MageAdd: Real-Time Interaction Simulation for Scene Synthesis Supplementary Materials

Shao-Kui Zhang
zhangsk18@mails.tsinghua.edu.cn
Tsinghua University
China

Yi-Xiao Li
liyixiao20@mails.tsinghua.edu.cn
Tsinghua University
China

Yu He
hooyeeevan@tsinghua.edu.cn
Tsinghua University
China

Yong-Liang Yang
y.yang@cs.bath.ac.uk
University of Bath
United Kingdom

Song-Hai Zhang*
shz@tsinghua.edu.cn
Tsinghua University & BNRist
China

ACM Reference Format:
Shao-Kui Zhang, Yi-Xiao Li, Yu He, Yong-Liang Yang, and Song-Hai Zhang. 2021. MageAdd: Real-Time Interaction Simulation for Scene Synthesis Supplementary Materials. In *Proceedings of the 29th ACM International Conference on Multimedia (MM '21), October 20–24, 2021, Virtual Event, China.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3474085.3475194

A OVERVIEW

This supplementary materials contains the following contents that strengthen our contributions:

- **Section B**: The details of computing the nearest wall $v_{(1)}$ and the second nearest wall $v_{(2)}$ with respective distances $d_{(1)}$ and $d_{(2)}$ in the section 4: Dominant Objects.
- **Section C**: The minor concerns that we addressed to achieve real-time performance and plausibility in practice, including collision avoidance, priority blocks, filtering object, etc.
- **Section D**: Figure 5, 6, 7 and 8 present the open-source platform of this paper for conducting experiments.
- **Section E**: More details on the qualitative and quantitative comparisons between three baselines and ours, including PlanIT [2], 3D-Front [1] and the Geometry-Based Approach [3]. The qualitative comparisons are shown at the end of this document.
- **Section F**: The statistical analysis with respect to the experiments in section 7.2 and 7.3 in the main paper.
- More qualitative results are attached separately through the online submission system.
- A demo video is attached separately through the online submission system, to show the proposed framework.

*Corresponding Author, with Tsinghua University and Beijing National Research Center for Information Science and Technology (BNRist).

B THE CALCULATION OF NEAREST WALLS

We devise Algorithm 1 to find the nearest vector $v_{(1)}$ and the second nearest vector $v_{(2)}$ of $\mathbf{\Lambda}$ and the respective distances $d_{(1)}$ and $d_{(2)}$. Since no other subordinate object or wall object is inserted before, the intersection point for placing dominant object is required to touch the ground [1], i.e., $\mathbf{\Lambda} = (x_{\mathbf{\Lambda}}, 0, z_{\mathbf{\Lambda}})$. Thus, we use $\mathbf{\Gamma} = (x_{\mathbf{\Lambda}}, z_{\mathbf{\Lambda}})$ to denote the intersection point on the ground.

We filter out wall vectors with inappropriate $TwiceInner(\mathbf{\Gamma}, v)$, which returns the inner product of $(v^1 - \mathbf{\Gamma})$ and $(v^1 - v^2)$ as shown in Eqn. 1, where v^1 and v^2 are endpoints of walls. The reason of term $TwiceInner(\cdot)$ is shown in Figure 1. It is obvious that the wall with green vector is the nearest wall to the desk, but according to $Tri(\cdot)$, the wall with red vector results in the lowest triangle area. Next, we computes distances of $\mathbf{\Gamma}$ with respect to the remaining wall vectors, where $wallDis = Tri(\mathbf{\Gamma}, v)/|v|$. $Tri(\mathbf{\Gamma}, v)$ is the area of the triangle formed by v^1, v^2 and $\mathbf{\Gamma}$. We use Heron's formula

[1] In real dataset, grounds in room meshes are not guaranteed to coincide with the XoZ plane, which requires a pre-alignment in practice.

ALGORITHM 1: Finding the indices and distances of the nearest wall and the second nearest vector (wall).

Input: Cursor $\mathbf{\Gamma}$, Shape of the room as vectors $\{v_i | i = 1, 2, ..., n\}$.
Output: The nearest vector(wall) $v_{(1)}$ and the second nearest vector $v_{(2)}$, the nearest distance $d_{(1)}$ and the second nearest distance $d_{(2)}$.

```
potentialWalls = [];
for Each vector v_i do
    t = TwiceInner(Γ, v_i);
    if 0 <= t and t <= |v_i| then
        potentialWalls.push(V_i);
    end
end
distances = [];
for Each vector V_i in potentialWalls do
    distances.push(wallDis(Γ, V_i));
end
wallIndices = argSort(distances);
v_(1) = potentialWalls[wallIndices[1]];
v_(2) = potentialWalls[wallIndices[2]];
d_(1) = distances[wallIndices[1]];
d_(2) = distances[wallIndices[2]];
```

for $Tri(\cdot)$ in Eqn. 2, where the additional $ReLU(x) = \max(0, x)$ is for numerical stability. Therefore, after sorting distances from the lowest to the highest with *argSort* and achieving *wallIndices*, the results are indexed accordingly using *wallIndices*.

$$TwiceInner(\mathbf{\Gamma}, v) = (v^1 - \mathbf{\Gamma}) \cdot (v^1 - v^2) \tag{1}$$

$$A = \frac{1}{4}\sqrt{ReLU(4a^2b^2 - (a^2 + b^2 - c^2)^2)} \tag{2}$$

C OTHER PRACTICAL CONCERNS

Collision Avoidance. The first concern is collision detection since a tiny collision may destructively influence results. Especially, the results of 3D scene synthesis are subjective to human. Thus, we should restrict the cursor from suggesting objects that may collide with existing objects in the scene. In our framework, three types of collisions are considered: "Object-Object", "Object-Wall", and "Object-Door/Window". First, a collision detection between a pair of objects is conducted based on their bounding boxes of constituent components. For example, to detect a potential collision between a desk and a office chair, the desk is formed by several components such as the desk top and legs. The chair is decomposed similarly. Each component has its own bounding box. As a result, they collide only if any component of the desk intersects one or more components of the chair. Second, we detect collision of a wall with an object by casting a ray from one endpoint v^1 of the wall to another endpoint v^2. The ray is generated over half the height of the object. The object is also decomposed into bounding boxes of components. Therefore, they collide only if any component of the object collides with the ray. Third, we also detect collision between objects and doors/windows, since we deal with indoor scenes. As shown in Figure 2, windows and doors are first represented as cube objects elevated in the normal direction of their corresponding walls, followed by the aforementioned collision detection between two objects. For windows, we introduce a hyper-parameter W_{cut} that cuts off them from their bottom, because objects such as corner/side tables can overlap windows partially without entirely blocking them.

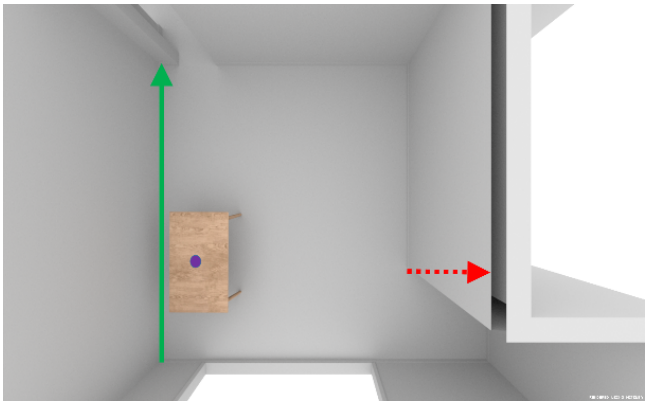


Figure 1: The problem of finding a nearest wall: based merely on "triangle area" to compute nearest walls is defective where the nearest vector is computed incorrectly as the red one.

In our experiments, we set $W_{cut} = 0.5$ which allows cursors to overlap half of windows from their bottom. After collision detection, if a collision happens between a suggested object and the scene, the suggested object is removed and the cursor should attempt next iteration.

Priority Blocks. As our framework suggests subordinate objects before dominant objects if an intersection is associated with both sub-priors and dom-priors, this may "block" suggestions of dominant objects. As shown in Figure 3, a TV stand is sufficient for the coherent group led by a double bed, thus another suggestion of TV stand blocks the suggestion of a new corner/side table which is intuitively a more reasonable suggestion. We fix this problem by restricting the quantity of subordinate objects of coherent groups. If the number of copies of a subordinate object o_{sec} increases to a threshold $T_{sec} \in [1, +\infty]$, the pattern of o_{sec} is removed from the prior base until the number of copies decreases.

Instance Swapping. Instance swapping is optional to end users and we do not swap instances if computers take the control of the cursor. By right-clicking, a new candidate will be chosen among objects with the same category of the current suggestion. However, a

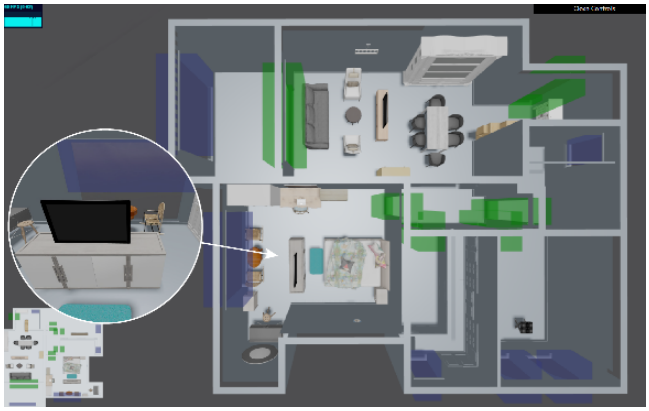


Figure 2: Window-blocks (Blue) and door-blocks (Green) for avoiding barricading paths of them.

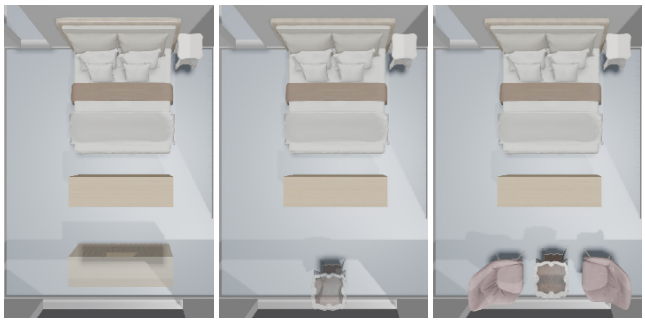


Figure 3: Priority Blocks. LEFT: after placing a TV stand, the pattern suggests another TV stand. MIDDLE and RIGHT: by restricting maximum quantities of copies of subordinate objects, priors of subordinate objects no longer block inference of other coherent groups.

category in datasets typically contains hundreds of 3D models with high-quality textures, which may cause memory and I/O overflow. Furthermore, it is likely that end users simply want to compare several candidates and choose one instead of exhaustively traversing the entire dataset. To address this, we design a swapping strategy as shown in Figure 4. We maintain a queue for each category when performing our method. Each time of right-clicking, an alternative object is selected, from the front to the end of the candidate object list. When a queue is completely traversed, a new model is inserted to the queue and a new traversal starts from the next right-click. An additional cache is maintained for loading shapes and textures in the background. A new model is selected from the cache prior and loaded from the database. Besides, we also allow end users to claim a particular instance if needed.

Object Filtering. Occasionally, due to various shapes and priors of objects, a small neighborhood of a ray-casted position may involve multiple objects. If the cursor has a tiny movement, emerged objects are swapped abruptly in-between, which may influence user experiences because end users would have to spend time on tuning the cursor in order to acquire the expected object. Therefore, we allow end users to specify a certain object. Our method then filters out all other unrelated priors. Note that this is no longer applicable if the cursor is taken control by the computer, which simply calculate an absolute position.

Interactive Thresholds. When applying our method for arranging objects, we introduce three hyper-parameters respectively: $THRES_{dom}$, $THRES_{sub}$, $THRES_{wall}$. Different thresholds are actually trade-offs between layout accuracy and user satisfaction. If we set thresholds too small, it does increase the plausibility since we want objects to strictly follow the priors. However, it limits the available area for user interactions, i.e., end users need to carefully tune their cursors in order to get a plausible suggestion. Thus, we set thresholds depending on scenarios. $THRES_{dom}$ equals to 0.1 in automatic mode, 0.6 when end users control the cursor, and 1.5 when filtering objects. $THRES_{sub}$ equals to 0.05 in automatic mode, 0.6 when end users control the cursor, and 1.5 when filtering objects. $THRES_{wall}$ equals to 0.1 for all scenarios. Setting the thresholds is empirical, and the choices of thresholds are not limited to the above values that are used in our experiments.

D THE OPEN-SOURCE PLATFORM

Figure 5, 6, 7 and 8 show different features of our platform. In addition to basic interactions, our platform also supports global illumination in the background. The code of this platform together with the proposed method will be publicly available.

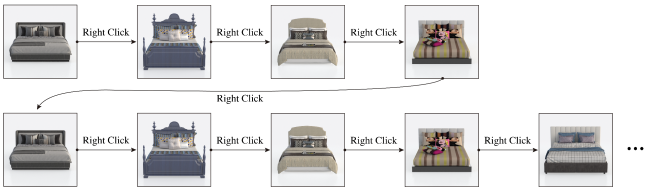


Figure 4: Each time of swapping instances, a new object is selected according to a list. The list is populated if it is traversed entirely to alleviate I/O traffic.

Figure 5: The overview of our platform. Our platform support rendering scenes with both local and global illuminations. The former is for efficient manipulations at the front-end. The latter is run in the back-ground and yield more photo-realistic results.

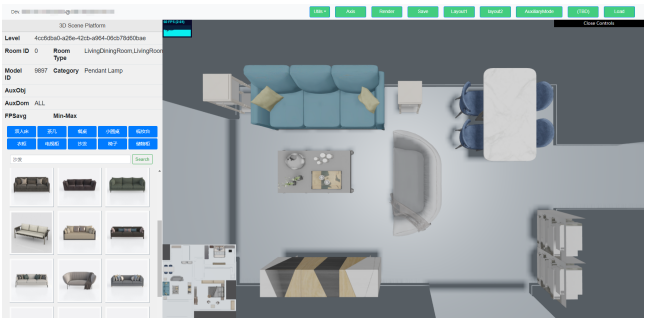


Figure 6: Searching objects. Users can search objects by keywords or shortcut buttons.

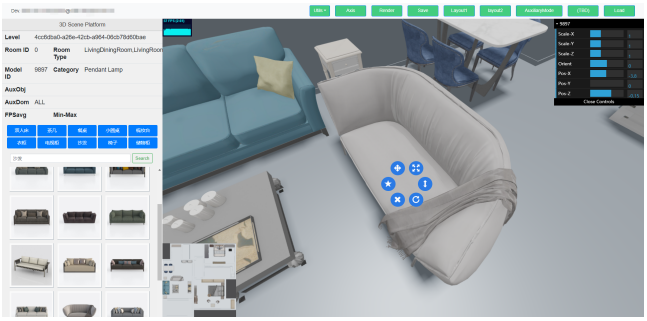


Figure 7: Manipulating objects. Objects can be manipulated by a roulette (Middle) or a panel (Top-Right), including translation, rotation and scale on three axis respectively. The former is more intuitive, where objects are transformed following the cursor. The latter is more numeric, where objects are set according to specific values.

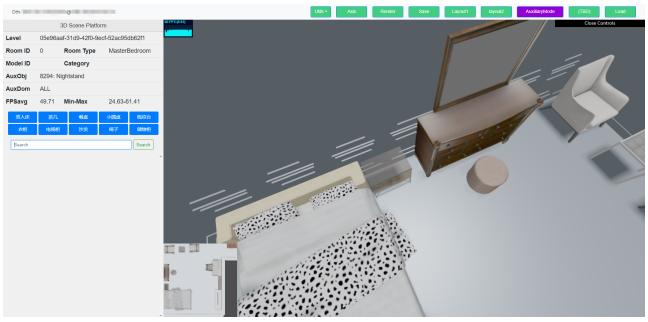


Figure 8: By clicking the "auxiliary mode" button, our method is run in the background and various objects are emerged with the cursor. See also the demo video. The source code will be publicly available.

E QUALITATIVE AND QUANTITATIVE COMPARISONS

The qualitative comparisons are shown in the end of this supplementary document. Each row of images show results of 3D-Front [1], PlanIT [2], GBA [3] and ours respectively. The measurement of scene quality is multi-modal. Thus, we first conduct the qualitative comparisons from the user feedback to illustrate the common concerns, e.g., the "leftover space", "room capacity" and "unexpected object-object relation". Subsequently, we also conducted a user study to analyze quantitative ratings on general aesthetics and plausibility according to user experience.

F STATISTICAL ANALYSIS

Table 1 and 2 show the Kruskal-Wallis H-Test with respect to the experiments in the main paper (See Section 7.2 and 7.3 of the main paper). According to the Kruskal-Wallis H-Test, there are significant statistical differences between our method and completing baselines.

REFERENCES

[1] Huan Fu, Bowen Cai, Lin Gao, Lingxiao Zhang, Cao Li, Qixun Zeng, Chengyue Sun, Yiyun Fei, Yu Zheng, Ying Li, Yi Liu, Peng Liu, Lin Ma, Le Weng, Xiaohang Hu, Xin Ma, Qian Qian, Rongfei Jia, Binqiang Zhao, and Hao Zhang. 2020. 3D-FRONT: 3D Furnished Rooms with layOuts and semaNTics. *arXiv preprint arXiv:2011.09127* (2020).

[2] Kai Wang, Yu-An Lin, Ben Weissmann, Manolis Savva, Angel X Chang, and Daniel Ritchie. 2019. Planit: Planning and instantiating indoor scenes with relation graph and spatial prior networks. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 132.

[3] Shao-Kui Zhang, Wei-Yu Xie, and Song-Hai Zhang. 2021. Geometry-Based Layout Generation with Hyper-Relations AMONG Objects. *arXiv preprint arXiv:2101.02903* (2021).

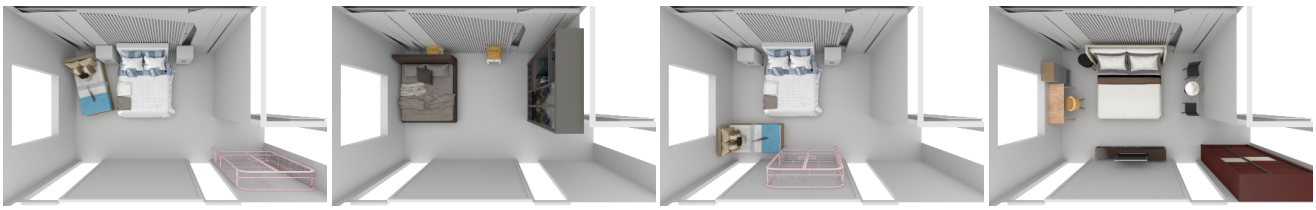
Table 1: User Satisfaction and Interactive Efficiency including the Kruskal-Wallis H-Test.

Measurement	Traditional	Ours	Kruskal-Wallis H-Test
Interactive Satisfaction	3.51 (1.06)	3.52 (1.09)	0.022 (0.882)
Result Satisfaction	3.74 (0.96)	3.75 (0.83)	0.061 (0.804)
Time Consumption	835.21 (523.05)	437.6 (301.78)	16.655 (0.0)

Table 2: The Kruskal-Wallis H-Test for the Aesthetics and Plausibility.

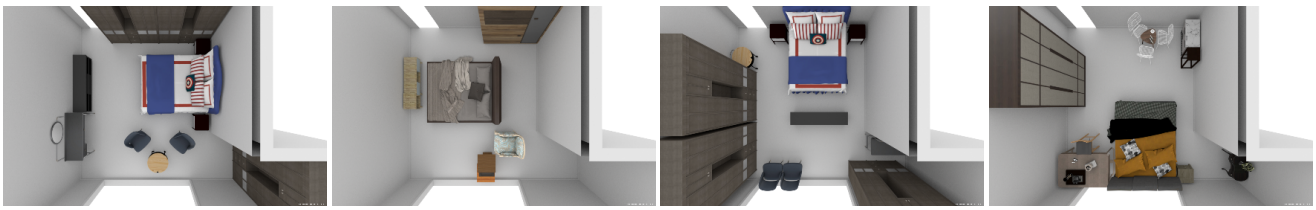
Methods	3D-Front with Ours	PlanIT with Ours	GBA with Ours
Master Bedroom	1.545 (0.214)	104.718 (0.0)	68.153 (0.0)
Second Bedroom	5.641 (0.018)	202.653 (0.0)	30.473 (0.0)
Kids Room	4.905 (0.027)	148.489 (0.0)	71.612 (0.0)
Living-Dinning Room	18.053 (0.0)	135.059 (0.0)	161.005 (0.0)
Living Room	5.003 (0.025)	159.961 (0.0)	110.913 (0.0)
Dinning Room	2.397 (0.122)	156.882 (0.0)	77.251 (0.0)
Total	3.711 (0.054)	884.731 (0.0)	492.114 (0.0)

(a) 3D-Front **(b) PlanIT** **(c) GBA** **(d) Ours**

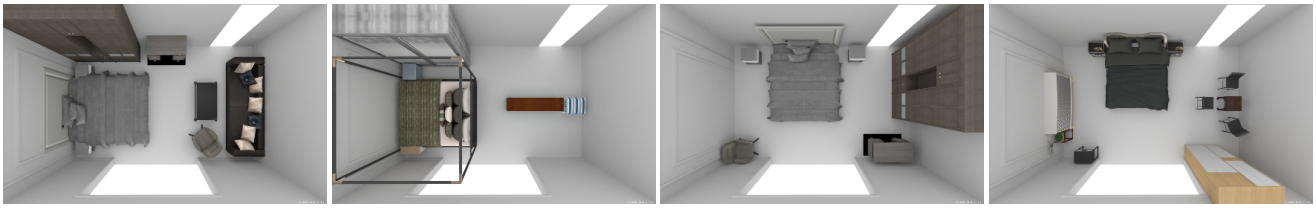


(a) 3D-Front **(b) PlanIT** **(c) GBA** **(d) Ours**

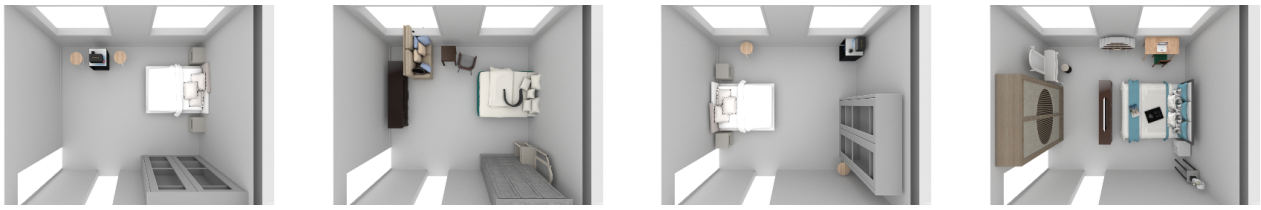
(a) 3D-Front **(b) PlanIT** **(c) GBA** **(d) Ours**



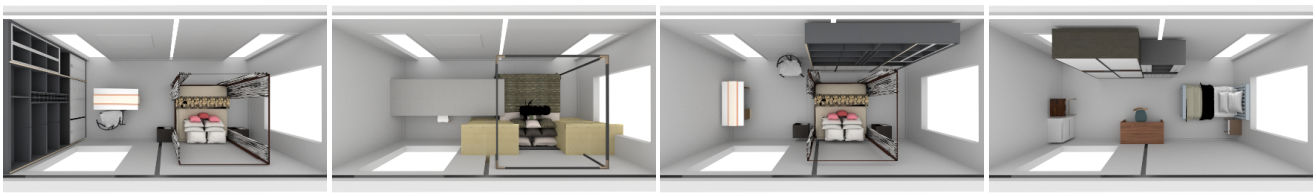
(a) 3D-Front **(b) PlanIT** **(c) GBA** **(d) Ours**



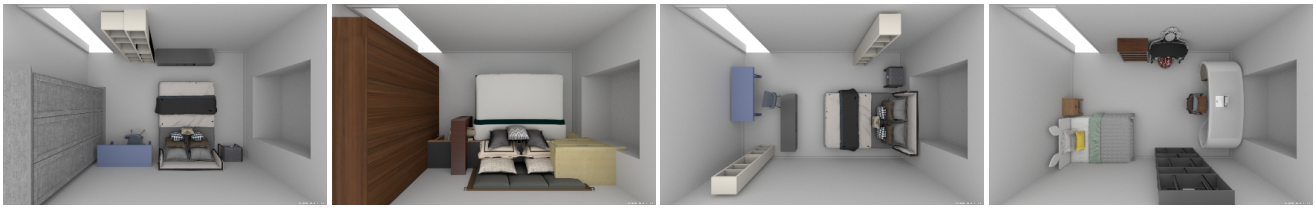
(a) 3D-Front (b) PlanIT (c) GBA (d) Ours



(a) 3D-Front (b) PlanIT (c) GBA (d) Ours

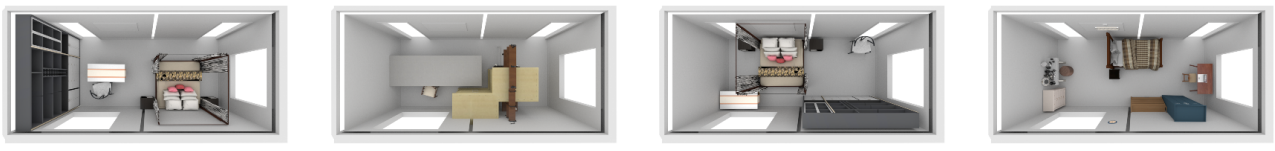


(a) 3D-Front (b) PlanIT (c) GBA (d) Ours

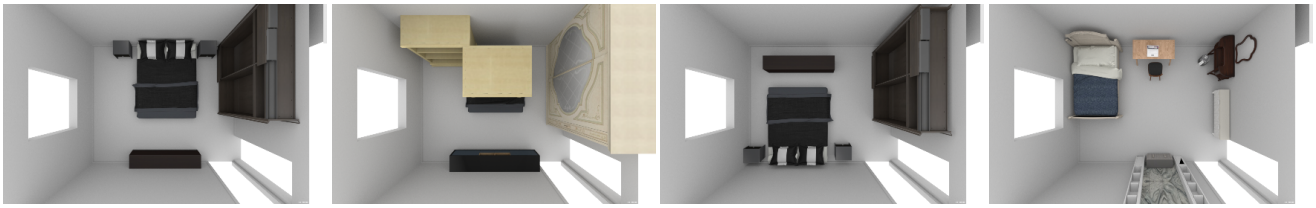


(a) 3D-Front (b) PlanIT (c) GBA (d) Ours

(a) 3D-Front (b) PlanIT (c) GBA (d) Ours

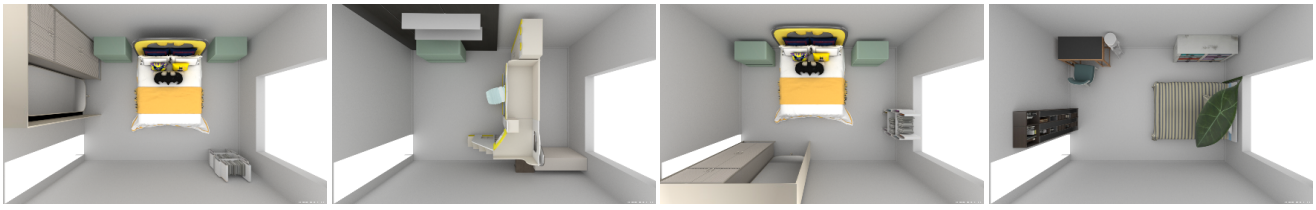


(a) 3D-Front (b) PlanIT (c) GBA (d) Ours

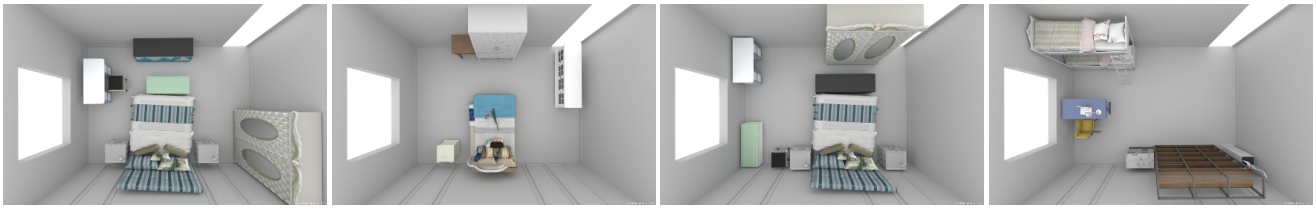


(a) 3D-Front (b) PlanIT (c) GBA (d) Ours

(a) 3D-Front (b) PlanIT (c) GBA (d) Ours



(a) 3D-Front (b) PlanIT (c) GBA (d) Ours

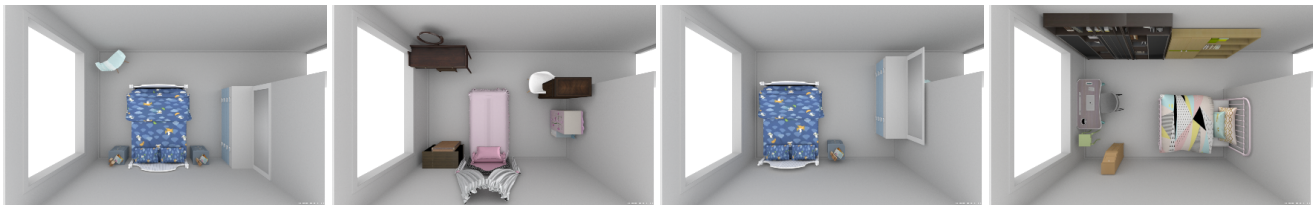


(a) 3D-Front (b) PlanIT (c) GBA (d) Ours

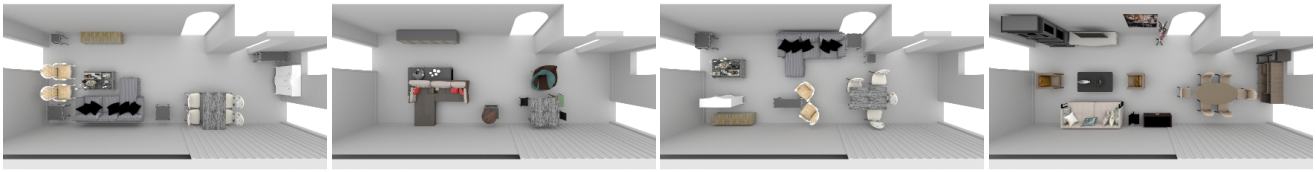
(a) 3D-Front (b) PlanIT (c) GBA (d) Ours

(a) 3D-Front (b) PlanIT (c) GBA (d) Ours

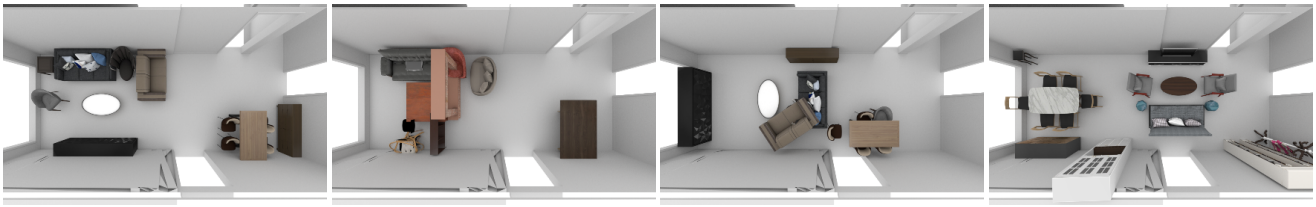
(a) 3D-Front (b) PlanIT (c) GBA (d) Ours



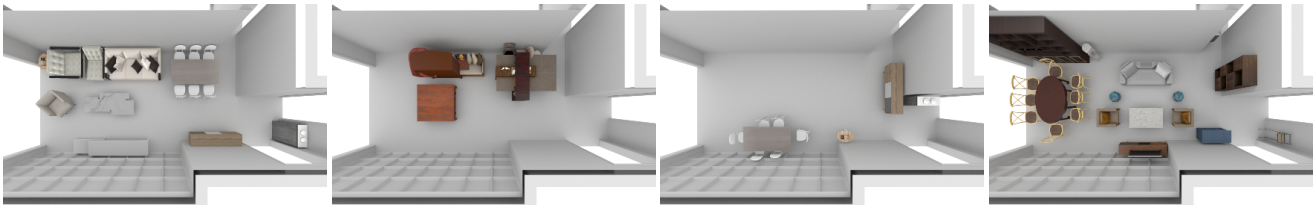
(a) 3D-Front (b) PlanIT (c) GBA (d) Ours



(a) 3D-Front (b) PlanIT (c) GBA (d) Ours

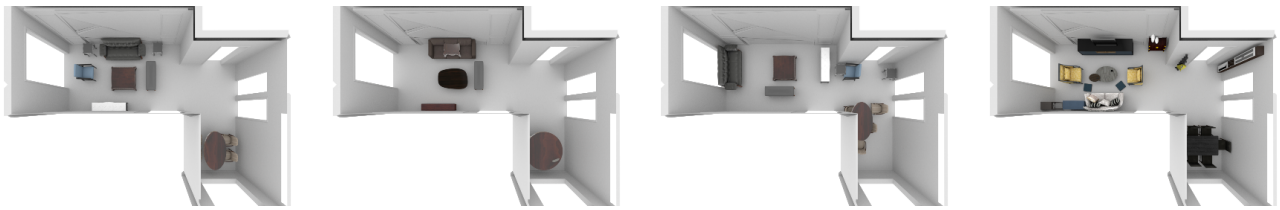


(a) 3D-Front (b) PlanIT (c) GBA (d) Ours

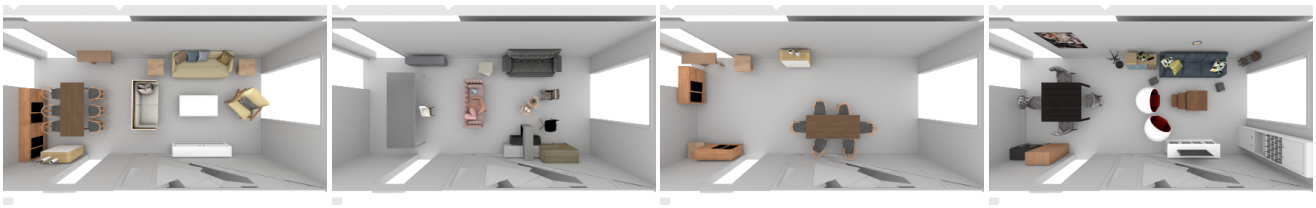


(a) 3D-Front (b) PlanIT (c) GBA (d) Ours

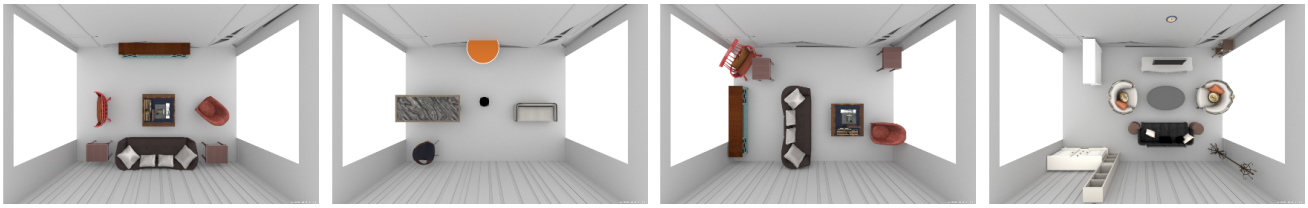
(a) 3D-Front (b) PlanIT (c) GBA (d) Ours



(a) 3D-Front (b) PlanIT (c) GBA (d) Ours

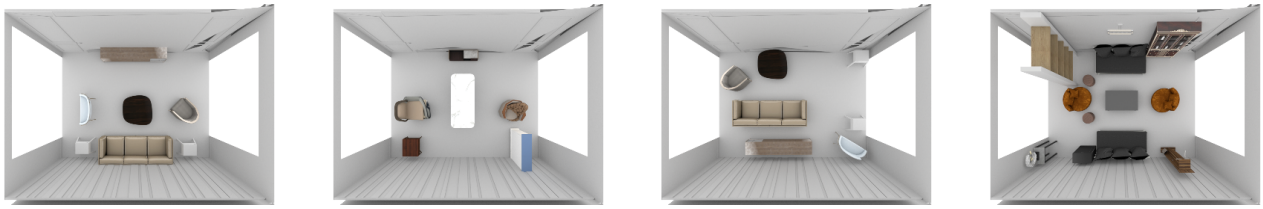


(a) 3D-Front (b) PlanIT (c) GBA (d) Ours

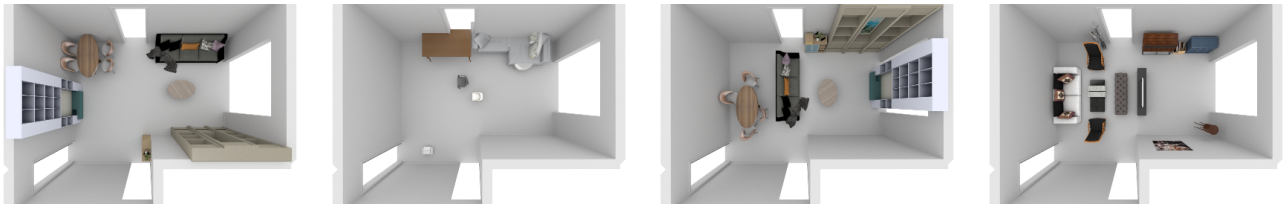


(a) 3D-Front (b) PlanIT (c) GBA (d) Ours

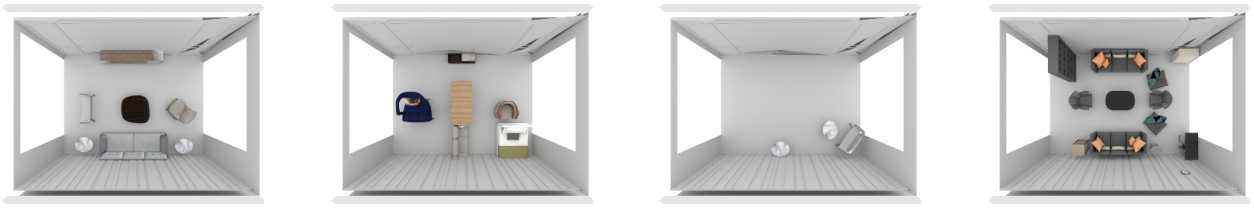
(a) 3D-Front (b) PlanIT (c) GBA (d) Ours



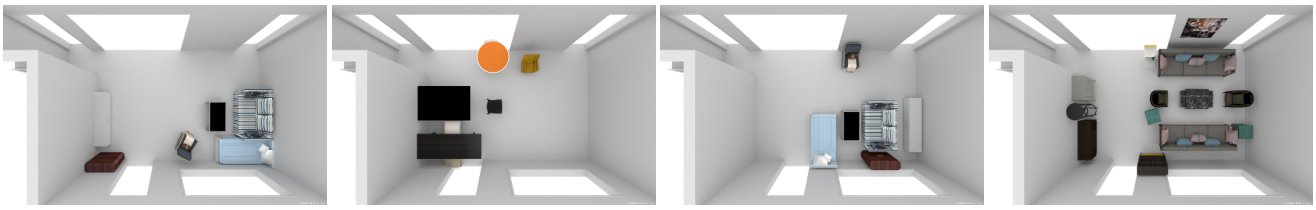
(a) 3D-Front (b) PlanIT (c) GBA (d) Ours



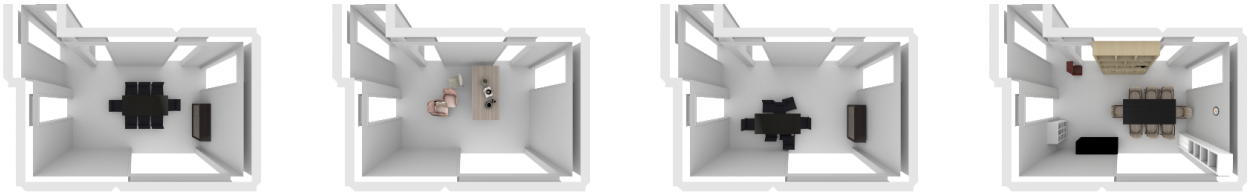
(a) 3D-Front (b) PlanIT (c) GBA (d) Ours



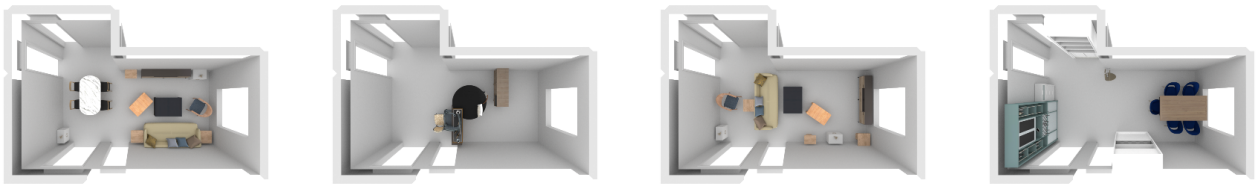
(a) 3D-Front (b) PlanIT (c) GBA (d) Ours



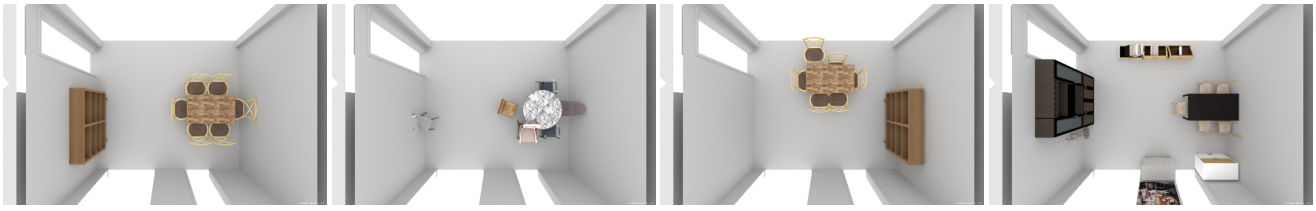
(a) 3D-Front (b) PlanIT (c) GBA (d) Ours



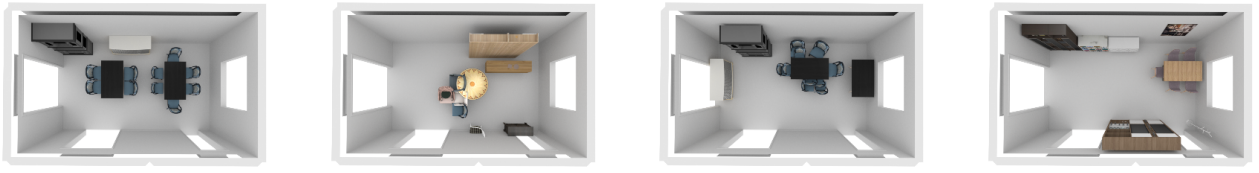
(a) 3D-Front (b) PlanIT (c) GBA (d) Ours



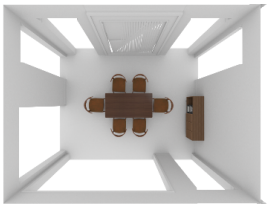
(a) 3D-Front (b) PlanIT (c) GBA (d) Ours



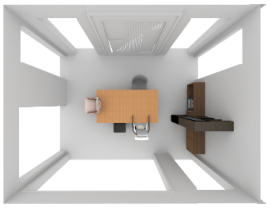
(a) 3D-Front (b) PlanIT (c) GBA (d) Ours



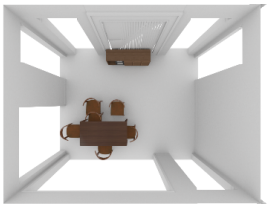
(a) 3D-Front (b) PlanIT (c) GBA (d) Ours



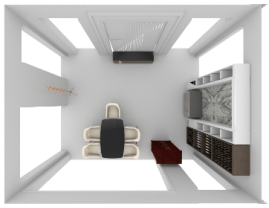
(a) 3D-Front



(b) PlanIT



(c) GBA



(d) Ours

(a) 3D-Front

(b) PlanIT

(c) GBA

(d) Ours