INDOORWORLD : Integrating Physical Task Solving and Social Simulation in A Heterogeneous Multi-Agent Environment

Dekun Wu[1,2] **Frederik Brudy**[4] **Bang Liu**[1,2,3] **Yi Wang**[4]
[1]Université de Montréal, [2]Mila - Quebec AI Institute, [3]Canada CIFAR AI Chair
[4]Autodesk Research
{dekun.wu, bang.liu}@umontreal.ca
frederik.brudy@autodesk.com, ywang485@gmail.com

Abstract

Virtual environments are essential to AI agent research. Existing environments for LLM agent research typically focus on either physical task solving or social simulation, with the former oversimplifying agent individuality and social dynamics, and the latter lacking physical grounding of social behaviors. We introduce INDOORWORLD , a heterogeneous multi-agent environment that tightly integrates physical and social dynamics. By introducing novel challenges for LLM-driven agents in orchestrating social dynamics to influence physical environments and anchoring social interactions within world states, INDOORWORLD opens up possibilities of LLM-based building occupant simulation for architectural design. We demonstrate the potential with a series of experiments within an office setting to examine the impact of multi-agent collaboration, resource competition, and spatial layout on agent behavior.

1 Introduction

The emergence of Large Language Model (LLM)-based agents has extended LLMs beyond traditional one-off interactions, equipping them with long-term memory, planning capabilities, and embodied actions (Yao et al., 2022; Shinn et al., 2024; Wang et al.). Among them, multi-agent systems leverage distinct agent roles to achieve greater collective intelligence for problem-solving (Hong et al., 2024; Qian et al., 2023; Tang et al., 2024) or to enable more realistic cognitive and psychological modeling in social simulations.

Like in traditional AI research (Maes, 1995), virtual environments are essential for LLM-based agents, enabling them to perceive and act. These environments provide external sensory input and introduce a world state, allowing agent actions to be grounded as operators that modify the environment. These environments serve as low-cost testbeds that accelerate LLM-agent development, typically falling into two categories: **physical task-solving environments** and **social simulation environments**.

Physical task-solving environments such as VirtualHome (Puig et al., 2018) and ALFWorld (Shridhar et al., 2020b) enable agents to interact with external objects to manipulate the world state towards specific objectives. However, these environments often assume identical action spaces, homogeneous agent abilities, neglecting individual differences among agents, and the impact of social dynamics on task solving.

On the other hand, social simulation environments, such as Smallville (Park et al., 2023), enable social interactions between agents, such as relationship building and information sharing. While these systems effectively simulate human-like social behaviors driven by individual personalities and roles, they often employ an oversimplified model of the physical world. This leads to a lack of groundings for agents' actions in the change of world state, resulting in social interactions that remain merely "plausible" without any correspondence to an external physical reality. For example, an agent may refer to non-existent physical objects in a dialog.

The gaps lead to missed opportunities to use AI agents for applications that require tight integration of physical task-solving and social simulation. One such application is building occupant simulation for architectural design (Yan et al., 2015; Feng et al., 2015), where occupant behaviors are driven by both dynamic physical and social factors. To respond to the missed opportunities, we present INDOORWORLD , a heterogeneous multi-agent environment that tightly integrates physical and social dynamics in an indoor space setting, introducing novel challenges for LLM-driven agents in orchestrating social dynamics to influence physical environments and anchoring social interactions within physical world states. As a multi-agent system, INDOORWORLD allows for fully decentralized agent

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 11078–11099
November 4-9, 2025

control and collective task assignment, facilitating self-regulated labor division, task prioritization and coordination. INDOORWORLD provides a scalable and expressive testbed for advancing research on multi-agent LLM systems.

Our key contributions are as follows:

- **Heterogeneous Agent Modeling:** We introduce a multi-level approach in which agents vary in roles, actions, capabilities, and knowledge, yielding individual differences that professionals judged to be more realistic.
- **Integrated Physical and Social Dynamics:** Our environment seamlessly combines physical object manipulation with social behaviors, posing novel challenges and setting the stage for developing LLM-based multi-agent systems for both task solving and social simulation.
- **Promising Tools for Architectural Design:** Experiments on multi-agent collaboration, resource competition, and layout effects demonstrate that our platform can aid spatial optimisation and resource allocation, making it a promising tool for architectural design work.

2 Related Works

Task Solving Environments for LLM-based Agents evaluate agents' ability to solve various types of tasks, such as web-based tasks (Cai et al., 2024; Chae et al., 2024), GUI tasks (Nguyen et al., 2024; Wang and Liu, 2024), coding tasks (Hong et al., 2024; Huang et al., 2024; Qian et al., 2023) and household tasks (Shridhar et al., 2021, 2020a; Zhang et al., 2024). In household task-solving environments, agents must explore their surroundings, sense and interpret object states, plan and execute actions. The advantage of such environments lies in their support for a diverse range of physical objects and extensive agent-object interactions. For example, ALFWorld (Shridhar et al., 2021) enables agents to interact with objects such as mugs, books, and lamps. Multi-agent platforms like TDW-MAT and VirtualHome (Zhang et al., 2024; Puig et al., 2018) supports interactions with objects like pens, beds, and apples. Similarly, MineLand (Yu et al., 2024) and AdaSociety (Huang et al., 2025), designed for wilderness survival, feature various tools and food items. However, a common limitation across these environments is the lack of explicit modeling for agent heterogeneity. ALFWorld is a single-agent environment, while TDW-MAT and VirtualHome (Puig et al., 2018) features homogeneous agents with identical capabilities. Although inventory variations in AdaSociety and MineLand introduce some level of heterogeneity, the agents remain fundamentally homogeneous, as they share the same action space.

Social Simulation Environments for LLM-based Agents enable agent-agent interactions, elevating the importance of social dynamics. These environments often model differences in personality, profession, and other traits among agents (Park et al., 2023; Wu et al.; Xu et al., 2023; Guan et al., 2025; Li et al., 2024), as well as incorporate human needs modeling (Wang et al., 2023, 2024). However, a major limitation of these environments is the oversimplified modeling of the physical world, including interaction with physical objects. For instance, an agent may perform the action of eating, without any explicit modeling of food items in the environment. This prevents these simulation environments to be applied in settings where the physical environment has impacts on agent behaviors, such as resource allocation and layout study.

Abstract Modeling with Text-based Environments Prior works such as TextWorld (Côté et al., 2018) and ALFWorld (Shridhar et al., 2020b) have explored the use of text-based environments for abstract modeling of visual and physical dynamics. These approaches are inspired by concepts from inverse graphics and inverse dynamics (Kulkarni et al., 2015; Wu et al., 2017), where high-level representations allow agents to reason about the environment and predict future outcomes. Building on this line of research, our work aligns with ALFWorld (Shridhar et al., 2020b) in treating text-based simulation as a platform for abstract modeling, while further extending this perspective toward heterogeneous and socially grounded agent interactions.

AI in Architectural Design has been widely adopted, mainly to generate static 2D and 3D artefacts, such as floorplans, interiors, and furniture layouts, but these visually oriented methods provide little insight into the dynamic occupant activities (Li et al., 2025; Raistrick et al., 2024; Leng et al., 2023). Moreover, traditional occupant-simulation tools rely on rule-based or state-based behavior transitions (Schaumann et al., 2017; Lee

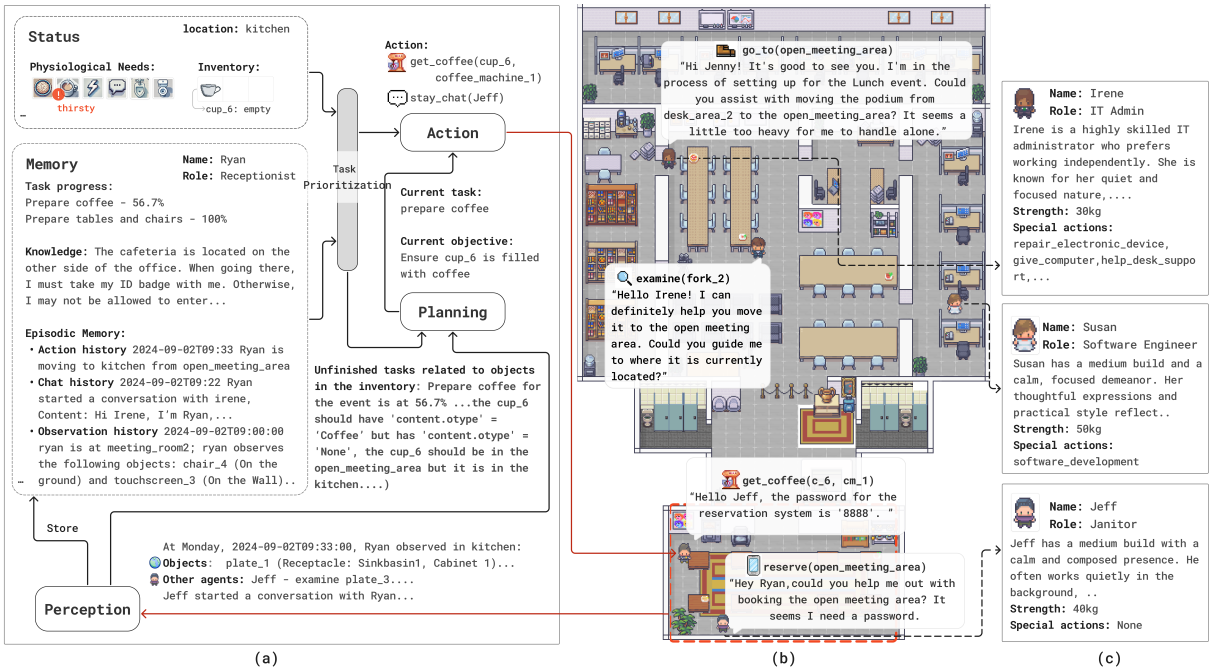


Figure 1: INDOORWORLD system. (a) Agent architecture; (b) Example agent behaviors; (c) Example of heterogeneous agent profiles

et al., 2021; Liu et al., 2024), offering far less flexibility and realism than LLMs in responding to changing environments. Our work addresses this gap by employing LLMs to simulate multi-agent interactions within indoor spaces, yielding insights into occupant behaviors that are crucial for effective spatial planning and resource management.

3 INDOORWORLD Environment

Our environment INDOORWORLD seamlessly integrates task-solving and social simulation in an indoor space setting, offering a versatile testbed to study the interplay between social and physical agent behavior. INDOORWORLD enables agents with rich individual differences to collaborate on tasks with complex hierarchy, involving self-regulated labor division, prioritization and coordination, while satisfying physiological needs through environmental interactions. By incorporating fine-grained modeling of agent internal structure and decision-making process, INDOORWORLD more accurately resembles real-world human behaviors. Tables 1, 5, and 7 summarize the key differences between INDOORWORLD and existing platforms, highlighting its potential to advance research in LLM-based multi-agent systems.

3.1 Environment Architecture

Framework Overview and Core Components Virtual environments that support planning and task solving often require managing the world state (Srivastava et al., 2021; Shridhar et al., 2021). Similarly, our INDOORWORLD adopts an object-oriented approach to define state transition systems through three key components: 1) **Agents**, 2) **Objects**, and 3) **Locations**.

Each agent and object is associated with state variables, such as a numerical value for an agent's hunger or a Boolean indicating whether a computer is broken. A dedicated variable tracks each entity's current location. The overall **world state** is the joint valuation of all these variables. Object affordances are defined by the set of actions that can be performed on them, and these actions are further constrained by the agent's role (e.g., only an IT admin can repair computers). When actions are executed, the world state updates accordingly. Objects can be marked as *receptacles* to store other objects. Locations can be interconnected to allow agent and object movement.

Agent interactions are through conversations. INDOORWORLD supports 4 actions related to conversation with other agents: 1) `initiating_chat`; 2) `stay_chat`; 3) `end_chat`, and 4) `join_chat`. We let the LLMs to generate free-form dialog content. Note that we allow any number of agents to

	MA	AT	OI	TE	LS	FH	HN	RF
ALFworld (Shridhar et al., 2021)	✗	–	✓	✓	✗	✗	✗	✓
Virtual-Home (Puig et al., 2018)	✓	Ho.	✓	✓	✓	✗	✗	✓
TDW-MAT (Zhang et al., 2024)	✓	Ho.	✓	✓	✗	✗	✗	✓
C-WAH (Zhang et al., 2024)	✓	Ho.	✓	✓	✗	✗	✗	✓
MineLand (Yu et al., 2024)	✓	Ho.	✓	✓	✓	✗	✓	✗
AdaSociety (Huang et al., 2025)	✓	Ho.	✓	✓	✓	✗	✓	✗
Smallville (Park et al., 2023)	✓	He.	Lmtd.	✗	✓	✗	✗	✓
Humanoid Agents (Wang et al., 2023)	✓	He.	Lmtd.	✗	✓	✗	✓	✓
Ours	✓	**He.**	✓	✓	✓	✓	✓	✓

Table 1: Comparison of environments. MA: Multi-Agents, AT: Agent Type, OI: Object Interaction, TE: Task Eval, LS: Life Simulation, FH: Fine-Grained Heterogeneity, HN: Human Needs, RF: Real-world Fit. "Ho." = Homogeneous, "He." = Heterogeneous, "Lmtd." = Limited.

be in a conversation, as long as they are at the same location. A dedicated agent state variable indicates which conversation session the agent is currently involved (if any). Action 2) and 3) only become admissible actions when the agent is in a conversation, and 4) becomes admissible when there is an ongoing conversation session at the agent's current location. Each agent can only be at one conversation session at a time. Conversations can be used to share information (including task progress), discuss labor division, and coordinating actions. They affect the agent's subsequent actions by updating the agent's internal state. Note that in task-solving scenarios, conversations are **utility-driven**—the dialog content need to serve task-solving. This aspect distinguishes our work from most existing environments featuring agent conversations.

Sessions and Scenarios INDOORWORLD supports both **task solving** and **simulation** sessions. In a task-solving session, a set of tasks are assigned as the shared objective for all agents. The agents need to collectively decide on task orders and assignments, as well as planning for specific action sequences to complete each task. In simulation sessions, no explicit objective is defined.

Both session types start with an initial world state defined by a **scenario**, a JSON configuration file specifying agents, objects, locations, inter-location connections, and receptacle assignments. (See Appendix E.1 for an example JSON file.)

To facilitate experimentation, INDOORWORLD comes with 25 predefined object types (including 7 receptacle types) and 4 predefined agent roles, resulting in a total of 38 action types. The current set of objects and agent roles cover typical activities in an office environment, such as booking meeting rooms, moving desks, cleaning utensils, repairing computers, etc. As shown in Table 7, our environment offers a broader action space compared to many existing text-based and 2D/3D platforms.

Customization and Expansion Although INDOORWORLD currently feature a limited number of object types and agent roles, our object-oriented approach allows easy customization of object type and agent roles. Introducing new object types and agent roles involves defining Python functions specifying new interactions, including preconditions and effects, which can be easily achieved by utilizing existing class hierarchy. We provide example code in Appendix E.2 to illustrate how to introduce new object types, agent roles and interaction type.

3.2 Agent Architecture

Figure 1 (a) illustrates the operational framework of our agents, highlighting their interactions with both the environment and other agents. Our architecture integrates cognitive modules inspired by recent research on LLM-based agents (Yao et al., 2022; Zhang et al., 2024) and consists of five core modules: perception, memory, planning, action, and task prioritization.

Perception: This module processes symbolic observations from the environment, such as nearby objects, receptacles, and other agents' activities, and updates the agent's internal state accordingly.

Memory: The memory module stores agent-specific information, long-term knowledge, and interaction history. Inspired by the COELA model (Zhang et al., 2024), it maintains a semantic map and tracks task progress, while also recording episodic events (e.g., past actions and conversations) and retaining pre-existing knowledge. It further monitors internal states like physiological needs and inventory status.

Planning: Using current observations and stored memories, the planning module determines the agent's current objective and task to address both task-specific and internal needs.

Action: Informed by the ReAct framework (Yao et al., 2022), the action module integrates infor-

mation from perception, memory, and planning to select and execute the next action. This process involves first reasoning about the current situation and evaluating available options before deciding the next move[1].

Task Prioritization: Our preliminary experiments revealed that LLM-based agents, particularly those using open-source models, struggle to maintain focus in multi-task scenarios, frequently switching tasks without completing them (See Sec. 4.3). To address this, we proposed a task prioritization module that encourages agents to concentrate on ongoing tasks. The module monitors the objects an agent holds and their relevance to the current task, reminding the agent of incomplete objects. For example, as shown in Figure 1, when agent Ryan is working on preparing coffee and is carrying an empty `cup_6`, the module highlights its incomplete status, such as the absence of coffee or its incorrect placement. When no active task is detected, the module reminds the agent about all unfinished tasks, guiding the agent to select an objective aligned with its role and skills. This approach promotes concentration on ongoing tasks and minimizes inefficient task switching. Note that the module does not introduce extra information but selectively reiterates relevant parts of the agent's memory, such as task progress, to reinforce task awareness.

Modeling Agent Heterogeneity INDOORWORLD addresses the limited diversity found in existing multi-agent benchmarks (Table 1 and 5) by assigning every agent a *multi-level profile*. Each profile combines a *role* that determines the agent's unique action space (e.g. only IT administrators can `repair` devices) with additional attributes such as *personality*, *strength*, *skill*, and *knowledge*. This layered design (illustrated in Fig. 1c) supports emergent division of labour and coordinated behaviors while greatly increasing realism.

We model agent heterogeneity at **four levels**: (i) *profile level*, capturing differences in personas and role configurations; (ii) *action space*, where roles have distinct actions; (e.g., only IT admins can `repair` devices); (iii) *capability*, meaning agents may perform the same action with different efficiency or outcomes, such as janitors cleaning more quickly or strong agents being able to move heavy objects; and (iv) *knowledge*, whereby agents hold different internal information (e.g., only receptionists know how to book a meeting room).

To validate this design, 20 practicing architects rated whether heterogeneity at each level increases realism and whether it is important for understanding real space use (details in Appendix B). The consistently high realism scores (55–70 %) and substantial importance ratings (55–95 %) demonstrate that multi-level heterogeneity is both credible and valuable for architectural analysis, thereby supporting the soundness of our design choices.

Modeling Human Needs In INDOORWORLD , agents are associated with physiological and social needs, such as hunger, thirst, and social interaction, that resemble actual building occupants (Figure 1). They are tracked with numerical state values that gradually decline over time, prompting agents to perform restorative actions like eating, drinking, socializing, or using the restroom. This explicit modeling fosters more natural and realistic behaviors as agents balance personal upkeep with assigned tasks.

4 Experiments

4.1 A Collaborative Task Solving Benchmark

To demonstrate how INDOORWORLD supports task solving with social interactions, we design a benchmark for collaborative task solving in an office setting. The benchmark consists of an overarching task: preparing for a company event, composed of five major subtasks, each involving common office activities such as transporting items, cleaning utensils, repairing equipment, booking meeting spaces, and preparing food and beverages. These subtasks can be broken down into smaller steps that require coordination among agents. Unlike many existing environments where agents execute a single task in isolation (Shridhar et al., 2021), in our setting, all agents simultaneously receive the complete task structure, but must autonomously determine how to divide responsibilities, prioritize actions, and coordinate efforts in a decentralized manner. This introduces additional challenges related to task prioritization, division of labor, communication, and coordination strategies. Each scenario runs for one hour of simulation time, during which agents must collaboratively complete as much of the task as possible. The scenarios feature 9 locations, 67 objects across 16 types (including 15 receptacles of 7

[1]The reasoning part is omitted in Figure 1 due to space limitations.

types), and 6 agents: four janitors, one IT administrator, and one receptionist. Details of each subtask and its decomposition can be found in Appendix A.

4.2 Social Simulation Experiments

Like Smallville (Park et al., 2023), our environment supports autonomous social simulation where agents generate their own objectives based on physiological/social needs, personality, and roles, driving interactions with both the environment and other agents.

As INDOORWORLD simulates both social and physical interactions, one potential real-world application is to evaluate how well a physical environment can facilitate the activities of its occupants. We demonstrate this application with two experiments, focusing on resource management and spatial layout design, respectively.

Resource Management experiment showcases how we can use INDOORWORLD to evaluate resource allocation in an environment. The scenario involves agents residing in a space with limited resource. We examine how agents compete for and utilize resources under different conditions.

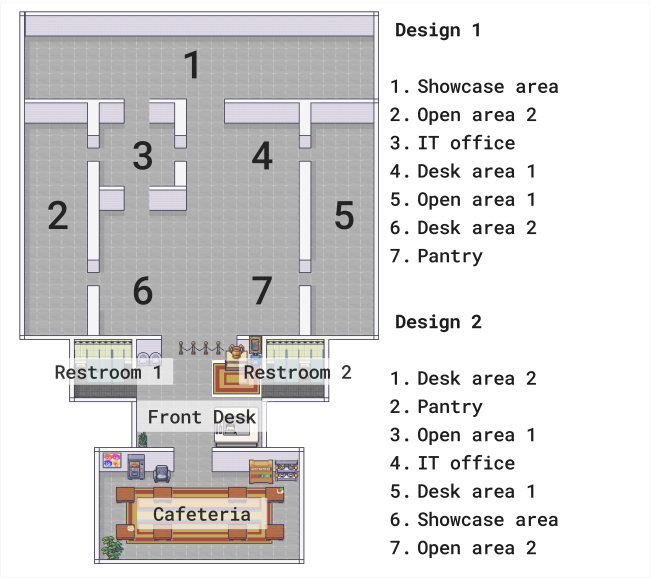
We ran 3 sets of simulations, with 2, 4, and 8 thirsty agents. Each set consist of three settings involving agents all preferring water, all preferring coffee, and no preference, and for each setting we ran simulation with different availability of beverages (one water dispenser/coffee machine vs. two water dispensers/coffee machines).

At the beginning of the simulation, agents experience thirst and autonomously decide whether to drink water or coffee based on their physiological needs and personal preferences. We analyze their resource selection behavior and measure the time required for all agents to fulfill their hydration needs [2]. This allows us to assess how different resource configurations influence competition and overall efficiency in meeting agent needs.

Spatial Layout experiment showcases how we can use INDOORWORLD to evaluate building layout designs. We ran simulations in two scenarios involving two office layout designs (see Figure 2), each lasting for an 8-hour simulation period. Both Design 1 and Design 2 contain the same 11 functional areas to host five agents: one IT administrator, one janitor, two software engineers, and one receptionist. Each agent has a designated workspace, such as the front desk for the receptionist and the IT office for the IT administrator.

[2]Unless otherwise specified, all time measurements in this paper refer to in-simulation time.

Figure 2: Spatial layout of design 1 and design 2



Furthermore, the pantry contains a coffee machine, a water dispenser, food, drinks, cups, and utensils, but with limited resources. The cafeteria, in contrast, is assumed to have no resource constraints, allowing agents to replenish food, drinks, and energy freely. However, activities in the cafeteria takes more time compared to the pantry. Additionally, the environment includes male and female restrooms.

In both Design 1 and Design 2, all areas share the same configurations, including resource availability and the activities they support. The only difference between the two designs lies in the relative positioning of these areas.

Since this experiment does not involve specific task execution, the task prioritization and task progress modules have been removed.

4.3 Results and Analysis

In the collaborative task solving experiment, we evaluated three different LLMs, including the open-source Llama 3.3 70B Instruct, Gemma 3 27B and the proprietary GPT-4o-08-06. For the social simulation experiments, the resource management experiment used GPT-4o-08-06, while the spatial layout experiment used GPT-4o-mini-0718. The temperature for all experiments was set to 0.6.

Collaborative Task Solving Quantitative results are shown in Table 2. We report *instance-level* (IS) task completion rate (percentages of objects that are at the desired state) and *attribute-level* (AS) task

Table 2: Results of collaborative task solving with mean and standard deviation (reported as superscript in small font). T1–T5 represent the five designed tasks. Reported values are IS (Instance-level Success) rate / AS (Attribute-level Success) rate. **FM = Full Model, RA = Random Agents**.

	T1 (IS/AS)	**T2** (IS/AS)	**T3** (IS/AS)	**T4** (IS/AS)	**T5** (IS/AS)	**AVG** (IS/AS)
RA	$8.3^{\pm 14.4}$ / $38.9^{\pm 9.6}$	$0.0^{\pm 0.0}$ / $30.6^{\pm 2.8}$	$0.0^{\pm 0.0}$ / $30.0^{\pm 17.3}$	$0.0^{\pm 0.0}$ / $0.0^{\pm 0.0}$	$0.0^{\pm 0.0}$ / $11.1^{\pm 1.9}$	$1.2^{\pm 2.0}$ / $23.1^{\pm 3.0}$
Llama 3.3 70B Instruct						
FM (w/o S&T+TP)	$83.3^{\pm 28.9}$ / $88.9^{\pm 19.2}$	$38.9^{\pm 21.0}$ / $61.1^{\pm 12.7}$	$0.0^{\pm 0.0}$ / $53.3^{\pm 11.5}$	$100.0^{\pm 0.0}$ / $100.0^{\pm 0.0}$	$0.0^{\pm 0.0}$ / $27.8^{\pm 5.1}$	$31.0^{\pm 9.1}$ / $51.8^{\pm 4.7}$
FM (w/o TP)	$91.7^{\pm 14.4}$ / $94.4^{\pm 9.6}$	$36.1^{\pm 31.5}$ / $59.3^{\pm 22.6}$	$0.0^{\pm 0.0}$ / $53.3^{\pm 11.5}$	$100.0^{\pm 0.0}$ / $100.0^{\pm 0.0}$	$0.0^{\pm 0.0}$ / $41.1^{\pm 9.6}$	$31.0^{\pm 11.9}$ / $56.1^{\pm 11.9}$
FM	$\mathbf{100.0}^{\pm 0.0}$ / $\mathbf{100.0}^{\pm 0.0}$	$\mathbf{58.3}^{\pm 0.0}$ / $\mathbf{72.2}^{\pm 0.0}$	$\mathbf{33.3}^{\pm 57.7}$ / $\mathbf{73.3}^{\pm 23.1}$	$100.0^{\pm 0.0}$ / $100.0^{\pm 0.0}$	$\mathbf{33.3}^{\pm 7.2}$ / $\mathbf{51.1}^{\pm 11.7}$	$\mathbf{55.2}^{\pm 9.1}$ / $\mathbf{67.8}^{\pm 4.8}$
Gemma 3 27B						
FM (w/o S&T+TP)	$75.0^{\pm 25.0}$ / $83.3^{\pm 16.7}$	$11.1^{\pm 4.8}$ / $45.4^{\pm 3.2}$	$33.3^{\pm 57.7}$ / $73.3^{\pm 23.1}$	$100.0^{\pm 0.0}$ / $100.0^{\pm 0.0}$	$12.5^{\pm 21.7}$ / $40.0^{\pm 29.1}$	$26.4^{\pm 10.5}$ / $51.4^{\pm 9.0}$
FM (w/o TP)	$50.0^{\pm 25.0}$ / $66.7^{\pm 16.7}$	$50.0^{\pm 50.0}$ / $70.4^{\pm 28.0}$	$0.0^{\pm 0.0}$ / $60.0^{\pm 0.0}$	$33.3^{\pm 57.7}$ / $33.3^{\pm 57.7}$	$0.0^{\pm 0.0}$ / $30.0^{\pm 12.0}$	$28.7^{\pm 24.2}$ / $53.3^{\pm 17.1}$
FM	$\mathbf{83.3}^{\pm 14.4}$ / $\mathbf{88.9}^{\pm 9.6}$	$\mathbf{72.2}^{\pm 12.7}$ / $\mathbf{81.5}^{\pm 8.5}$	$25.0^{\pm 43.3}$ / $70.0^{\pm 17.3}$	$66.7^{\pm 57.7}$ / $66.7^{\pm 57.7}$	$\mathbf{45.8}^{\pm 38.2}$ / $\mathbf{65.6}^{\pm 26.9}$	$\mathbf{59.8}^{\pm 12.1}$ / $\mathbf{74.5}^{\pm 10.7}$
GPT-4o						
FM (w/o S&T+TP)	$75.0^{\pm 25.0}$ / $83.3^{\pm 16.7}$	$52.8^{\pm 50.2}$ / $77.8^{\pm 21.0}$	$0.0^{\pm 0.0}$ / $50.0^{\pm 17.3}$	$66.7^{\pm 57.7}$ / $66.7^{\pm 57.7}$	$33.3^{\pm 28.9}$ / $58.9^{\pm 19.5}$	$43.7^{\pm 29.3}$ / $67.8^{\pm 14.2}$
FM (w/o TP)	$91.7^{\pm 14.4}$ / $94.4^{\pm 9.6}$	$77.8^{\pm 19.2}$ / $85.2^{\pm 12.8}$	$66.7^{\pm 57.7}$ / $86.7^{\pm 23.1}$	$100.0^{\pm 0.0}$ / $100.0^{\pm 0.0}$	$62.5^{\pm 21.7}$ / $76.7^{\pm 14.5}$	$74.7^{\pm 5.3}$ / $83.5^{\pm 6.2}$
FM	$\mathbf{100.0}^{\pm 0.0}$ / $\mathbf{100.0}^{\pm 0.0}$	$61.1^{\pm 4.8}$ / $74.1^{\pm 3.2}$	$\mathbf{100.0}^{\pm 0.0}$ / $\mathbf{100.0}^{\pm 0.0}$	$100.0^{\pm 0.0}$ / $100.0^{\pm 0.0}$	$\mathbf{83.3}^{\pm 14.4}$ / $\mathbf{87.8}^{\pm 10.7}$	$\mathbf{79.3}^{\pm 3.5}$ / $\mathbf{84.7}^{\pm 3.5}$

Table 3: Facility Resource Stress Testing. The second row (1/1, 2/2) indicates the number of water dispensers and coffee machines. The left column (2, 4, 8) represents the number of agents. Each cell (X/Y/Z) shows the number of agents who drank water (X) and coffee (Y), and the total time (Z) for all agents to hydrate.

	All Like Water		**All Like Coffee**		**No Preference**	
	1/1	2/2	1/1	2/2	1/1	2/2
2	2/0/4	2/0/3	0/2/4	0/2/3	2/0/4	2/0/3
4	4/0/6	4/0/4	0/4/6	**1/3/4**	4/0/6	4/0/5
8	8/0/10	8/0/7	0/8/10	0/8/7	8/0/12	8/0/7

completion rate (percentages of object state variables that are at the desired value). Removing task prioritization (TP) led to performance degradation across all models, with the most significant drop observed in Llama 3.3 and Gemma 3. This suggests that these models struggle with maintaining focus in multi-task scenarios and benefit substantially from explicit prioritization to reduce inefficient task switching. Further removing the semantic map and task progress tracking (S&T) led to additional performance degradation. GPT-4o exhibited a substantial drop in performance, indicating that it can effectively leverage the additional structured information for task planning and execution. In contrast, the removal of S&T had a more limited impact on Llama 3.3 and Gemma 3, likely due to their weaker information utilization capabilities. These models may struggle to extract key insights from the complex semantic map and task progress data, making it difficult for them to identify and prioritize the most relevant information. This observation suggests that while structured task representations are beneficial, their effectiveness is contingent on the model's ability to process and utilize the provided information efficiently. In addition, our results report standard deviations across three independent runs, which consistently show that the Full Model not only achieves the highest mean performance but also exhibits lower variability. This reduced variance further highlights the robustness of our complete framework with both S&T and TP components.

In the experiments, we observe various collaborative behaviors among the agents. The collaboration and labor division reflects individual differences across agents. For example, Irene (IT Admin) asked Jenny (Janitor) for help moving a podium beyond her strength. (Figure 1 (b)). Collaboration does not only involve physical interaction, but also information sharing via communication. In a different session, Jeff is trying to reserve a meeting room, for which a password is needed. Initially, only the receptionists have knowledge about the password. Jeff first attempted a random password (hallucinated by LLMs). After failing, he asked the receptionist Ryan for the password, and successfully booked the meeting room (Figure 1 (b)).

We observe that without task prioritization module, agents tend to switch back and forth between different tasks. For example, Jake is in the kitchen with an uncleaned `fork_3` in his inventory. As a janitor, Jake should clean them as part of Task 2. However, he suddenly remembers that `chair_4` needs to be moved to the open area 1 for Task 1, so he dropped `fork_3` to pick up `chair_4`. But afterwards he recalls the former task again and picks `fork_3` again. The behavior repeated and resulted in Jake not achieving any goals.

Admittedly, even for models with a high completion rate, tasks may not always be completed in

the most efficient way. For example, a receptionist may choose to clean the dishes themselves instead of asking for help from the janitor, who can clean dishes much faster. Lack of global coordination also sometimes leads to duplicated task completion by agents in different locations. For instance, Irene was unaware that her colleagues had already prepared enough tea in the open area 1, so she continued preparing tea in the kitchen and attempted to carry it over. Optimizing collective task completion efficiency remains a challenge.

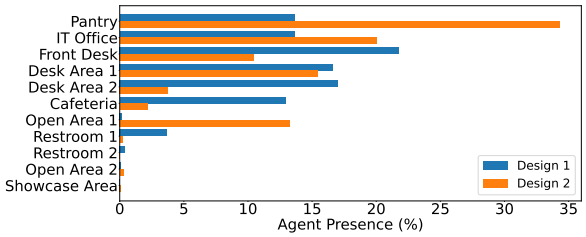


Figure 3: Agent Presence Across Locations.

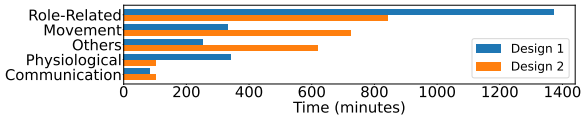


Figure 4: Activity Time Distribution.

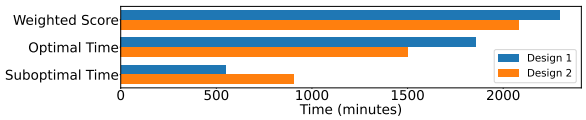


Figure 5: Agent Well-being Metrics.

Resource Management Results are shown in Table 3. We observed that increasing the number of agents led to longer hydration times due to increased competition and queuing for resources. However, when the number of available water dispensers and coffee machines was increased, agents were able to hydrate more quickly, demonstrating that more resources can alleviate competition and improve efficiency.

We also found that agent preferences significantly influenced their resource selection. When all agents preferred water, they consistently chose the water dispenser, even when a coffee machine was available. Similarly, when all agents preferred coffee, they almost exclusively used the coffee machine, ignoring the water dispenser. In the no-preference scenario, all agents opted for the water dispenser. This may suggest that in the absence of a strong preference, LLM-based agents default to an prior knowledge that biased towards water as the primary hydration method. This may also explain why, even when all agents preferred coffee, a small number of them still chose the water dispenser.

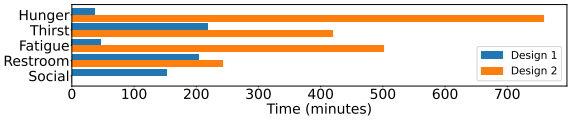


Figure 6: Suboptimal State Distribution.

As can be seen, simulation in INDOORWORLD can effectively reflect the impact of different resource allocation strategies, making it potentially a useful tool to aid real-world decision making on resource allocation.

Spatial Layout Results illustrates the impact of different spatial layouts on agent behavior, focusing on location usage, activity time allocation, and overall well-being. The results indicate that spatial design significantly influences resource accessibility, social interactions, and agent efficiency and well-being.

As shown in Figure 3, Design 1 maintained a more balanced distribution of agents across locations, while Design 2 saw a higher concentration in resource-rich areas, such as the pantry, leading to reduced workspace utilization, particularly for Desk Area 2, which is positioned in the farthest corner. This suggests that inefficient resource distribution may cause agents to move away from their designated workspaces, opting instead to work or engage in activities elsewhere.

Figure 4 illustrates that agents in Design 2 spent more time moving and less time on role-related work due to longer distances from their workspace to resource-rich areas. Social interaction time was also higher, while time spent addressing physiological needs was lower. According to the simulation log, in Design 2, agents spent a significant portion of their time in the pantry engaging in conversations. As a result, despite their prolonged stay in the pantry, the time spent on physiological needs-related activities remained relatively low.

Figure 5 and Figure 6 further show that agents in Design 2 spent less time in an optimal state and more time experiencing unmet needs, particularly hunger and fatigue. This can be attributed to the limited resources available in the pantry, such as only two pieces of bread, two apples and a few clean cups, which were quickly consumed.

Implications for Architectural Design To evaluate how INDOORWORLD can support professional practice, we surveyed 9 architects and summarized their feedback in Table 6. The results indicate that our features (human need modelling, resource management, and spatial layout experiments) align closely with the considerations architects make during design, and the spatial layout experiments were rated as especially helpful for understanding occupant behavior. Results of the survey and discussions are provided in Appendix C.

5 Conclusion

This study introduces INDOORWORLD , a multi-agent simulation environment that integrates fine-grained heterogeneous agent modeling with physical interactions. INDOORWORLD enables agents with varied abilities to coordinate roles, satisfy physiological needs, and interact with a rich array of objects within realistic settings.

We evaluate three LLMs in collaborative task solving using a benchmark of common office tasks that involve both social and physical interactions, highlighting the effectiveness of task prioritization. Our simulation experiments on resource management and spatial layouts demonstrate the potential real-world applications of INDOORWORLD in architectural design.

Limitations

Impact of Experimental Variability on Conclusions Due to the inherent randomness of LLMs, LLM-based agents may make different decisions even when facing identical scenarios. Furthermore, since our environment involves multiple agents making sequential decisions across multiple rounds, and their choices influence one another, the results may vary even when using the same experimental settings and LLM model. In our experiments, we observed that fluctuations in results were often associated with the following phenomena: (a) Agents engaging in prolonged conversations without progressing on task completion. (b) Agents performing incorrect actions after making substantial progress, leading to task failure. For instance, in one run of the GPT-4o Full model without S&T and TP (see Table 2), an agent unexpectedly moved a table, already placed in the open area 1, back to the kitchen. As a result, both the table and the clean utensils on it were relocated to an incorrect position, significantly lowering the overall task success rate.

To mitigate the impact of experimental variability on the reliability of conclusions, we conducted three independent runs for each task-solving experiment. Additionally, we commit to publicly releasing our code after acceptance to enable further reproducibility and facilitate research on LLM-based agents' behavior, reasoning capabilities, and collaborative problem-solving.

Scalability, Customization, and Future Improvements Although we have designed our environment to be easily scalable and customizable, the attributes of objects and the interaction mechanisms between objects and agents currently rely on user-defined configurations. This customization process is subject to user preferences and research objectives. In the future, we plan to follow the approach outlined in (Srivastava et al., 2021) by leveraging WordNet (Miller) to automatically generate object attributes and interaction methods. This will enable the rapid expansion of the environment's object library while allowing users to further tailor interactions through direct modifications in the Python code.

Our current environment adopts a text-based game engine where all physical and visual interactions are abstracted. Moving forward, we aim to extend our framework to support 3D assets and physical simulation engines, enhancing the realism of agent-environment interactions. This expansion will allow researchers to flexibly utilize our environment in both abstract and more concrete settings, depending on their experimental needs.

Current State of LLM-based Agents The practical utility of INDOORWORLD is currently constrained by the capabilities and costs of large language models. In our survey, architects acknowledged that they could obtain valuable insights from the interactions of LLM agents with the environment in INDOORWORLD , but these agents still fall short in approximating real human behavior. In addition, their deployment remains relatively expensive for large-scale, long-duration studies. Nevertheless, we are optimistic that as LLMs continue to improve, their behavioral outputs will become increasingly human-like, and that reductions in model costs will further enhance their usability, thereby strengthening the realism and applicability of our proposed framework.

Testing of Reasoning Models Due to time and computational constraints, we did not test reasoning models such as GPT-O1 or Deepseek R1 (DeepSeek-AI et al., 2025) in our experiments. Consequently, some unsolved cases in our study may be successfully addressed by more advanced reasoning models, potentially leading to agent behaviors that better resemble human decision-making.

Acknowledgements

This work was initially developed during Dekun Wu's internship at Autodesk Research. We thank Autodesk Research for their support and resources. This academic work is also supported in part by the Canada CIFAR AI Chair Program and the Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery Grant (RGPIN-2021-03115). We also acknowledge the use of GitHub Copilot Enterprise for low-novelty tasks such as code auto-completion and suggesting Python class definitions, which helped accelerate the coding process.

References

Hongru Cai, Yongqi Li, Wenjie Wang, Fengbin Zhu, Xiaoyu Shen, Wenjie Li, and Tat-Seng Chua. 2024. Large language models empowered personalized web agents. *Preprint*, arXiv:2410.17236.

Hyungjoo Chae, Namyoung Kim, Kai Tzu iunn Ong, Minju Gwak, Gwanwoo Song, Jihoon Kim, Sunghwan Kim, Dongha Lee, and Jinyoung Yeo. 2024. Web agents with world models: Learning and leveraging environment dynamics in web navigation. *Preprint*, arXiv:2410.13232.

Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, Wendy Tay, and Adam Trischler. 2018. Textworld: A learning environment for text-based games. In *Computer Games Workshop at ICML/IJCAI 2018*, pages 1–29.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Xiaohang Feng, Da Yan, and Tianzhen Hong. 2015. Simulation of occupancy in buildings. *Energy and Buildings*, 87:348–359.

Chuang Gan, Jeremy Schwartz, Seth Alter, Damian Mrowca, Martin Schrimpf, James Traer, Julian De Freitas, Jonas Kubilius, Abhishek Bhandwaldar, Nick Haber, Megumi Sano, Kuno Kim, Elias Wang, Michael Lingelbach, Aidan Curtis, Kevin Feigelis, Daniel M. Bear, Dan Gutfreund, David Cox, Antonio Torralba, James J. DiCarlo, Joshua B. Tenenbaum, Josh H. McDermott, and Daniel L. K. Yamins. 2021. Threedworld: A platform for interactive multi-modal physical simulation. *Preprint*, arXiv:2007.04954.

Zhenyu Guan, Xiangyu Kong, Fangwei Zhong, and Yizhou Wang. 2025. Richelieu: Self-evolving llm-based agents for ai diplomacy. *Advances in Neural Information Processing Systems*, 37:123471–123497.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang

Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2024. MetaGPT: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*.

Dong Huang, Jie M. Zhang, Michael Luck, Qingwen Bu, Yuhao Qing, and Heming Cui. 2024. Agentcoder: Multi-agent-based code generation with iterative testing and optimisation. *Preprint*, arXiv:2312.13010.

Yizhe Huang, Xingbo Wang, Hao Liu, Fanqi Kong, Aoyang Qin, Min Tang, Song-Chun Zhu, Mingjie Bi, Siyuan Qi, and Xue Feng. 2025. Adasociety: An adaptive environment with social structures for multi-agent decision-making. *Preprint*, arXiv:2411.03865.

Tejas D Kulkarni, William F. Whitney, Pushmeet Kohli, and Josh Tenenbaum. 2015. Deep convolutional inverse graphics network. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Bokyung Lee, Michael Lee, Jeremy Mogk, Rhys Goldstein, Jacobo Bibliowicz, Frederik Brudy, and Alexander Tessier. 2021. Designing a multi-agent occupant simulation system to support facility planning and analysis for covid-19. In *Proceedings of the 2021 ACM Designing Interactive Systems Conference*.

Sicong Leng, Yang Zhou, Mohammed Haroon Dupty, Wee Sun Lee, Sam Joyce, and Wei Lu. 2023. Tell2Design: A dataset for language-guided floor plan generation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Chengyuan Li, Tianyu Zhang, Xusheng Du, Ye Zhang, and Haoran Xie. 2025. Generative ai models for different steps in architectural design: A literature review. *Frontiers of Architectural Research*.

Nian Li, Chen Gao, Mingyu Li, Yong Li, and Qingmin Liao. 2024. Econagent: large language model-empowered agents for simulating macroeconomic activities. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15523–15536.

Yuanyuan Liu, Ying Zhou, Le Yang, and Yangpeng Xin. 2024. Simulating staff activities in healthcare environments: An empirical multi-agent modeling approach. *Journal of Building Engineering*.

Pattie Maes. 1995. Artificial life meets entertainment: lifelike autonomous agents. *Communications of the ACM*, 38(11):108–114.

George A. Miller. Wordnet: a lexical database for english. *Commun. ACM*.

Dang Nguyen, Jian Chen, Yu Wang, Gang Wu, Namyong Park, Zhengmian Hu, Hanjia Lyu, Junda Wu, Ryan Aponte, Yu Xia, Xintong Li, Jing Shi, Hongjie Chen, Viet Dac Lai, Zhouhang Xie, Sungchul Kim, Ruiyi Zhang, Tong Yu, Mehrab Tanjim, Nesreen K. Ahmed, Puneet Mathur, Seunghyun Yoon, Lina Yao, Branislav Kveton, Thien Huu Nguyen, Trung Bui, Tianyi Zhou, Ryan A. Rossi, and Franck Dernoncourt. 2024. Gui agents: A survey. *Preprint*, arXiv:2412.13501.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *In the 36th Annual ACM Symposium on User Interface Software and Technology (UIST '23)*, UIST '23, New York, NY, USA. Association for Computing Machinery.

Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. 2018. Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8494–8502.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2023. Chatdev: Communicative agents for software development. *arXiv preprint arXiv:2307.07924*.

Alexander Raistrick, Lingjie Mei, Karhan Kayan, David Yan, Yiming Zuo, Beining Han, Hongyu Wen, Meenal Parakh, Stamatis Alexandropoulos, Lahav Lipson, Zeyu Ma, and Jia Deng. 2024. Infinigen indoors: Photorealistic indoor scenes using procedural generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21783–21794.

Davide Schaumann, Simon Breslav, Rhys Goldstein, Azam Khan, and Yehuda E Kalay. 2017. Simulating use scenarios in hospitals using multi-agent narratives. *Journal of Building Performance Simulation*, 10(5-6):636–652.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2024. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. 2020a. ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2020b. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2021. ALFWorld: Aligning Text and Embodied Environments for Interactive Learning.

In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Sanjana Srivastava, Chengshu Li, Michael Lingelbach, Roberto Martín-Martín, Fei Xia, Kent Vainio, Zheng Lian, Cem Gokmen, Shyamal Buch, C. Karen Liu, Silvio Savarese, Hyowon Gweon, Jiajun Wu, and Li Fei-Fei. 2021. Behavior: Benchmark for everyday household activities in virtual, interactive, and ecological environments. *Preprint*, arXiv:2108.03332.

Xiangru Tang, Anni Zou, Zhuosheng Zhang, Ziming Li, Yilun Zhao, Xingyao Zhang, Arman Cohan, and Mark Gerstein. 2024. MedAgents: Large language models as collaborators for zero-shot medical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 599–621, Bangkok, Thailand.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Ji-Rong Wen. A Survey on Large Language Model based Autonomous Agents. *Preprint*, arXiv:2308.11432.

Xiaoqiang Wang and Bang Liu. 2024. Oscar: Operating system control via state-aware reasoning and re-planning. *arXiv preprint arXiv:2410.18963*.

Yiding Wang, Yuxuan Chen, Fangwei Zhong, Long Ma, and Yizhou Wang. 2024. Simulating human-like daily activities with desire-driven autonomy. *arXiv preprint arXiv:2412.06435*.

Zhilin Wang, Yu Ying Chiu, and Yu Cheung Chiu. 2023. Humanoid agents: Platform for simulating human-like generative agents. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 167–176, Singapore. Association for Computational Linguistics.

Dekun Wu, Haochen Shi, Zhiyuan Sun, and Bang Liu. Deciphering digital detectives: Understanding LLM behaviors and capabilities in multi-agent mystery games. In *Findings of the Association for Computational Linguistics: ACL 2024*.

Jiajun Wu, Erika Lu, Pushmeet Kohli, Bill Freeman, and Josh Tenenbaum. 2017. Learning to see physics via visual de-animation. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Yuzhuang Xu, Shuo Wang, Peng Li, Fuwen Luo, Xiaolong Wang, Weidong Liu, and Yang Liu. 2023. Exploring large language models for communication games: An empirical study on werewolf. *arXiv preprint arXiv:2309.04658*.

Da Yan, William O'Brien, Tianzhen Hong, Xiaohang Feng, H Burak Gunay, Farhang Tahmasebi, and Ardeshir Mahdavi. 2015. Occupant behavior modeling for building performance simulation: Current state and future challenges. *Energy and buildings*, 107:264–278.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Xianhao Yu, Jiaqi Fu, Renjia Deng, and Wenjuan Han. 2024. Mineland: Simulating large-scale multi-agent interactions with limited multimodal senses and physical needs. *Preprint*, arXiv:2403.19267.

Hongxin Zhang, Weihua Du, Jiaming Shan, Qinhong Zhou, Yilun Du, Joshua B Tenenbaum, Tianmin Shu, and Chuang Gan. 2024. Building cooperative embodied agents modularly with large language models. *ICLR*.

A Task Coordination Details

In Table 4, we list the specific task descriptions used in our collaborative task-solving experiment. During the experiment, all five tasks were simultaneously input into the memory of all agents. Due to agent heterogeneity, certain task components require specific agents to complete or can be completed more efficiently by particular agents. Therefore, agents must determine which part of the task they should be responsible for and then take action simultaneously.

Hierarchically, the highest-level task is preparing the company event, under which the five tasks listed in 12 serve as sub-tasks. Each task description further contains low-level tasks, such as the task Prepare Enough Tables and Chairs in the Event Area, which includes sub-tasks like Move 2 tables and 2 chairs to the open area 1 from other locations. These sub-tasks, in turn, may require multiple steps to complete.

This multi-level hierarchical task structure evaluates not only the agents' planning abilities but also their collaboration and communication skills.

B Architect Survey on Multi–level Heterogeneity

Table 5 compares the modeling of fine-grained heterogeneity types across four heterogeneous multi-agent environments. Our approach is the only one that comprehensively supports all four types of heterogeneity: profile, action, capability, and knowledge. To validate the necessity of modeling these levels of heterogeneity, we conducted a survey with 20 architects, asking for their views on the realism, importance, and consideration of heterogeneity modeling.

Participants and Survey Design Twenty practicing architects with at least 3 years' experience were recruited through the external platform `UserTesting.com`. Compensation levels were set in accordance with internal company guidelines, and in all cases participants were remunerated above the minimum wage in their regions. Each participant signed a consent form that explained the study purpose and data use and before completing an online questionnaire. For each of the four heterogeneity levels introduced in Section 3.2, respondents (i) compared two design options, *A*: fully homogeneous occupants, and *B*: heterogeneous occupants, and indicated which looked *more realistic*; (ii) evaluated how *important* the heterogeneity was for understanding real space use (5-point Likert scale); (iii) reported whether they normally consider that difference in their own design work, and, if not, *why*. This study was reviewed by and received approval through Autodesk's internal ethics review process.

Results Overview Figure 7 shows that heterogeneity was judged more realistic by 55–70 % of architects depending on the level, with the highest endorsement for *profile*, *action* and *capability* heterogeneity. Importance ratings (Fig. 8) follow a similar trend: 55–95 % of respondents marked the heterogeneity as "important" or "very important". Figure 9 shows that most architects already incorporate heterogeneity consideration in their own workflow: all 20 architects consider profile-level differences, and large majorities do so for action (90 %), capability (60 %), and knowledge (65 %) heterogeneity. This strong uptake suggests that heterogeneous agent modeling is both familiar and valued in practice, particularly at the profile and action levels. However, some architects still do not consider these differences in their design work, citing various reasons. Fig. 10 presents the further analysis of these reasons. For action-level heterogeneity, the two architects who do not consider it pointed to external factors: one mentioned that *clients or stakeholders did not require this level of detail*, while the other cited *having to focus on more important constraints*. At the capability level, the reasons were more diverse. While 25 % of those who opted out mentioned stakeholder requirements, another 25 % indicated that *this type of variation is usually ignored in current design workflows*. An additional 25 % referenced focusing on more pressing design aspects, while the remaining 25 % were divided between *not being aware this kind of modeling was possible* and *lacking the tools or data to support it*. At the knowledge level, the primary reason for non-consideration was a perceived lack of importance. Other respondents mentioned stakeholder scope (29 %), and a smaller portion cited workflow limitations or practical constraints.

Implications This architect survey indicates that our multi-level heterogeneity yields more realistic simulations than existing environments and aligns more closely with the factors architects consider during design. This feature not only justifies the design choices behind INDOORWORLD , but also strengthens its potential as a supportive tool for

Table 4: Task Descriptions

Task Number	Task Name	Description
1	Prepare Enough Tables and Chairs in the Event Area	Move 2 tables and 2 chairs to the open area 1 from other locations.
2	Prepare Clean Utensils for the Event	Transport 4 clean plates, 4 clean knives, and 4 clean forks from other locations to the open area 1 and place them on the tables.
3	Check and Repair Broken Computer, Projector, and Microphone	Move a podium to the open area 1 if there is no podium there. Bring one computer, one projector, and one microphone to the open area 1 and place them on the podium. Ensure they are in working condition.
4	Book a Meeting Room for the Event	Use the touch screen in the open area 1 or a computer to remotely reserve the open area 1. Ensure the meeting room is booked with the following details: Event Name: Lunch and Listen; Start time: 2024-09-02T12:00:00; End time: 2024-09-02T13:00:00.
5	Prepare Coffee, Tea, and Lunch for the Event	Take clean cups and use them to prepare 3 cups of coffee and 3 cups of tea and place them on the tables in the open area 1. Bring 2 meals to the open area 1 and place them on the tables in the open area 1. Ensure that meals are heated.

architectural practice.

C Architect Survey on Design Problem Relevance and Tool Usefulness

Participants and Survey Design Nine practicing architects (3–15 years of experience) were recruited through Autodesk's internal employee mailing lists and snowball sampling to rate ten statements on a five-point Likert scale (1 = Strongly Disagree, 2 = Disagree, 3 = Neutral, 4 = Agree, 5 = Strongly Agree). Participants were compensated with $35 USD for their participation. Each participant provided informed consent through a form that explained the study purpose and data use before completing an online questionnaire. This study was reviewed and approved through Autodesk's internal ethics review process.

The questionnaire comprised two blocks:

1. **Relevance of Design Problems** (q1–q4): This section assessed the perceived importance of key design issues, including resource allocation, spatial layout, human-need considerations, and waiting-time analysis, in everyday architectural practice.

2. **Usefulness of INDOORWORLD** (q5–q10): This section evaluated whether the proposed simulation tool supports architects in understanding and addressing these design problems, as well as improving their design decisions.

Questions and average scores are summarized in Table 6.

Key Findings The survey results indicate strong recognition of the relevance of the design problems addressed. All four items in the first block received high average ratings (M = 3.8–4.7). Notably, the importance of spatial layout (q2, M = 4.7) was overwhelmingly endorsed, reflecting its critical role in influencing occupant behavior and well-being. Similarly, the significance of analyzing waiting times or competition for shared resources

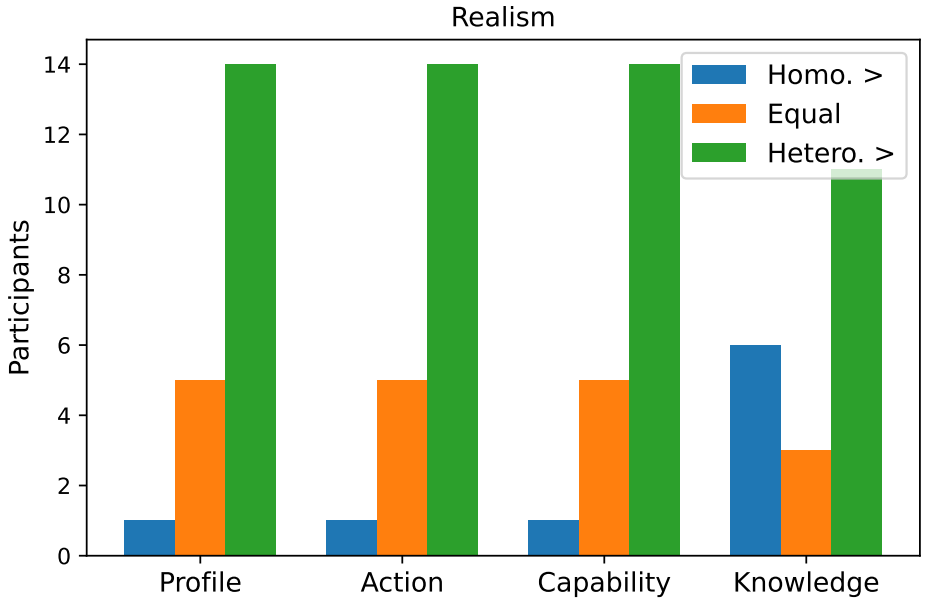


Figure 7: Realism comparison (**Homo.**=homogeneity, **Hetero.**=heterogeneity). A majority consider heterogeneity more realistic than homogeneity at every level.

Environment	Profile	Action	Capability	Knowledge
D2A (Wang et al., 2024)	✓	✗	✗	–
Humanoid Agents (Wang et al., 2023)	✓	✗	✗	–
Smallville (Park et al., 2023)	✓	✗	✗	✓
Ours	✓	✓	✓	✓

Table 5: Comparison of fine-grained heterogeneity types across heterogeneous multi-agent environments. "Yes" indicates support for a type, "–" means not mentioned.

(q4, M = 4.0) was highlighted, affirming the relevance of this problem to practical design.

For tool usefulness, respondents generally expressed positive views regarding the benefits of IndoorWorld. The simulation was appreciated for its ability to reveal layout impacts (q6, M = 4.0), support design refinement processes (q7, M = 3.9), and spark interest for direct use in design projects (q9, M = 4.1). Its potential to bridge the gap between architectural design and user behavior was also well-received (q10, M = 4.2). These responses suggest that architects see IndoorWorld as a promising tool for enhancing their design workflows, offering insights that can inform spatial planning and resource management.

However, two items received only neutral-to-slightly-positive ratings: actionable insights from resource-competition simulation (q5, M = 3.1) and the realism of observed agent behavior (q8, M = 3.0). These lower scores reflect a current gap between the simulation's fidelity and user expectations. Respondents generally acknowledged the concept of resource competition but found "hydration competition" less realistic compared to competition for meeting spaces or other workspace resources, which were viewed as more contextually relevant. One architect expressed concerns that the current game-like interface of IndoorWorld, where agents continuously move around and interact with each other, might deter some architects. They suggested that the simulation could be executed in the background, with results presented as concise, actionable recommendations for spatial planning, minimizing unnecessary visual complexity. Despite these reservations, respondents recognized that the simulation provides useful insights and proposed several enhancements. These included introducing more diverse agent profiles with varying needs and preferences to better assess universal design principles and accessibility, as well as enabling the visualization of agent movement flows to enhance spatial analysis.

Implications Overall, the survey confirms that our modeling choices address key design chal-

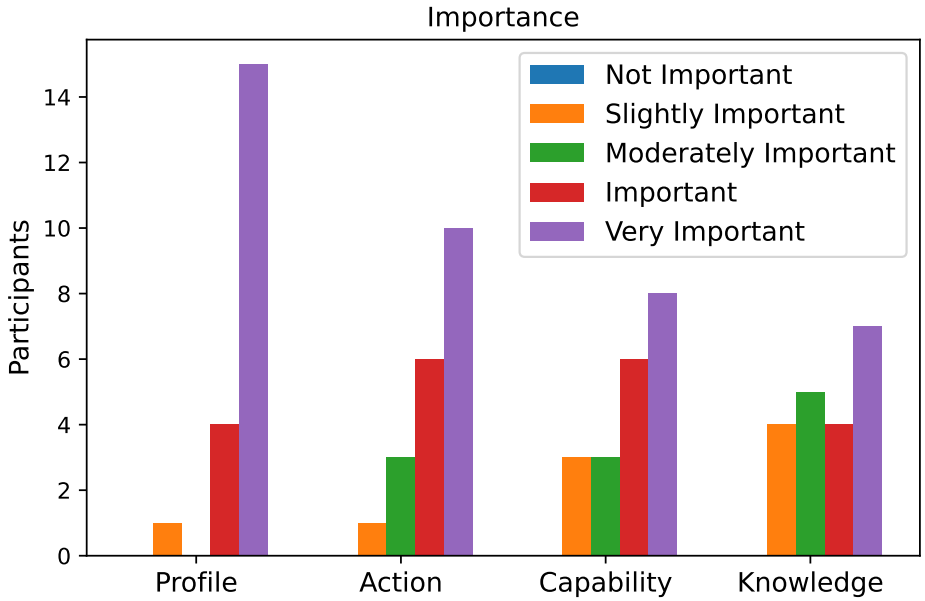


Figure 8: Importance ratings for understanding real space use (5-point scale). Most participants rate heterogeneity as "important" or "very important" at each level.

ID	Question	Avg
Relevance of Design Problems		
q1	Resource allocation is a critical factor in office space planning.	3.8
q2	Spatial layout significantly affects occupant behavior and well-being.	4.7
q3	I consider human needs when designing building layouts.	3.8
q4	Evaluating waiting time or competition for shared resources is useful for improving space efficiency.	4.0
Usefulness of IndoorWorld		
q5	The simulation of resource competition provides actionable insights for real-world resource placement.	3.1
q6	The spatial layout experiment helps reveal how different designs impact occupant behavior and efficiency.	4.0
q7	I can imagine using such simulations to evaluate and refine my own design decisions.	3.9
q8	The observed agent behavior reflects realistic office usage patterns.	3.0
q9	I would be interested in trying out this simulation tool with my own design layouts.	4.1
q10	I believe simulation tools like this can bridge the gap between architectural design and human behavioral modeling.	4.2

Table 6: Average Likert ratings (1 = Strongly Disagree, 2 = Disagree, 3 = Neutral, 4 = Agree, 5 = Strongly Agree) provided by nine practicing architects, summarizing perceived relevance of key design problems and the usefulness of the proposed IndoorWorld simulation tool.

lenges that architects recognize as important. The high scores for spatial layout and resource competition demonstrate that the simulated scenarios align well with real-world concerns. The positive feedback on tool usefulness suggests that IndoorWorld is perceived as a valuable support tool for architectural design, providing insights that can inform spatial planning, resource management, and design optimization.

Moreover, the relatively lower scores on agent behavior realism and the actionability of insights indicate areas for improvement. These results highlight a common challenge for many simulation tools: accurately capturing the complexity of human behavior while ensuring that the generated insights are directly applicable to design decisions. Currently, LLM-based agents are at an early stage of development, with simplified behaviors and limited contextual awareness. We anticipate that ongoing advancements in LLM capabilities will enhance agent realism, enabling more nuanced interactions and generating insights that align more closely with real-world user expectations.

D Action-space comparison

Table 7 summarizes the size of the action space supported by several widely used task-oriented simulation environments. Most benchmarks provide fewer than 20 primitive actions: VirtualHome offers 18, ALFWorld 9, and the TDW variants between 7 and 12. Our environment markedly expands this spectrum to **38 distinct actions**, more

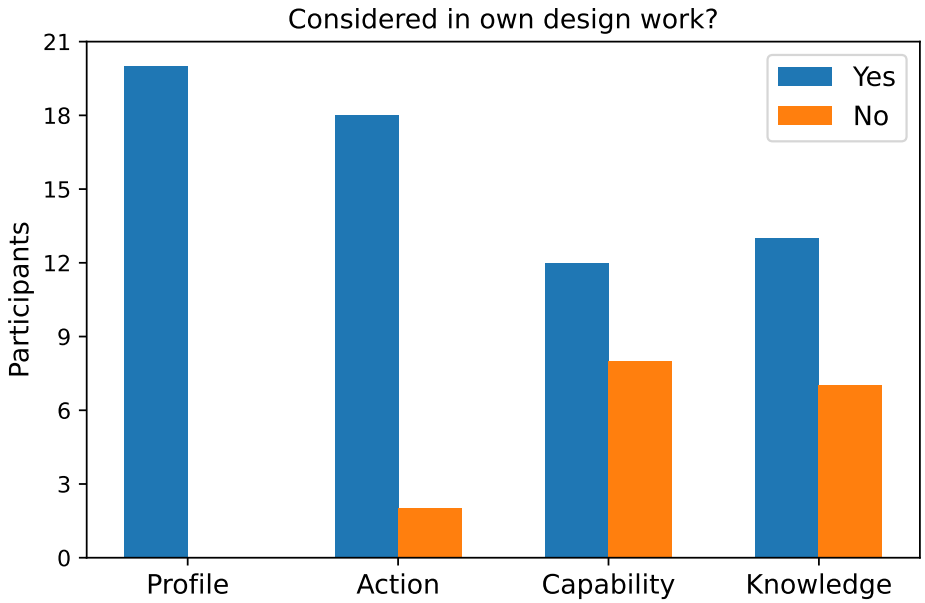


Figure 9: Whether architects typically consider each heterogeneity type in their own work.

	#Actions
Virtual-Home (Puig et al., 2018)	18
ALFWorld (Shridhar et al., 2021)	9
TDW-MAT (Zhang et al., 2024)	7
C-WAH (Zhang et al., 2024)	8
ALFRED (Shridhar et al., 2020a)	13
TDW (Gan et al., 2021)	12
AdaSociety (Huang et al., 2025)	12
Ours	**38**

Table 7: Comparison of the number of actions supported by different environments.

than twice that of any prior system listed. The richer verb set enables agents to engage in a broader range of household manipulations (e.g., `brewCoffee`, `repairDevice`, `heatFood`), which in turn supports more realistic task decompositions, greater behavioral diversity, and nuanced capability differences.

Importantly, the set of 38 actions was designed based on two principles: (1) **Common human actions shared across roles**, such as `go_to`, `take`, and `put`, ensuring coverage of generic physical interactions. We also incorporated basic need-related actions such as eating, drinking, and restroom use, which are absent in ALFWorld (Shridhar et al., 2020b) but are critical for simulating realistic human behavior. (2) **Role-specific actions tailored to each agent type**, such as `repair_electronic_device` for IT admins or `software_development` for software engineers, reflecting distinct occupational responsibilities and supporting our environment's heterogeneity. As further described in Appendix E.2, the action space can be easily extended with new roles and actions, allowing researchers to customize it according to their needs. This flexibility is important for giving researchers greater control over scenario design.

E Resources

We release the INDOORWORLD environment under an MIT license, comprising the codebase, configuration files, experimental setups, and detailed documentation covering installation and usage. These resources are intended to facilitate reproduction of our experiments and enable further extensions by the research community. Interested parties can request access by contacting the first author.

E.1 Environment Initialization via JSON

Our environment is defined using a JSON configuration file, which specifies key components such as agents, locations, inter-location connections, receptacles within locations, and objects. This structured definition allows for flexible customization of the simulation environment.

Our code automatically initializes the environment based on the JSON file, creating corresponding instances of `Location`, `Receptacle`, `Object`, and `Agent` classes. Each element in the JSON file is mapped to its respective class, ensuring that all objects and agents are correctly instantiated with their predefined attributes and relationships.

An example of this JSON configuration is shown

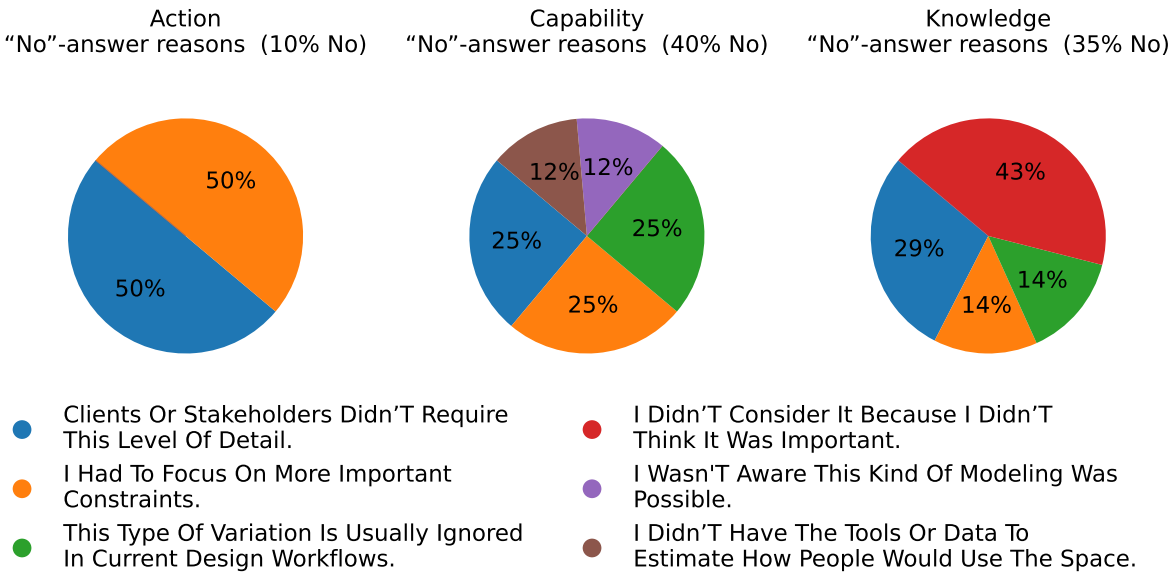


Figure 10: Reasons given by those who answered "No" in Fig. 9.

in Figure 11, where locations, objects, receptacles, and agents are defined. This JSON-based approach enables researchers to easily modify and extend the environment without changing the core simulation code, making it a highly adaptable framework for various research scenarios.

E.2 Extending Agent-Object Interactions

Figures 12, 13, and 14 provide three example classes that illustrate how researchers can extend the environment by adding new object types and by defining interactions between agents and those objects.

For example, if a researcher wants to introduce a new class, such as `Computer`, they first need to specify what actions agents can perform on it, such as `turn_on`. This is done by defining interaction methods like `power_on()` within the `Computer` class. The new class should inherit from `BaseObject` and implement `get_admissible_actions()`, which determines when and how agents can interact with the object. For instance, if the computer is in an accessible state, it can return a command like `turn_on {self.name}`.

On the agent side, new interaction methods must be defined accordingly. For instance, the `Agent` class should implement a `turn_on()` method that handles the interaction logic for powering on a `Computer`. Additionally, in the `act()` function, a new command mapping should be added, linking the `turn_on` command to the `turn_on()` method. This way, when the agent's `get_admissible_actions()` method runs, it will include the new `turn_on` action if the object is in the same location as the agent and is not placed inside a closed receptacle. The agent can then decide whether to execute this action, effectively enabling interaction with the newly introduced `Computer` object.

The `ITAdmin` class further demonstrates how to define role-specific actions. This is done by first adding `"repair_computer"` to the `skills` dictionary. Then, a new method, such as `repair_electronic_device()`, is defined, and the `act()` function maps the `"repair_computer"` command to this method. This allows the `ITAdmin` role to perform repair actions that are unavailable to other agents.

Note: These examples have been simplified for clarity and may not exactly match the original source code. Researchers should refer to the actual implementation details to fully integrate new objects and interactions into the environment.

```
{
  "locations": ["kitchen", "meeting_room1"],
  "location_distances": {
    "kitchen": {"meeting_room1": 1},
    "meeting_room1": {"kitchen": 1}
  },
  "receptacles": [
    {"name": "Sinkbasin1", "location": "kitchen", "rtype": "Sinkbasin",
     "weight_kg": 15, "state": {"fixed": true, "closable": false, "is_open": true,
     "is_clean": true, "temperature": 20, "is_working": true}},
    {"name": "Cabinet1", "location": "kitchen", "rtype": "Cabinet",
     "weight_kg": 30, "state": {"fixed": true, "closable": true, "is_open": false,
     "is_clean": true, "temperature": 20, "is_working": true}}
  ],
  "objects": [
    {"name": "touchscreen_1", "otype": "TouchScreen", "location": "meeting_room1",
     "weight_kg": 3, "state": {"is_turned_on": true, "is_working": true, "is_clean":
          true,
     "temperature": 20}},
    {"name": "cup_1", "otype": "Cup", "location": "kitchen", "receptacle": "
       Countertop1",
     "weight_kg": 0.3, "state": {"is_clean": false, "temperature": 20}}
  ],
  "agents": [
    {
      "name": "ryan",
      "gender": "male",
      "role": "receptionist",
      "location": "meeting_room1",
        "fullness": 100,
        "hydration": 100,
        "energy": 100,
        "social_fulfillment": 100,
      "strength_kg": 65,
      "internal_profile": "Ryan is a professional and welcoming receptionist.
        Known for his friendly personality and exceptional communication skills...."
          ,
      "appearance": "Ryan is a receptionist who is tall and well-built, with ..."
    },
    {
      "name": "irene",
      "gender": "female",
      "role": "IT_admin",
      "location": "kitchen",
        "fullness": 100,
        "hydration": 100,
        "energy": 100,
        "social_fulfillment": 100,
      "strength_kg": 30,
      "internal_profile": "Irene is an organized and skilled IT administrator.
        She is quick to troubleshoot and efficiently repair a wide range of ...",
      "appearance": "Irene has a petite, tidy appearance with a focused expression.
        She is usually dressed casually but professionally, with an attentive ..."
    }
  ]
}
```


Figure 11: Excerpt from the JSON configuration file defining locations, agents, objects, and receptacles in the environment.

```python
class Computer(BaseObject):
    def __init__(self, name, otype='Computer', location, environment, weight_kg,
                 carryable=False, requires_receptacle=True,
                 state={"is_clean": True, 'temperature': 20}):
        super().__init__(name, otype, location, environment, weight_kg,
                         carryable, requires_receptacle, state=state)
    def power_on(self):
        if self.state['is_turned_on']:
            return f"{self.name} is already turned on.", False
        self.state['is_turned_on'] = True
        return f"{self.name} is now turned on.", True

    .....

    def get_admissible_actions(self, agent):
        actions = super().get_admissible_actions(agent)

        # Add repair action
        if f'repair_{self.otype.lower()}' in agent.skills:
            actions.append(f"repair_{self.otype.lower()} {self.name}")

        # Check if the computer has an 'owner' attribute
        if self.location == agent.location:
            if not self.state['is_turned_on']:
                actions.append(f"turn_on {self.name}")
            else:
                actions.append(f"turn_off {self.name}")

        return actions
```

Figure 12: Illustration of the `Computer` class, defining interaction methods like `power_on()` and specifying admissible agent actions.

```python
class Agent:
    def __init__(self, name, role, location, skills=None):
        self.name = name
        self.role = role
        self.location = location
        self.skills = skills if skills else {}

    def act(self, command):
        """Execute an action if it's admissible."""
        command_mapping = {
            "turn_on": (self.turn_on,1), # 1 means one argument
            ...

        }

        tokens = command.split()
        if len(tokens) >= 1:
            action = tokens[0]

            if action in command_mapping:
                action_func, arg_count = command_mapping[action]
                if len(tokens[1:]) >= arg_count:
                    args = tokens[1:1 + arg_count]
                    return action_func(*args)
                else:
                    return f"{self.name} received an incorrect number of arguments for action {action}.", False
            else:
                return f"{self.name} cannot perform action {action}.", False

        return f"{self.name} cannot parse command {command}.", False

    def turn_on(self, electronic_device_name):


        # Search for the electronic device in the current location's objects
        target_device = next((obj for obj in self.location.objects if obj.name == electronic_device_name), None)

        # If the device is found, attempt to turn it on
        if target_device:
            if hasattr(target_device, "turn_on"):
                return target_device.turn_on()
            else:
                return f"{target_device.name} cannot be turned on.", False
        else:
            return f"{self.name} cannot find {electronic_device_name}.", False

    def get_admissible_actions(self):
        admissible_actions = []
        ...
        if self.location.objects:
            for obj in self.location.objects:
                if obj.receptacle == None or obj.receptacle.state['is_open']:
                    admissible_actions.extend(obj.get_admissible_actions(self))
        ...
        return admissible_actions
```

Figure 13: Illustration of the `Agent` class, where `act()` maps commands to interaction methods like `turn_on()`.

```python
class ITAdmin(Agent):
    def __init__(self, name, gender, location, environment, **kwargs):
        super().__init__(name, gender, location, environment, **kwargs)
        # ITAdmin specific skills for repairing devices
        self.skills.update({
            "repair_computer": 1,
            ...
        })

    def repair_electronic_device(self, obj_name):
        # Find the object with the specified name in the current location
        target_device = next((obj for obj in self.location.objects if obj.name ==
             obj_name), None)

        if target_device:
            # Check if the object is a repairable electronic device
            electronic_devices = ["Microphone", "Projector", "Computer", "CoffeeMachine",
                 "WaterDispenser", "Microwave"]
            if target_device.otype in electronic_devices:
                # Check if the device is broken
                if "is_working" in target_device.state and not target_device.state["
                     is_working"]:
                    target_device.state["is_working"] = True
                    return f"{self.name} repaired the {target_device.name}.", True
                else:
                    return f"{target_device.name} is already in working condition.", False
            else:
                return f"{target_device.name} is not a repairable electronic device.",
                     False
        else:
            return f"{self.name} cannot find {obj_name} in the current location.", False

    def act(self, command):
        """Execute an ITAdmin-specific action or fall back to the Agent's act method."""
        command_mapping = {
            "repair_computer": (self.repair_electronic_device, 1),
            ...
        }

        tokens = command.split()
        if len(tokens) >= 1 and tokens[0] in command_mapping:
            action_func, arg_count = command_mapping[tokens[0]]
            if len(tokens[1:]) >= arg_count:
                return action_func(*tokens[1:1 + arg_count])
            return f"{self.name} received an incorrect number of arguments.", False

        return super().act(command)
```

Figure 14: Illustration of the `ITAdmin` class, where `repair_computer()` enables role-specific actions via command mapping.