute_gradients(self):
14            ## Compute pseudo gradients
15            objects = self.group.children
16            for i in range(len(objects)):
17                for j in range(i + 1, len(objects)):
18                    obj1, obj2 = objects[i], objects[j]
19                    status, degree = obj1.is_overlap(obj2)
20                    if status:
21                        v1, v2 = obj1.get_location(), obj2.get_location()
22                        grad1, grad2 = (v1 - v2) * degree, (v2 - v1) * degree
23                        obj1.grad += grad1 ## Add gradient to the total gradient of each object
24                        obj2.grad += grad2
25
1 class InteriorAgentVLMConstraint:
2     def __init__(self, scene, group, *args, **kwargs):
3         self.name = "ObjectProportionsConstraint"
4         self.description, self.example = ... ## Set description, example
5         self.type = "VLM" # VLM-based constraints
6         self.system_desc = f"""
7         You are given an image showing front, right, back, and left views of a scene with several objects.
8         Check whether the objects' proportions make sense; if any object is too big or too small,
9         respond only with a rescale instruction in the form rescale {object} by {factor}.
10        where the factor is a float between 0.1 and 0.9 (e.g., rescale coffee table by 0.5).
11        If everything looks correct, respond with no rescale. """
12
13        def compute_gradients(self):
14            ## Compute VLM-based feedback
15            render = self.group.render()
16            descriptions = self.group.get_descriptions()
17            prompt = f"""
18            The scene has the following objects: {",".join(descriptions)}
19            Now reason about the relative proportions of the objects in the scene and to ensure that
20            they make sense in the context of the scene.
21            """
22            feedback = self.VLM(prompt, image_paths=[render])
23            self.scene.vlm_feedback += feedback

```

Figure 7. Template for defining constraints in IDSDL, with an example gradient based OverlapConstraint () (top) and VLM based ObjectProportionsConstraint () (bottom)

wall at arrangement specified by the user. Figure 12 shows the constraint template being filled with relevant context for LLM to use as well as a logic for computing pseudo gradients to allow movement of the assets such that they match the target as best as possible. Figure 13 shows the scene program generated and the obtained results. See Figure 7 for object arrangement optimization template.

7. Scene Editing Application

Figure 14 shows the visualization of scene editing application discussed in the main paper. INTERIORAGENT ’s chat-like interface allows users to generate and iteratively edit scenes, accommodating complex requirements beyond a single prompt.

8. Failure Cases

While INTERIORAGENT significantly outperforms prior work with its program synthesis approach, challenges remain to be addressed in future works. Most importantly, we note that INTERIORAGENT may suffer when the underlying tool malfunctions or doesn’t show expected behavior. Figure 15 shows a few such examples. In the leftmost example, the retrieval tool incorrectly retrieves a bundle of t-shirts when asked to retrieve a ‘packing station’ to be placed in the warehouse. In the middle two examples, while ergonomics constraints account for visibility and clearance, it is possible that the LLM simply misses out on applying relevant constraints when it should, resulting in scenes not being op-

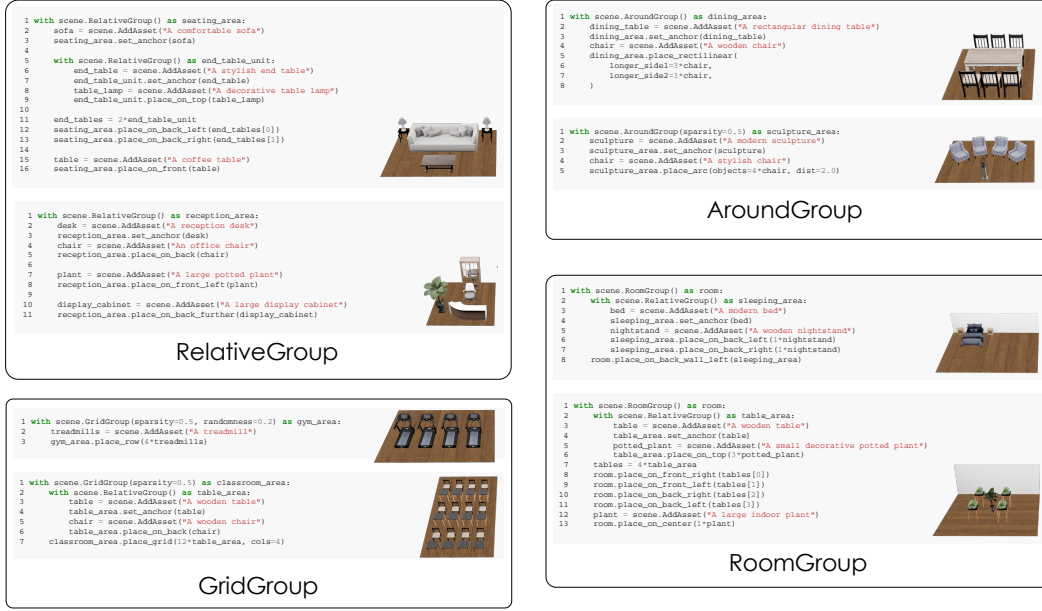


Figure 8. Placement programs involving various group types demoed in method section. Note as few as 10 lines of code are sufficient to represent a wide variety of scenes including living room, reception area (RelativeGroup), dining area, a art museum viewing area (AroundGroup), gym, classroom (GridGroup), bedroom and floweriest shop. (RoomGroup)

timized for those constraints. In the rightmost example, a large number of objects coupled with their constraints pose a significant optimization challenge, which can lead to issues such as out of bound problems, as is the case for the example of a densely packed warehouse. We believe that these problems can be mitigated in the future by using more advanced tools and by fine-tuning the LLM for improved tool-awareness. Additionally, INTERIORAGENT’s approach to use task-specific tools for scene synthesis raises important considerations on optimal and robust utilization of multiple tools.

9. Ablation

9.1. Debugging Scene Programs

The Debugger performs a key role in program synthesis which is to ensure that programs synthesized by PROGRAMSYNTHESIZER are executable and free from syntax issues. In order to achieve this, it leverages a combination of CodeRefine and TraceRefine in an iterative loop. Upon generating 100 scene programs using the PROGRAMSYNTHESIZER, we observe that CodeRefine was used 135 times while TraceRefine was called 35 times. Note that CodeRefine is invoked every time post PROGRAMSYNTHESIZER to remove potential errors.

9.2. In-Context Examples

A few in context examples are visualized in Fig 16.


```

1 class SceneMotifCoderObject(SceneProgAssetRetrieverBase):
2     def __init__(self):
3         super().__init__()
4         self.name = "SceneMotifCoderObject"
5         self.description = f"""
6 This tool returns 'stacked' objects based on an input description which can be used like any other objects in the scene program
7 """
8         self.examples = """
9 Following are a few examples of this tool in action:
10 Example 1:
11 scene.SceneMotifCoderObject('A table with a chair in front of it')
12 ## Adds a new object to the scene where a chair is placed in front of a table
13 Example 2:
14 scene.SceneMotifCoderObject('A stack of 5 cups')
15 ## Adds a new object to the scene where 5 cups are stacked on top of each other
16 Example 3:
17 scene.SceneMotifCoderObject('A grid of 5x5 chairs')
18 ## Adds a new object to the scene where 25 chairs are arranged in a 5x5 grid
19 """
20
21     def __call__(self, query):
22         code = f"""
23 #!/bin/bash
24 # Path to the Python executable
25 PYTHON_EXECUTABLE="/opt/miniconda3/envs/smc/bin/python"
26 # Path to the inference script
27 SCRIPT_PATH="/<path to smc>/smc/inference.py"
28
29 # Arguments for the script
30 DESC="{query}"
31 OUT_DIR="/<path to sceneprog>/sceneprog/tmp/"
32
33 cd /<path to smc>/smc
34 # Execute the Python script with the arguments
35 $PYTHON_EXECUTABLE $SCRIPT_PATH --desc "$DESC" --out_dir "$OUT_DIR"
36
37 # Wait for the program to complete and check exit status
38 if [ $? -eq 0 ]; then
39     echo "Inference completed successfully."
40 else
41     echo "Inference failed with exit code $?."
42     exit 1
43 fi
44 """
45         with open("tmp/smc_run.sh", "w") as f:
46             f.write(code)
47         import os
48         os.system(f"bash tmp/smc_run.sh")
49         import trimesh
50         mesh = trimesh.load("tmp/stacked.glb", process=False, force='mesh')
51         scale = mesh.bounds[1,0] - mesh.bounds[0,0]
52         return "tmp/stacked.glb", scale

```

Figure 9. Adding SceneMotifCoder[4] to INTERIORAGENT amounts to filling out the tool template which requires a tool name, description of its role, its input/output as well as a few examples of its demonstration. The core logic of tool is implemented under the `__call__()` method which follows the official documentation on correctly using the tool.

```

1 with scene.RelativeGroup() as desk_area:
2     desk = scene.AddAsset("A large wooden desk with a wooden finish")
3     desk_area.set_anchor(desk)
4     chair = scene.AddAsset("A comfortable ergonomic chair")
5     desk_area.place_on_back(chair)
6
7     table_lamp = scene.AddAsset("A stylish desk lamp with a modern glow")
8     stacked_books1 = scene.AddAsset("A stack of 3 books of varying sizes and colors")
9     stacked_books2 = scene.AddAsset("A stack of 3 books of varying sizes and colors")
10    stacked_books3 = scene.AddAsset("A stack of 4 books of varying sizes and colors")
11    indoor_plant = scene.AddAsset("A small indoor plant")
12
13    desk_area.place_on_top([indoor_plant, stacked_books1, stacked_books2, stacked_books3, table_lamp])
14    desk_area.place_rug("A cozy area rug in neutral color", size=0.3)
15    desk_area.add_lighting("A modern chandelier", 0.0)
16
17 with scene.RoomGroup() as room:
18    room.place_walls("floor_texture=wooden", ceiling_texture="plain", wall_texture="calming blue")
19    room.place_window_standard("left_wall", position="middle", curtain=True)
20    room.place_door("back_wall", position="right")
21
22    painting = scene.AddAsset("A motivational poster")
23    room.place_on_wall_front_center(painting)
24
25    shelf = scene.AddAsset("A bookshelf filled with books and personal items")
26    room.place_on_right_wall_center(shelf)
27    room.place_on_center(desk_area)

```

Scene Program using SceneMotifCoder



Generated 3D Scene

Figure 10. INTERIORAGENT can effectively use novel tools. Here, we show INTERIORAGENT writings programs using the SceneMotifCoder tool for the caption “Create a study room scene with a desk with 3 heaps of stacked books”.

```

1 class AlphabetGenerator:
2     def __init__(self):
3         self.llm = LLM(
4             name="ASCIITartGroup",
5             system_desc="You are a large language model based assistant, expert at generating ASCII art representations for alphabets and numbers. Return only python code in Markdown format, e.g.:
6             python
7             ...
8             """
9         )
10    def sanitize(self, text):
11        H = len(text)
12        W = len(text[0])
13        pos = []
14        # Loop through each row of the ASCII representation
15        for y, row in enumerate(text):
16            # Loop through each character of the row
17            for x, char in enumerate(row):
18                # If the character is 'Q', add the coordinates to the list
19                if char == "Q":
20                    pos.append((x, y))
21        return np.array(pos), len(text[0])-1
22    def sanitize_output(self, text: str):
23        _, after = text.split("```python")
24        return after.split("\n")[0]
25    def run(self, query):
26        prompt = """
27        User Input: Generate ASCII art for 'Q'
28        Your Response:
29        python [
30            " " " " " "
31            " " " " " "
32            " " " " " "
33            " " " " " "
34            " " " " " "
35            " " " " " "
36            " " " " " "
37            " " " " " "
38            " " " " " "
39            " " " " " "
40            " " " " " "
41            " " " " " "
42            " " " " " "
43        ]"""
44        User Input: Generate ASCII art for 'S'
45        Your Response:
46        python [
47            " " " " " "
48            " " " " " "
49            " " " " " "
50            " " " " " "
51            " " " " " "
52            " " " " " "
53            " " " " " "
54            " " " " " "
55            " " " " " "
56            " " " " " "
57            " " " " " "
58            " " " " " "
59            " " " " " "
60            " " " " " "
61            " " " " " "
62            " " " " " "
63            " " " " " "
64            " " " " " "
65            " " " " " "
66            " " " " " "
67            " " " " " "
68            " " " " " "
69            " " " " " "
70            " " " " " "
71            " " " " " "
72            " " " " " "
73            " " " " " "
74            " " " " " "
75            " " " " " "
76            " " " " " "
77            " " " " " "
78            " " " " " "
79            " " " " " "
80            " " " " " "
81            " " " " " "
82            " " " " " "
83            " " " " " "
84            " " " " " "
85            " " " " " "
86            " " " " " "
87            " " " " " "
88            " " " " " "
89            " " " " " "
90            " " " " " "
91            " " " " " "
92            " " " " " "
93            " " " " " "
94            " " " " " "
95            " " " " " "
96            " " " " " "
97            " " " " " "
98            " " " " " "
99            " " " " " "
100           " " " " " "
101           " " " " " "
102           " " " " " "
103           " " " " " "
104           " " " " " "
105           " " " " " "
106           " " " " " "
107           " " " " " "
108           " " " " " "
109           " " " " " "
110           " " " " " "
111           " " " " " "
112           " " " " " "
113           " " " " " "
114           " " " " " "
115           " " " " " "
116           " " " " " "
117           " " " " " "
118           " " " " " "
119           " " " " " "
120           " " " " " "
121           " " " " " "
122           " " " " " "
123           " " " " " "
124           " " " " " "
125           " " " " " "
126           " " " " " "
127           " " " " " "
128           " " " " " "
129           " " " " " "
130           " " " " " "
131           " " " " " "
132           " " " " " "
133           " " " " " "
134           " " " " " "
135           " " " " " "
136           " " " " " "
137           " " " " " "
138           " " " " " "
139           " " " " " "
140           " " " " " "
141           " " " " " "
142           " " " " " "
143           " " " " " "
144           " " " " " "
145           " " " " " "
146           " " " " " "
147           " " " " " "
148           " " " " " "
149           " " " " " "
150           " " " " " "
151           " " " " " "
152           " " " " " "
153           " " " " " "
154           " " " " " "
155           " " " " " "
156           " " " " " "
157           " " " " " "
158           " " " " " "
159           " " " " " "
160           " " " " " "
161           " " " " " "
162           " " " " " "
163           " " " " " "
164           " " " " " "
165           " " " " " "
166           " " " " " "
167           " " " " " "
168           " " " " " "
169           " " " " " "
170           " " " " " "
171           " " " " " "
172           " " " " " "
173           " " " " " "
174           " " " " " "
175           " " " " " "
176           " " " " " "
177           " " " " " "
178           " " " " " "
179           " " " " " "
180           " " " " " "
181           " " " " " "
182           " " " " " "
183           " " " " " "
184           " " " " " "
185           " " " " " "
186           " " " " " "
187           " " " " " "
188           " " " " " "
189           " " " " " "
190           " " " " " "
191           " " " " " "
192           " " " " " "
193           " " " " " "
194           " " " " " "
195           " " " " " "
196           " " " " " "
197           " " " " " "
198           " " " " " "
199           " " " " " "
200           " " " " " "
201           " " " " " "
202           " " " " " "
203           " " " " " "
204           " " " " " "
205           " " " " " "
206           " " " " " "
207           " " " " " "
208           " " " " " "
209           " " " " " "
210           " " " " " "
211           " " " " " "
212           " " " " " "
213           " " " " " "
214           " " " " " "
215           " " " " " "
216           " " " " " "
217           " " " " " "
218           " " " " " "
219           " " " " " "
220           " " " " " "
221           " " " " " "
222           " " " " " "
223           " " " " " "
224           " " " " " "
225           " " " " " "
226           " " " " " "
227           " " " " " "
228           " " " " " "
229           " " " " " "
230           " " " " " "
231           " " " " " "
232           " " " " " "
233           " " " " " "
234           " " " " " "
235           " " " " " "
236           " " " " " "
237           " " " " " "
238           " " " " " "
239           " " " " " "
240           " " " " " "
241           " " " " " "
242           " " " " " "
243           " " " " " "
244           " " " " " "
245           " " " " " "
246           " " " " " "
247           " " " " " "
248           " " " " " "
249           " " " " " "
250           " " " " " "
251           " " " " " "
252           " " " " " "
253           " " " " " "
254           " " " " " "
255           " " " " " "
256           " " " " " "
257           " " " " " "
258           " " " " " "
259           " " " " " "
260           " " " " " "
261           " " " " " "
262           " " " " " "
263           " " " " " "
264           " " " " " "
265           " " " " " "
266           " " " " " "
267           " " " " " "
268           " " " " " "
269           " " " " " "
270           " " " " " "
271           " " " " " "
272           " " " " " "
273           " " " " " "
274           " " " " " "
275           " " " " " "
276           " " " " " "
277           " " " " " "
278           " " " " " "
279           " " " " " "
280           " " " " " "
281           " " " " " "
282           " " " " " "
283           " " " " " "
284           " " " " " "
285           " " " " " "
286           " " " " " "
287           " " " " " "
288           " " " " " "
289           " " " " " "
290           " " " " " "
291           " " " " " "
292           " " " " " "
293           " " " " " "
294           " " " " " "
295           " " " " " "
296           " " " " " "
297           " " " " " "
298           " " " " " "
299           " " " " " "
300           " " " " " "
301           " " " " " "
302           " " " " " "
303           " " " " " "
304           " " " " " "
305           " " " " " "
306           " " " " " "
307           " " " " " "
308           " " " " " "
309           " " " " " "
310           " " " " " "
311           " " " " " "
312           " " " " " "
313           " " " " " "
314           " " " " " "
315           " " " " " "
316           " " " " " "
317           " " " " " "
318           " " " " " "
319           " " " " " "
320           " " " " " "
321           " " " " " "
322           " " " " " "
323           " " " " " "
324           " " " " " "
325           " " " " " "
326           " " " " " "
327           " " " " " "
328           " " " " " "
329           " " " " " "
330           " " " " " "
331           " " " " " "
332           " " " " " "
333           " " " " " "
334           " " " " " "
335           " " " " " "
336           " " " " " "
337           " " " " " "
338           " " " " " "
339           " " " " " "
340           " " " " " "
341           " " " " " "
342           " " " " " "
343           " " " " " "
344           " " " " " "
345           " " " " " "
346           " " " " " "
347           " " " " " "
348           " " " " " "
349           " " " " " "
350           " " " " " "
351           " " " " " "
352           " " " " " "
353           " " " " " "
354           " " " " " "
355           " " " " " "
356           " " " " " "
357           " " " " " "
358           " " " " " "
359           " " " " " "
360           " " " " " "
361           " " " " " "
362           " " " " " "
363           " " " " " "
364           " " " " " "
365           " " " " " "
366           " " " " " "
367           " " " " " "
368           " " " " " "
369           " " " " " "
370           " " " " " "
371           " " " " " "
372           " " " " " "
373           " " " " " "
374           " " " " " "
375           " " " " " "
376           " " " " " "
377           " " " " " "
378           " " " " " "
379           " " " " " "
380           " " " " " "
381           " " " " " "
382           " " " " " "
383           " " " " " "
384           " " " " " "
385           " " " " " "
386           " " " " " "
387           " " " " " "
388           " " " " " "
389           " " " " " "
390           " " " " " "
391           " " " " " "
392           " " " " " "
393           " " " " " "
394           " " " " " "
395           " " " " " "
396           " " " " " "
397           " " " " " "
398           " " " " " "
399           " " " " " "
400           " " " " " "
401           " " " " " "
402           " " " " " "
403           " " " " " "
404           " " " " " "
405           " " " " " "
406           " " " " " "
407           " " " " " "
408           " " " " " "
409           " " " " " "
410           " " " " " "
411           " " " " " "
412           " " " " " "
413           " " " " " "
414           " " " " " "
415           " " " " " "
416           " " " " " "
417           " " " " " "
418           " " " " " "
419           " " " " " "
420           " " " " " "
421           " " " " " "
422           " " " " " "
423           " " " " " "
424           " " " " " "
425           " " " " " "
426           " " " " " "
427           " " " " " "
428           " " " " " "
429           " " " " " "
430           " " " " " "
431           " " " " " "
432           " " " " " "
433           " " " " " "
434           " " " " " "
435           " " " " " "
436           " " " " " "
437           " " " " " "
438           " " " " " "
439           " " " " " "
440           " " " " " "
441           " " " " " "
442           " " " " " "
443           " " " " " "
444           " " " " " "
445           " " " " " "
446           " " " " " "
447           " " " " " "
448           " " " " " "
449           " " " " " "
450           " " " " " "
451           " " " " " "
452           " " " " " "
453           " " " " " "
454           " " " " " "
455           " " " " " "
456           " " " " " "
457           " " " " " "
458           " " " " " "
459           " " " " " "
460           " " " " " "
461           " " " " " "
462           " " " " " "
463           " " " " " "
464           " " " " " "
465           " " " " " "
466           " " " " " "
467           " " " " " "
468           " " " " " "
469           " " " " " "
470           " " " " " "
471           " " " " " "
472           " " " " " "
473           " " " " " "
474           " " " " " "
475           " " " " " "
476           " " " " " "
477           " " " " " "
478           " " " " " "
479           " " " " " "
480           " " " " " "
481           " " " " " "
482           " " " " " "
483           " " " " " "
484           " " " " " "
485           " " " " " "
486           " " " " " "
487           " " " " " "
488           " " " " " "
489           " " " " " "
490           " " " " " "
491           " " " " " "
492           " " " " " "
493           " " " " " "
494           " " " " " "
495           " " " " " "
496           " " " " " "
497           " " " " " "
498           " " " " " "
499           " " " " " "
500           " " " " " "
501           " " " " " "
502           " " " " " "
503           " " " " " "
504           " " " " " "
505           " " " " " "
506           " " " " " "
507           " " " " " "
508           " " " " " "
509           " " " " " "
510           " " " " " "
511           " " " " " "
512           " " " " " "
513           " " " " " "
514           " " " " " "
515           " " " " " "
516           " " " " " "
517           " " " " " "
518           " " " " " "
519           " " " " " "
520           " " " " " "
521           " " " " " "
522           " " " " " "
523           " " " " " "
524           " " " " " "
525           " " " " " "
526           " " " " " "
527           " " " " " "
528           " " " " " "
529           " " " " " "
530           " " " " " "
531           " " " " " "
532           " " " " " "
533           " " " " " "
534           " " " " " "
535           " " " " " "
536           " " " " " "
537           " " " " " "
538           " " " " " "
539           " " " " " "
540           " " " " " "
541           " " " " " "
542           " " " " " "
543           " " " " " "
544           " " " " " "
545           " " " " " "
546           " " " " " "
547           " " " " " "
548           " " " " " "
549           " " " " " "
550           " " " " " "
551           " " " " " "
552           " " " " " "
553           " " " " " "
554           " " " " " "
555           " " " " " "
556           " " " " " "
557           " " " " " "
558           " " " " " "
559           " " " " " "
560           " " " " " "
561           " " " " " "
562           " " " " " "
563           " " " " " "
564           " " " " " "
565           " " " " " "
566           " " " " " "
567           " " " " " "
568           " " " " " "
569           " " " " " "
570           " " " " " "
571           " " " " " "
572           " " " " " "
573           " " " " " "
574           " " " " " "
575           " " " " " "
576           " " " " " "
577           " " " " " "
578           " " " " " "
579           " " " " " "
580           " " " " " "
581           " " " " " "
582           " " " " " "
583           " " " " " "
584           " " " " " "
585           " " " " " "
586           " " " " " "
587           " " " " " "
588           " " " " " "
589           " " " " " "
590           " " " " " "
591           " " " " " "
592           " " " " " "
593           " " " " " "
594           " " " " " "
595           " " " " " "
596           " " " " " "
597           " " " " " "
598           " " " " " "
599           " " " " " "
600           " " " " " "
601           " " " " " "
602           " " " " " "
603           " " " " " "
604           " " " " " "
605           " " " " " "
606           " " " " " "
607           " " " " " "
608           " " " " " "
609           " " " " " "
610           " " " " " "
611           " " " " " "
612           " " " " " "
613           " " " " " "
614           " " " " " "
615           " " " " " "
616           " " " " " "
617           " " " " " "
618           " " " " " "
619           " " " " " "
620           " " " " " "
621           " " " " " "
622           " " " " " "
623           " " " " " "
624           " " " " " "
625           " " " " " "
626           " " " " " "
627           " " " " " "
628           " " " " " "
629           " " " " " "
630           " " " " " "
631           " " " " " "
632           " " " " " "
633           " " " " " "
634           " " " " " "
635           " " " " " "
636           " " " " " "
637           " " " " " "
638           " " " " " "
639           " " " " " "
640           " " " " " "
641           " " " " " "
642           " " " " " "
643           " " " " " "
644           " " " " " "
645           " " " " " "
646           " " " " " "
647           " " " " " "
648           " " " " " "
649           " " " " " "
650           " " " " " "
651           " " " " " "
652           " " " " " "
653           " " " " " "
654           " " " " " "
655           " " " " " "
656           " " " " " "
657           " " " " " "
658           " " " " " "
659           " " " " " "
660           " " " " " "
661           " " " " " "
662           " " " " " "
663           " " " " " "
664           " " " " " "
665           " " " " " "
666           " " " " " "
667           " " " " " "
668           " " " " " "
669           " " " " " "
670           " " " " " "
671           " " " " " "
672           " " " " " "
673           " " " " " "
674           " " " " " "
675           " " " " " "
676           " " " " " "
677           " " " " " "
678           " " " " " "
679           " " " " " "
680           " " " " " "
681           " " " " " "
682           " " " " " "
683           " " " " " "
684           " " " " " "
685           " " " " " "
686           " " " " " "
687           " " " " " "
688           " " " " " "
689           " " " " " "
690           " " " " " "
691           " " " " " "
692           " " " " " "
693           " " " " " "
694           " " " " " "
695           " " " " " "
696           " " " " " "
697           " " " " " "
698           " " " " " "
699           " " " " " "
700           " " " " " "
701           " " " " " "
702           " " " " " "
703           " " " " " "
704           " " " " " "
705           " " " " " "
706           " " " " " "
707           " " " " " "
708           " " " " " "
709           " " " " " "
710           " " " " " "
711           " " " " " "
712           " " " " " "
713           " " " " " "
714           " " " " " "
715           " " " " " "
716           " " " " " "
717           " " " " " "
718           " " " " " "
719           " " " " " "
720           " " " " " "
721           " " " " " "
722           " " " " " "
723           " " " " " "
724           " " " " " "
725           " " " " " "
726           " " " " " "
727           " " " " " "
728           " " " " " "
729           " " " " " "
730           " " " " " "
731           " " " " " "
732           " " " " " "
733           " " " " " "
734           " " " " " "
735           " " " " " "
736           " " " " " "
737           " " " " " "
738           " " " " " "
739           " " " " " "
740           " " " " " "
741           " " " " " "
742           " " " " " "
743           " " " " " "
744           " " " " " "
745           " " " " " "
746           " " " " " "
747           " " " " " "
748           " " " " " "
749           " " " " " "
750           " " " " " "
751           " " " " " "
752           " " " " " "
753           " " " " " "
754           " " " " " "
755           " " " " " "
756           " " " " " "
757           " " " " " "
758           " " " " " "
759           " " " " " "
760           " " " " " "
761           " " " " " "
762           " " " " " "
763           " " " " " "
764           " " " " " "
765           " " " " " "
766           " " " " " "
767           " " " " " "
768           " " " " " "
769           " " " " " "
770           " " " " " "
771           " " " " " "
772           " " " " " "
773           " " " " " "
774           " " " " " "
775           " " " " " "
776           " " " " " "
777           " " " " " "
778           " " " " " "
779           " " " " " "
780           " " " " " "
781           " " " " " "
782           " " " " " "
783           " " " " " "
784           " " " " " "
785           " " " " " "
786           " " " " " "
787           " " " " " "
788           " " " " " "
789           " " " " " "
790           " " " " " "
791           " " " " " "
792           " " " " " "
793           " " " " " "
794           " " " " " "
795           " " " " " "
796           " " " " " "
797           " " " " " "
798           " " " " " "
799           " " " " " "
800           " " " " " "
801           " " " " " "
802           " " " " " "
803           " " " " " "
804           " " " " " "
805           " " " " " "
806           " " " " " "
807           " " " " " "
808           " " " " " "
809           " " " " " "
810           " " " " " "
811           " " " " " "
812           " " " " " "
813           " " " " " "
814           " " " " " "
815           " " " " " "
816           " " " " " "
817           " " " " " "
818           " " " " " "
819           " " " " " "
820           " " " " " "
821           " " " " " "
822           " " " " " "
823           " " " " " "
824           " " " " " "
825           " " " " " "
826           " " " " " "
827           " " " " " "
828           " " " " " "
829           " " " " " "
830           " " " " " "
831           " " " " " "
832           " " " " " "
833           " " " " " "
834           " " " " " "
835           " " " " " "
836           " " " " " "
837           " " " " " "
838           " " " " " "
839           " " " " " "
840           " " " " " "
841           " " " " " "
842           " " " " " "
843           " " " " " "
844           " " " " " "
845           " " " " " "
846           " " " " " "
847           " " " " " "
848           " " " " " "
849           " " " " " "
850           " " " " " "
851           " " " " " "
852           " " " " " "
853           " " " " " "
854           " " " " " "
855           " " " " " "
856           " " " " " "
857           " " " " " "
858           " " " " " "
859           " " " " " "
860           " " " " " "
861           " " " " " "
862           " " " " " "
863           " " " " " "
864           " " " " " "
865           " " " " " "
866           " " " " " "
867           " " " " " "
868           " " " " " "
869           " " " " " "
870           " " " " " "
871           " " " " " "
872           " " " " " "
873           " " " " " "
874           " " " " " "
875           " " " " " "
876           " " " " " "
877           " " " " " "
878           " " " " " "
879           " " " " " "
880           " " " " " "
881           " " " " " "
882           " " " " " "
883           " " " " " "
884           " " " " " "
885           " " " " " "
886           " " " " " "
887           " " " " " "
888           " " " " " "
889           " " " " " "
890           " " " " " "
891           " " " " " "
892           " " " " " "
893           " " " " " "
894           " " " " " "
895           " " " " " "
896           " " " " " "
897           " " " " " "
898           " " " " " "
899           " " " " " "
900           " " " " " "
901           " " " " " "
902           " " " " " "
903           " " " " " "
904           " " " " " "
905           " " " " " "
906           " " " " " "
907           " " " " " "
908           " " " " " "
909           " " " " " "
910           " " " " " "
911           " " " " " "
912           " " " " " "
913           " " " " " "
914           " " " " " "
915           " " " " " "
916           " " " " " "
917           " " " " " "
918           " " " " " "
919           " " " " " "
920           " " " " " "
921           " " " " " "
922           " " " " " "
923           " " " " " "
924           " " " " " "
925           " " " " " "
926           " " " " " "
927           " " " " " "
928           " " " " " "
929           " " " " " "
930           " " " " " "
931           " " " " " "
932           " " " " " "
933           " " " " " "
934           " " " " " "
935           " " " " " "
936           " " " " " "
937           " " " " " "
938           " " " " " "
939           " " " " " "
940           " " " " " "
941           " " " " " "
942           " " " " " "
943           " " " " " "
944           " " " " " "
945           " " " " " "
946           " " " " " "
947           " " " " " "
948           " " " " " "
949           " " " " " "
950           " " " " " "
951           " " " " " "
952           " " " " " "
953           " " " " " "
954           " " " " " "
955           " " " " " "
956           " " " " " "
957           " " " " " "
958           " " " " " "
959           " " " " " "
960           " " " " " "
961           " " " " " "
962           " " " " " "
963           " " " " " "
964           " " " " " "
965           " " " " " "
966           " " " " " "
967           " " " " " "
968           " " " " " "
969           " " " " " "
970           " " " " " "
971           " " " " " "
972           " " " " " "
973           " " " " " "
974           " " " " " "
975           " " " " " "
976           " " " " " "
977           " " " " " "
978           " " " " " "
979           " " " " " "
980           " " " " " "
981           " " " " " "
982           " " " " " "
983           " " " " " "
984           " " " " " "
985           " " " " " "
986           " " " " " "
987           " " " " " "
988           " " " " " "
989           " " " " " "
990           " " " " " "
991           " " " " " "
992           " " " " " "
993           " " " " " "
994           " " " " " "
995           " " " " " "
996           " " " " " "
997           " " " " " "
998           " " " " " "
999           " " " " " "
1000          " " " " " "

```

```

1 class WordGenerator:
2     def __init__(self):
3         self.alpha_gen = AlphabetGenerator()
4
5     def run(self, word):
6         points = []
7         cw = 0
8         for letter in word:
9             pt, w = self.alpha_gen.run(letter)
10            pt[1,0] += cw
11            points.append(pt)
12            cw += w
13        return np.vstack(points)

```

```

1 class SentenceASCIIGenerator(SceneProgObject):
2     def __init__(self, scene):
3
4         self.name = "SentenceASCIIGenerator"
5         self.description = """
6         Places assets in an ASCII art representation of a sentence.
7         Inputs:
8         - obj: An object to place in the scene.
9         - sentence: The sentence to represent in ASCII art.
10        """
11        self.usage = """
12        with scene.SentenceASCIIGenerator() as ascii_gen:
13            plant = scene.AddAsset("A large potted plant")
14            ascii_gen.place(plant, sentence="World's best")
15        """
16        self.word_gen = WordGenerator()
17        super().__init__(scene)
18
19    def run(self, sentence):
20        points = []
21        ch = 0
22        for line in sentence.split('\n'):
23            cw = 0
24            for word in line.split(' '):
25                pt = self.word_gen.run(word)
26                h = np.max(pt[1,1])+1
27                w = np.max(pt[1,0])+5
28                pt[1,1] += ch
29                pt[1,0] += cw
30                tmp.append(pt)
31                cw += w
32            tmpmp.vstack(tmp)
33            points.append(tmp)
34            ch += h
35        return points
36
37    @staticmethod
38    def place(self, obj, sentence):
39        points = self.run(sentence)
40        total_points = np.vstack(points).shape[0]
41        objs = total_points*obj
42        height = self.compute_obj_y(obj)
43        count = 0
44        from tqdm import tqdm
45        for line in tqdm(points):
46            for pt in tqdm(line):
47                obj[count].set_location(pt[0], height, pt[1])
48                self.add_child(objs[count])
49                count += 1
50        return points
51
52    def compile(self):
53        if self.operation_order is None:
54            self.operation_order = [key for key in self.operations.keys() if self.operations[key] is not None]
55        for key in self.operation_order:
56            if key in self.operations:
57                if self.operations[key] is not None:
58                    op = self.operations[key]
59                    op.execute()

```

```

1 with scene.SentenceASCIIGenerator() as ascii_gen:
2     sentence = "INTERIORAGENT\n3DV\t2026\nVANCOUVER"
3     plant = scene.AddAsset("A large cherry blossom tree")
4     ascii_gen.place(plant, sentence=sentence)

```

Figure 11. **INTERIORAGENT tool use example:** Using the Group template (1–3), we implement an ASCII art generation tool driven by an LLM prompt. The teaser illustration was produced from the caption “*forest made to look like INTERIORAGENT 3DV 2026 VANCOUVER*”. (4) INTERIORAGENT enables such expressive scenes with minimal code.

```

1 class RenderingConstraint(ConstraintBase):
2     def __init__(self, group, wall, paintings, target_image_path):
3         from painting_detector import PaintingDetector
4         self.name = 'RenderingConstraint'
5         self.description = f"""
6 Helps in optimizing placement of paintings on the wall to match a given image target.
7 Inputs:
8 - wall: The wall name (str)
9 - paintings: List of painting objects (list)
10 - target_image_path: Path to the target image (str)
11 """
12         self.examples = f"""
13 with scene.RoomGroup() as room:
14     ...
15     painting = scene.AddAsset("A Beautiful Landscape")
16     paintings = 3*paintings
17     room.place_on_wall_back_left(paintings[0])
18     room.place_on_wall_back_center(paintings[1])
19     room.place_on_wall_back_right(paintings[2])
20
21     room.RenderingConstraint("back_wall", paintings, "path/to/target/image.jpg")
22 """
23
24         self.type = 'GRADIENT'
25         self.painting_detector = PaintingDetector()
26         self.target_image_path = target_image_path
27
28         self.target_centroids, self.target_bbox = self.painting_detector(self.target_image_path, resize=(1920,1080))
29         self.wall = wall
30         self.paintings = paintings
31
32         super().__init__(self.name, group)
33
34     def compute_gradients(self):
35         ## Render the wall with paintings
36         current_image_path = self.group.render_wall(self.wall, self.paintings)
37         ## Detect centroids of each painting using OwlV2
38         centroids, tmp = self.painting_detector(current_image_path, resize=(1920,1080))
39
40         ## Use hungarian method to derieve optimal 1-1 mapping between centroids.
41         perm = self.painting_detector.compute_mapping(centroids, self.target_centroids)
42         mapped_centroids = [self.target_centroids[i] for i in perm]
43
44         for i, painting in enumerate(self.paintings):
45             grad = mapped_centroids[i] - centroids[i] ## pseudo gradient
46             img_grad[0] *= 1/1920
47             img_grad[1] *= 1/1080
48             painting.grad += np.array([img_grad[0], img_grad[1], 0], dtype=np.float32)

```

Figure 12. **INTERIORAGENT tool use example:** Using the Constraints template, we implement an optimization routine for arranging wall scenery to match a target image. Frames are detected with OWLv2 using prompts such as “painting”, “picture frame”, “wall art”, and “poster”. The centers of bounding boxes from target and source renderings are extracted, matched via the Hungarian algorithm, and the scenery is shifted toward their assigned centroids.



Figure 13. **Using Rendering constraint in code:** INTERIORAGENT uses the available context to easily write a program for the caption “Create a living room with the back wall having three paintings: symbolizing Christianity, Buddhism and Hinduism as per the following arrangement (pass image path for target)”.

"Create a minimal dining scene"



"Switch one of the chairs with a yellow chair"

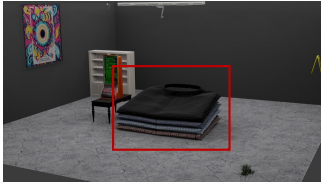


"Improve the aesthetics of the scene"



"Move dining area close to window and add a sideboard"

Figure 14. INTERIORAGENT allows user-driven scene customization through a chat interface. Starting from a minimal scene, INTERIORAGENT responds to user inputs to change colors, rearrange, add furniture and improve decor, while maintaining the scene balance.



Wrong retrieval



Missed visibility



Missed clearance



Optimization too difficult

Figure 15. Some failure cases of INTERIORAGENT .



Figure 16. A few in-context examples used in INTERIORAGENT .