Virtual Community: An Open World for Humans, Robots, and Society

Qinhong Zhou[1*] **Hongxin Zhang**[1*] **Xiangye Lin**[1*] **Zheyuan Zhang**[2*]
Yutian Chen[3] **Wenjun Liu**[1] **Zunzhe Zhang**[1] **Sunli Chen**[1] **Lixing Fang**[1]
Qiushi Lyu[1] **Xinyu Sun**[1] **Jincheng Yang**[1] **Zeyuan Wang**[1] **Bao Chi Dang**[1]
Zhehuan Chen[1] **Daksha Ladia**[1] **Jiageng Liu**[1] **Chuang Gan**[1]
[1]UMass Amherst [2] Johns Hopkins University [3]Carnegie Mellon University

https://virtual-community-ai.github.io/

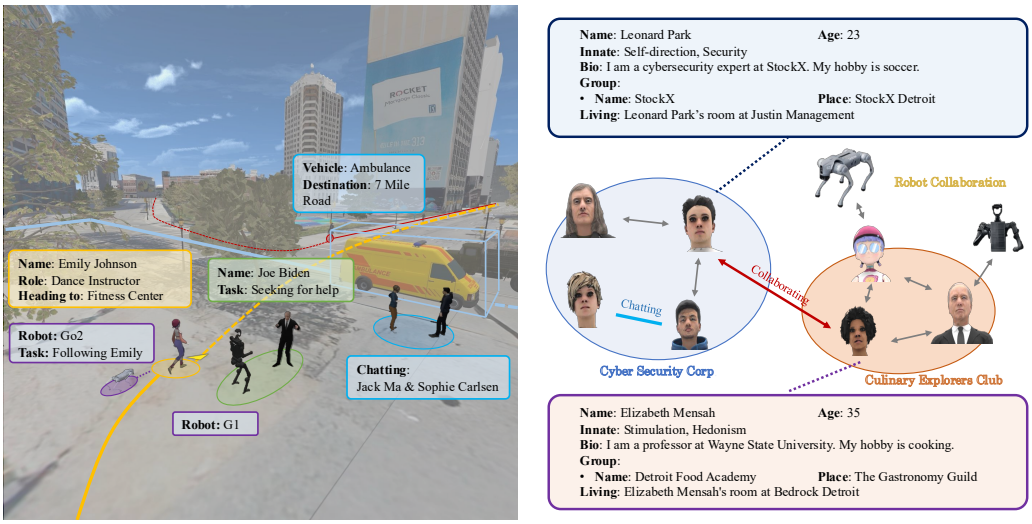


Figure 1: **Virtual Community enables simulation of humans, robots, and their societies within open-world environments.** We present an automated pipeline that transforms real-world geospatial data into large-scale 3D scenes and contextually grounded agent communities. Built on a universal physics engine, our system supports a diverse range of robots—including mobile manipulators, quadrupeds, humanoids, and drones—as well as visually expressive avatars capable of realistic motion and language-based interactions. This foundation facilitates complex social interactions and collective behaviors within the simulated communities.

Abstract

The rapid progress in AI and Robotics may lead to a profound societal transformation, as humans and robots begin to coexist within shared communities, introducing both opportunities and challenges. To explore this future, we present Virtual Community—an open-world platform for humans, robots, and society—built on a universal physics engine and grounded in real-world 3D scenes. With Virtual Community, we aim to study embodied social intelligence at scale: 1) How robots can intelligently cooperate or compete 2) How humans develop social relations and build community 3) More importantly, how intelligent robots and humans can co-exist in an open world. To support these, Virtual Community features: 1) An open-source multi-agent physics simulator supports robots, humans, and their interactions within a society; 2) A large-scale, real-world aligned community generation pipeline, including vast outdoor space, diverse indoor scenes, and a community of

Preprint.

grounded agents with rich characters and appearances. Leveraging Virtual Community, we introduce two novel tasks: the multi-agent *Campaign* task measures skill in connecting with and persuading other; and the multi-agent *Community Assistant* task tests cooperative reasoning and planning. We evaluate various baselines on these tasks and demonstrate the challenges in both high-level open-world task planning and low-level cooperation controls. We hope that Virtual Community will unlock further study of human-robot cohabitation within open-world environments.

1 Introduction

In recent years, the development of intelligent embodied agents has been propelled by advances in virtual simulators [49, 31, 3, 63, 48, 33, 76, 37, 22, 11, 74, 67, 34, 84]. However, most of these platforms focus on robots [62, 76, 34], human-like agents [42, 43], or only a limited number of agents with simple interactions [44, 22]. In contrast, support for large, heterogeneous communities of human and robot agents collaborating in scalable open worlds remains limited, constraining the study of complex multi-agent behaviors between humans and robots.

To address this challenge, simulators must support the following key features. First, they should offer physically realistic simulations that accommodate large communities of human-like avatars and robots. Existing multi-agent embodied AI platforms [42, 44, 22, 33, 74] typically handle only small groups of avatars or robots, or provide limited physics-based interactions, thereby constraining the realism of community-level behaviors. Second, the simulator must support the creation of diverse, populated worlds, including large-scale 3D environments and scene-grounded agent communities. Current approaches fall into two categories: manual design or procedural generation [67, 22, 65, 74, 23], which enable rich agent–environment interactions but suffer from limited diversity and realism; and 3D reconstruction methods [49, 54], which produce visually realistic and varied scenes but require extensive visual input and often yield low-interactivity environments in open-world settings.

In this paper, we present Virtual Community, an open world for humans, robots, and society. Virtual Community addresses these challenges by building a unified simulation framework for human-like agents and robots based on the Genesis [4] physics engine and integrating large-scale, real-world geospatial data with generative models to produce interactive, scalable open worlds (Figure 1). The platform offers two key advancements:

Unified Simulation for Avatars and Robots Virtual Community simulates human-like avatars and diverse robots within the generated open worlds using a unified framework based on the Genesis [4] physics engine, supporting diverse physical and social interactions among different types of agents. Virtual Community also provides robot and avatar agents with a unified interface with distinct observation and action spaces.

Open World Generation from Real Scenarios. Virtual Community fully automates the generation of open worlds with several key features: (1) scalable, real-world–aligned outdoor scenes of customizable size and quantity, along with corresponding indoor scenes and annotations; and (2) generation of agent communities endowed with scene-grounded profiles and social relationship networks. Virtual Community combines generative models with real-world geospatial data, ensuring scalability in data volume, realism, and extent.

Virtual Community enables a variety of new possibilities in embodied AI research. The expansive robot and human communities introduce new challenges for multi-agent interaction in open worlds. We introduce two novel tasks to address these challenges. In the campaign task, agents must efficiently explore the community and connect with one another. Virtual Community also supports physically realistic simulations of interactions among community agents, for which we propose the *Community Assistant* task. This task engages heterogeneous robots cooperating to complete missions that assist human agents.

Our simulator advances the field by enabling unified simulations of human and robot communities in generated open worlds, surpassing existing solutions in both scope and capability. By overcoming limitations in the scalable simulation of humans, robots, and societies, we pave the way for studying embodied general intelligence that can coexist with complex, interconnected human communities.

2 Related Works

Embodied AI Simulation Recently, embodied AI has seen significant advancements through the development of simulation platforms. Most existing simulators primarily focus on household tasks within indoor environments [5, 49, 80, 12, 77, 54, 49, 33, 42, 31, 78, 34, 62, 13, 15], while some have extended support to outdoor scenes [22, 65, 67, 17, 30, 24]. However, existing platforms lack the diverse and scalable outdoor environments needed to support larger agent populations and more complex multi-agent interactions. In contrast, this paper introduces a simulation platform with expansive open-world environments, integrating both indoor and scalable outdoor scenes to facilitate broader agent interactions and enable more intricate task scenarios.

Embodied Social Intelligence Current research on *Embodied Social Intelligence* is often limited to small agent populations in constrained household scenarios [43, 82, 56, 49, 29, 60, 83] or simplified to 2D or grid worlds [10, 57, 64, 47, 81], hindering model development in the open world. Specifically, [40] demonstrates the robust simulation of human-like agents within a symbolic community, ignoring the 3D perception and realistic physics in the open world. [71] studies human-like simulation guided by system 1 processing with basic needs. Predominant approaches, such as multi-agent reinforcement learning (MARL) and other planning models, face several limitations when applied to open-world settings. MARL, for instance, often struggles with scalability due to the exponential growth of state and action spaces as the number of agents increases [73]. This makes it difficult to learn effective policies in complex, dynamic environments. Additionally, MARL approaches typically require extensive training data and computational resources, which may not be feasible in real-world applications. Other planning models, while potentially more efficient, often lack the adaptability required to handle the unpredictable nature of open-world interactions. They may rely on predefined rules or assumptions that do not hold in all scenarios, leading to suboptimal performance and limited generalization to new contexts [43].

Foundation and Generative models for Embodied AI With the recent advance of foundation models [9, 35, 18, 8], numerous works have explored how they can help build powerful embodied agents [68, 75, 58, 72, 2, 52, 66, 40, 27, 7], and scenes for simulation [26, 51, 16, 21, 79, 20, 61, 41, 53, 14]. RoboGen [70] utilizes foundation models to automatically generate diverse tasks, scenes, and training supervision, scaling up robotic skill learning with minimal human input. In contrast, our work fully integrates a generative pipeline into the simulation platform to create expansive open-world scenes and agent communities.

3 Generating Open Worlds for Simulation

3.1 Scalable 3D Scene Creation

The existing 3D geospatial data API[1] provides extensive data in terms of quantity and diversity. However, they are not directly suitable for embodied AI research because of several limitations. First, these geospatial data often contain noise, including pedestrians, vehicles, other transient objects, and unrealistically rugged terrain that can disrupt simulations. Second, visual quality is inadequate for ground-level agent perspectives because these environments are typically reconstructed from aerial imagery, leading to less detailed textures and geometries at street level.

To bridge this gap, we propose an online pipeline that performs comprehensive cleaning and enhancement in both geometry and texture to make the scenes suitable for embodied AI simulations. This pipeline consists of four steps: mesh simplification, texture refinement, object placement, and automatic annotation. We generated 35 annotated scenes of various cities worldwide with this pipeline and present some qualitative examples of these scenes in Figure 3.

Geometry Reconstruction and Simplification Since the mesh topologies in 3D geospatial data sources are unreliable for embodied AI simulations, we decompose scenes into terrain, building, and decorative-roof elements, then apply specialized reconstruction operations to each component to make the entire scene simulation-ready. The terrain is generated procedurally from sparse reference elevation points via bilinear interpolation. We then derive simple, topologically sound building meshes using OpenStreetMap (OSM) data. Each building mesh is automatically adjusted to better match the Google 3D Tiles geometry and to align with the terrain elevation. By aligning mesh

[1] https://www.google.com/maps/

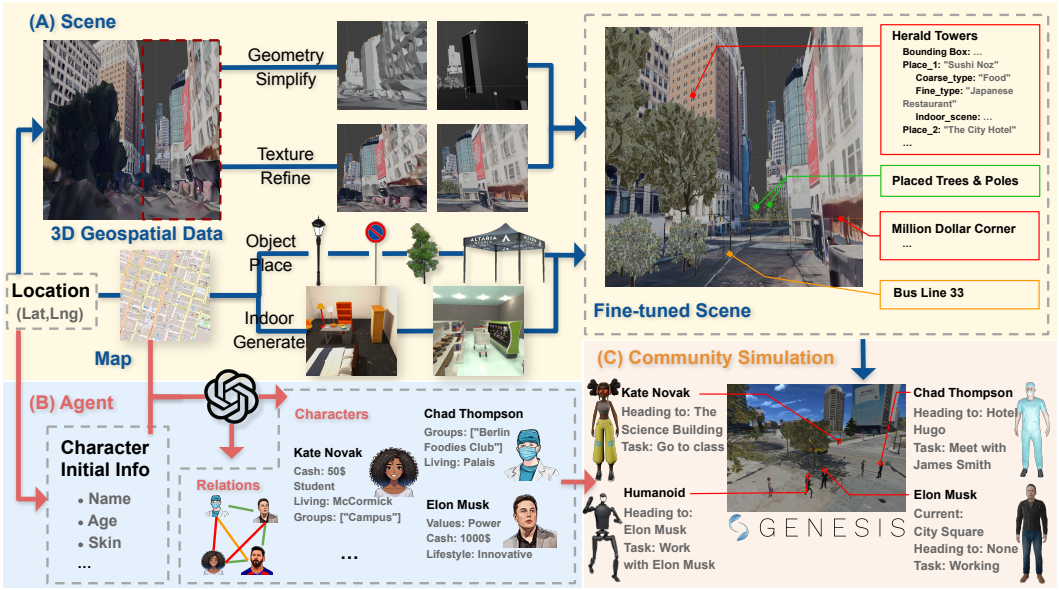


Figure 2: **Framework of the Virtual Community Generation Pipeline.** This pipeline generates scenes and corresponding agents from real-world geospatial data. The **scene generation** component (A) refines rough 3D data by using generative models to enhance textures and geospatial data to simplify geometry. It also utilizes generative methods to create interactive objects and detailed indoor scenes. The **agent generation** component (B) leverages LLMs to generate agent characters and social relationship networks based on scene descriptions. (C) We simulate the avatar communities and robots in the open world scenes based on Genesis engine.

geometries to OSM primitives, we remove unnecessary details and artifacts—such as distorted surfaces and irregular shapes resulting from aerial reconstruction errors—thereby denoising the meshes for more efficient physics simulations and improved rendering performance.

Texture Enhancement for Realistic Simulation We further apply advanced image-processing techniques to enhance mesh textures. During mesh construction and simplification, textures from the original 3D Tiles are baked onto new geometries, which can result in missing or distorted regions. To address these issues, we first employ a Stable Diffusion 3 [55] based inpainting method to remove noise and repair damaged or incomplete textures. We then refine texture details using street-view imagery. This two-step process significantly improves visual fidelity, making textures more suitable for ground-level rendering.

Object Replacement for Interactive Scene To enhance scene interactivity, we combine generative and retrieval methods to populate the environment with interactive objects (e.g., bikes and tents). Using OSM annotations, we identify object types and locations to reflect real-world contexts. For relatively simple objects, such as tents, we adopt a generative pipeline that uses OSM text annotations on amenities as input: a Stable Diffusion model [46] first generates images of the relevant objects, which are then processed by the One-2-3-45 framework [36] to produce corresponding 3D meshes. For more complex objects, such as trees, we use the retrieval pipeline, which randomly samples assets whose categories match the OSM annotations from a pre-collected dataset.

Place and Transit Annotations with Geospatial Data To facilitate alignment with real-world locations and provide semantic context for community activities, we developed a pipeline to automatically annotate places, buildings, and public transit within scenes using geospatial data. First, we query Google Maps Places for location information in the target area and organize results into

Figure 3: **Egocentric view of the generated scenes.** The resulting scene features clean geometry and realistic textures, which support physical simulation and enhance real-world style fidelity.

six categories: accommodation, entertainment, food, office, stores, and open spaces. Next, we use OSM to retrieve building names and bounding boxes, matching them with the place entries. We then filter out unmatched or inaccessible locations to generate accurate place annotations. Finally, we annotate bus transit routes based on these place annotations. These metadata enable agents to access location-specific information and support tasks that require spatial context—such as navigation and location-based decision-making—and also power traffic simulation, including buses, pedestrians, and other vehicles.

Indoor Scenes Creation To create indoor scenes in the communities, we employ a pipeline combining generation and retrieval to produce detailed, realistic multi-room environments. The pipeline's input is the building names in the target area, obtained from Google Maps and OSM. We first retrieve indoor layouts from GRUTopia [67] for categories such as offices, restaurants, and stores. For building types not covered by GRUTopia, we use Architect [69] to generate the corresponding indoor rooms for simulation.

3.2 Agent Community Generation

Given diverse generated scenes from real-world geospatial data, we introduce a generative pipeline to populate these environments with communities of agents endowed with grounded character profiles and social relationship networks, define their embodiments.

Characters and Social Network Generation We utilize the open-world knowledge of the Large Language Model (LLM) to generate agent character profiles and personalities grounded in the scene. The input to the LLM is structured into two parts to create characters grounded in a specific scene. The first part contains scene-related information, such as the scene name and details about various places, including their names, types, and functionalities. The second part includes details on the agents' appearances to ensure consistency between their visual attributes and generated profiles, which are annotated with the name and age. With both parts provided, the LLM generates agent profiles along with their social relationships. The profiles consist of basic attributes such as names, ages, occupations, personalities, and hobbies, which influence each agent's daily activities. Social relationships are structured as groups, each containing a subset of agents along with a text description and a designated place for group activities, connecting these agents into a cohesive community.

Grounding Validator We implemented a grounding validator that verifies whether generated character profiles are accurately grounded in the scene by ensuring all referenced places exist. If validation fails, the LLM receives feedback from the validator and attempts to correct the mismatch. Detailed examples of prompts used in the pipeline, generated characters, and social relationship networks are provided in the Appendix.

Human-Like Avatar Creation We first obtained 20 avatar skins representing diverse genders, professions, and appearances from Mixamo[2] for integration into the Virtual Community. We also used the Avatar SDK[3] to generate high-fidelity human meshes from real-world images, enabling representation of diverse individuals, including celebrities. Virtual Community supports 62 celebrity skins across five domains.

[2] https://www.mixamo.com/
[3] https://avatarsdk.com

4 Unified Simulation for Human and Robot Community

Virtual Community provides a unified framework for simulating both robots and human agents in the community. It implements an avatar simulation framework for human agents, while robot simulations are largely inherited from Genesis.

Avatar Simulation and Control To simulate avatars in Genesis physics engine, we combine SMPL-X human skeletons with these avatar skins to model human avatars. The motions of these avatars are parameterized by SMPL-X pose vectors $J \in \mathbb{R}^{162}$ and global translation and rotation vectors $T, R \in \mathbb{R}^3$. Specifically, we download FBX files from Mixamo that record human skeletal motion sequences and parse them into a hierarchical structure of human joints. Each skeleton joint is mapped one by one with the joints of the SMPL-X model. Then, we recursively traverse the joint tree structure to calculate the global coordinate system vector for each joint after its rotation at each time step t, and use this to drive the movement of the human skeleton. Based on these pose representations, each avatar's skin mesh is computed via forward kinematics. Our humanoid motion model supports over 2,000 distinct actions, such as walking, object manipulation, and vehicle entry. We use motion clips from Mixamo and adjust their playback speeds to match our avatars. For walking, we loop the clip until the avatar covers the required distance. For object-related actions, objects are kinematically attached to or detached from the avatars' hands based on the action. Similarly, during vehicle-related motions, avatars are kinematically attached to or detached from vehicles. We also incorporate physics constraints: collision detection is performed between avatars and scene entities, and motion terminates upon detection of a potential collision.

Daily Schedule Generation and Simulation Given the scene-grounded character profiles and social relationship networks, we prompt foundation models to generate each agent's daily schedule [40]. However, we structure each schedule so that every activity includes a start time, an end time, an activity description, and a corresponding location. We also explicitly account for the commute time between activities at different locations to reflect the actual cost of navigating an expansive 3D environment. During simulation, agents follow the generated schedules to carry out daily activities. Examples of detailed daily plans are provided in the Appendix.

Dynamic Traffic Flow Simulation We developed an automated dynamic traffic generation mechanism based on OSM data, enabling rapid reconstruction of urban road networks and autonomous traffic simulation worldwide.

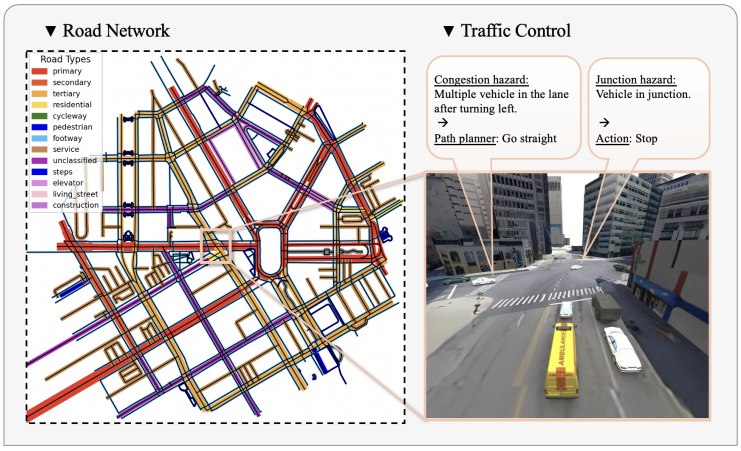


Figure 4: **Traffic flow example**. We use the road map (left) to generate the corresponding traffic flow at one of the junctions (right).

First, road data for target cities is retrieved via the OSM API and then converted into structured, semantically rich road models in OpenDRIVE format. This format supports lane-level accuracy and includes road geometry and connectivity, providing a high-fidelity foundation for traffic simulation.

Based on this road network, we implement an automated Traffic Manager, capable of dynamically generating vehicles and pedestrians and enabling their autonomous navigation and behavior control.

The Traffic Manager integrates path planning, traffic behavior modeling, and collision avoidance mechanisms to ensure safe and efficient movement of traffic agents according to predefined strategies. The system allows configuration of parameters such as vehicle and pedestrian numbers, offering flexibility and scalability.

Figure 4 shows a road map and part of the traffic flow in Detroit. Since the traffic simulation takes OSM data as input, it can be generated in any scene where OSM data is available.

Robot Agent Simulation We simulate robots alongside avatars in the Genesis simulator. Virtual Community supports five types of robots: drones, quadratic robots, humanoid robots, wheeled robots, and mobile manipulators, each with a distinct robot controller. The robot controller bridges the interface between Virtual Community and Genesis, exposing only selected action spaces. For example, the action space of Google Robot, a mobile manipulator, comprises an 11-dimensional control signal (7 DoF for the arm, 2 for the gripper, and 2 for locomotion), in velocity, position, or force control. Virtual Community shares the same simulation loop between avatars and robots with different control frequencies. To support faster collision detection during robot physics simulation, we use an invisible terrain mesh and decomposed building meshes as collision geometry for the background scene, enabling more efficient physics simulation.

5 Community Multi-Agent Tasks

Leveraging the new capabilities unlocked by Virtual Community, we introduce two novel embodied multi-agent tasks: a *Campaign* task involving multiple human agents and a **Community Assistant** task involving both robots and human agents. To succeed in these tasks, agents need planning abilities within a community context and social intelligence to interact with others.

As the foundation for both tasks, agents in the community follow a default daily plan and routineif no specific tasks are assigned. At each episode, several agents are selected and assigned one task. When an agent is given a task, it suspends its daily plan and focuses on completing the assigned social task in the community.

5.1 Campaign: Connection and Influence in the Society

To explore the social abilities of human agents in an open-world, multi-agent setting, we designed a novel open-ended *Campaign* task in which candidate agents must efficiently plan to connect with and persuade voter agents within the community. Because voters differ in character and social ties, some may initially favor certain candidates, requiring each candidate to devise adaptive strategies to influence and shift voter opinions throughout the election process.

Task settings In each scene, Agents are divided into two groups: 2 candidates and 13 voters. The candidate agents must navigate the community, locate potential voters, and attempt to persuade them through communication. The election concludes at the end of the day, with voters prompted to cast their ballots.

Observation and action spaces The observation and action spaces for all agents are the same as in the *Assistant tasks*, with modifications on the *communicate* action, which only allows candidates to broadcast a message to other agents within a 2-meter radius. To assist candidates in locating voters, both candidates are given access to the daily plans of all voters.

Experimental settings and evaluation metrics We conduct experiments using a Donald Trump agent and a Kamala Harris agent as candidates across five different scenes. Performance in the election is measured by the vote share among all voter agents.

Baseline We implement all baseline agents within a hierarchical framework. Since our focus is the open-world planning ability, all agents use the same low-level point-based navigation algorithm, which reconstructs the point cloud based on RGB-D images from the ego-centric observation at each step and converts the point cloud into a volume grid representation with a resolution of $0.1m$. Subsequently, a 2D occupancy map is established with a resolution of $0.5m$ based on this representation and an A* algorithm is used to search for the shortest path efficiently. For high-level planner, we evaluate

Figure 5: **Results on the Campaign task across five cities**. Results show that more powerful LLMs are better able to connect with and influence other agents in the community.

different LLMs as the planning backbone for the candidate agent. Given voters' daily schedules and character traits, the candidate agent prompts the LLM to select the next voter to visit based on distance and potential influence. After navigating to the target voter, we use the same LLM backbone to generate up to three rounds of conversation with the voter, using the candidate and voter profiles as input. Upon conclusion of the conversation, the candidate identifies and proceeds to persuade the next voter. More details are provided in the Supplementary Material.

Metrics We use two metrics: average vote share (Share) and conversion rate (Conv.). Vote share is defined as the number of votes gained at the end of the day divided by the total number of voters. Conversion rate is defined as the number of new supporters gained during the day divided by the number of originally non-supporting voters.

Results As shown in Figure 5, the candidate with `gpt-4o` backbone has a higher average vote share and conversion rate than the GPT-35-turbo backbone, which means it is more capable able to change voters opinion on most of the scenes. Specifically, in the Denver experiments in Figure 5, the Trump Candidate persuaded Brian Carter by focusing on his identity as a member of the library book club during their conversation, affirming the value of libraries in city development and making several commitments. We also observe that with a large advantage, the Harris Candidate the GPT-4o backbone didn't get any more votes mostly due to the wrong target selection and less effective persuasion strategy. The results show even with advanced LLMs, there is still much room to improve the social connection and influence-build ability of embodied agents.

5.2 Community Assistant: Robots Collaborating to Assist Humans

In addition to high-level, competitive planning among human agents, we also address community challenges in low-level, cooperative settings for both robot and human agents. We introduce the *Community Assistant* task, which features scenarios where two heterogeneous robots cooperate to assist humans in open-world environments.

Task settings The community assistant tasks include the following categories, which require agents to plan cooperatively to assist human avatars with daily activities—*Carry*, where agents accompany people outdoors while helping to carry objects; and *Delivery*, where agents move objects from source locations (indoor or outdoor) to a destination.

Task Generation We employ a procedural task-generation pipeline that produces tasks for all scenes. The pipeline consists of:

- **Place Selection**: When a task requires a specific location (e.g., the destination in a carry task), we use the agent's profile and a list of known places as inputs, and prompt a large language model to select a valid target. Outdoor regions are included among the options.

Table 1: **Results of the Community Assistant Task.** We report Success Rate (SR) and Time Consumed (Ts) for each task.

Method	Carry		Deliver		Avg SR↑
	SR↑	Ts↓	SR↑	Ts↓	
Heuristic	17.6	126.9	22.2	**129.4**	19.9
Heuristic w Oracle Grasp	**23.5**	**124.4**	**50.0**	131.2	**36.8**

- **Object Selection**: To determine the target object, we prompt the language model with the task description and provide nine candidate object types for it to choose from.
- **Object Placement**: After the object type and place are chosen, we retrieve the corresponding asset and position it appropriately.
  - *Outdoor locations*: placed at a random point not occluded by any building.
  - *Indoor locations*: placed on the surface of a randomly selected semantic container (floor, sofa, table, chair, desk, or bed).
- **Evaluation**: Once the object is placed, the pipeline automatically generates evaluation metadata (e.g., bounding boxes for the target object or agent). After the agent signals task completion, the simulator checks whether the success criteria are met and records the result.

Robot Settings We use two robot assistants: a mobile manipulator—based on the Google robot model in MuJoCo [63], augmented with one degree of freedom for forward translation and another for rotation about the z-axis [31, 49, 62]—and a wheeled robot carrier with four degrees of freedom (one per wheel).

Observation and Control Spaces Robots read buffered observations and apply control signals at 100 Hz, including base and joint velocities, joint positions, and clock inputs. Additionally, at 1 Hz, robots receive RGB–D images, segmentation masks, base pose, and task-related information.

Baseline Pipeline We implement the baseline in a hierarchical manner by inheriting the navigation module from the avatars used in the *Campaign* task. For low-level navigation, the robot computes collision-free paths using A* search. For pickup actions, it computes a feasible grasp pose via inverse kinematics and plans and executes the grasp motion with RRT-Connect [32]. For the high-level planner, we employ a heuristic approach with task-specific priors to generate subplans following predefined patterns.

Results of the Community Assistant Task Table 1 shows both baselines perform better on delivery than on carry, reflecting the added difficulty of simultaneous object manipulation and human following in dynamic open worlds. Performance drops sharply without an oracle grasp, underscoring manipulation challenges.

6 Conclusion

We introduce Virtual Community, an open-world simulation platform for multi-agent embodied AI that supports unified simulation of human and robot agents in scalable open worlds, along with physically realistic simulation for agent interactions. As an initial demonstration, we propose two novel open-world multi-agent tasks—the **Campaign** and the **Community Assistant**. We hope Virtual Community will advance embodied AI research toward embodied general intelligence capable of handling real-world complexities and coexisting with human communities.

Future Directions

The proposed open-world simulation framework in Virtual Community opens up multiple meaningful yet challenging research directions for studying the social intelligence of embodied agents. To explore how humans and robots coexist in the same society, we must develop additional tasks that evaluate

robots' ability to interact intelligently with humans and other robots. To support this, we aim to simulate large numbers of agents and enable more detailed, physically realistic social interactions among them.

Acknowledgement

We gratefully acknowledge Haoyu Zhen, Ruxi Deng, Quang Dang, Hao Zou, and Siyuan Cen for their efforts in the project website and data collection. We thank Chunru Lin, Yian Wang, Xiaowen Qiu for their insightful discussion and Zhou Xian, Yuncong Yang, Zeyuan Yang, Jiaben Chen for their feedback on the project.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. 24

[2] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022. 3

[3] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. Vision-and-Language Navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[4] Genesis Authors. Genesis: A universal and generative physics engine for robotics and beyond, December 2024. 2, 20

[5] Charles Beattie, Joel Z Leibo, Denis Teplyashin, Tom Ward, Marcus Wainwright, Heinrich Küttler, Andrew Lefrancq, Simon Green, Víctor Valdés, Amir Sadik, et al. Deepmind lab. *arXiv preprint arXiv:1612.03801*, 2016. 3

[6] Mikołaj Bińkowski, Danica J Sutherland, Michael Arbel, and Arthur Gretton. Demystifying mmd gans. *arXiv preprint arXiv:1801.01401*, 2018. 22

[7] Kevin Black, Mitsuhiko Nakamoto, Pranav Atreya, Homer Rich Walke, Chelsea Finn, Aviral Kumar, and Sergey Levine. Zero-shot robotic manipulation with pre-trained image-editing diffusion models. In *The Twelfth International Conference on Learning Representations*, 2024. 3

[8] Andreas Blattmann, Robin Rombach, Huan Ling, Tim Dockhorn, Seung Wook Kim, Sanja Fidler, and Karsten Kreis. Align your latents: High-resolution video synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22563–22575, 2023. 3

[9] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. Sparks of artificial general intelligence: Early experiments with gpt-4, 2023. 3

[10] Micah Carroll, Rohin Shah, Mark K Ho, Tom Griffiths, Sanjit Seshia, Pieter Abbeel, and Anca Dragan. On the utility of learning about humans for human-ai coordination. *Advances in neural information processing systems*, 32, 2019. 3

[11] Zhili Cheng, Zhitong Wang, Jinyi Hu, Shengding Hu, An Liu, Yuge Tu, Pengkai Li, Lei Shi, Zhiyuan Liu, and Maosong Sun. Legent: Open platform for embodied agents. In *The 62nd Annual Meeting of the Association for Computational Linguistics, System Demonstration*, 2024. 2

[12] Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Embodied question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–10, 2018. 3

[13] Matt Deitke, Winson Han, Alvaro Herrasti, Aniruddha Kembhavi, Eric Kolve, Roozbeh Mottaghi, Jordi Salvador, Dustin Schwenk, Eli VanderBilt, Matthew Wallingford, et al. Robothor: An open simulation-to-real embodied ai platform. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3164–3174, 2020. 3

[14] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13142–13153, 2023. 3

[15] Matt Deitke, Eli VanderBilt, Alvaro Herrasti, Luca Weihs, Kiana Ehsani, Jordi Salvador, Winson Han, Eric Kolve, Aniruddha Kembhavi, and Roozbeh Mottaghi. Procthor: Large-scale embodied ai using procedural generation. *Advances in Neural Information Processing Systems*, 35:5982–5994, 2022. 3

[16] Matt Deitke, Eli VanderBilt, Alvaro Herrasti, Luca Weihs, Kiana Ehsani, Jordi Salvador, Winson Han, Eric Kolve, Aniruddha Kembhavi, and Roozbeh Mottaghi. Procthor: Large-scale embodied ai using procedural generation. *Advances in Neural Information Processing Systems*, 35:5982–5994, 2022. 3

[17] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017. 3, 18

[18] Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, Andy Zeng, Igor Mordatch, and Pete Florence. Palm-e: An embodied multimodal language model. In *arXiv preprint arXiv:2303.03378*, 2023. 3

[19] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. 24

[20] Weixi Feng, Wanrong Zhu, Tsu-jui Fu, Varun Jampani, Arjun Akula, Xuehai He, Sugato Basu, Xin Eric Wang, and William Yang Wang. Layoutgpt: Compositional visual planning and generation with large language models. *Advances in Neural Information Processing Systems*, 36, 2024. 3

[21] Huan Fu, Bowen Cai, Lin Gao, Ling-Xiao Zhang, Jiaming Wang, Cao Li, Qixun Zeng, Chengyue Sun, Rongfei Jia, Binqiang Zhao, et al. 3d-front: 3d furnished rooms with layouts and semantics. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10933–10942, 2021. 3

[22] Chuang Gan, Jeremy Schwartz, Seth Alter, Damian Mrowca, Martin Schrimpf, James Traer, Julian De Freitas, Jonas Kubilius, Abhishek Bhandwaldar, Nick Haber, et al. Threedworld: A platform for interactive multi-modal physical simulation. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021. 2, 3

[23] Chen Gao, Baining Zhao, Weichen Zhang, Jinzhu Mao, Jun Zhang, Zhiheng Zheng, Fanhang Man, Jianjie Fang, Zile Zhou, Jinqiang Cui, et al. Embodiedcity: A benchmark platform for embodied agent in real-world city environment. *arXiv preprint arXiv:2410.09604*, 2024. 2

[24] Cole Gulino, Justin Fu, Wenjie Luo, George Tucker, Eli Bronstein, Yiren Lu, Jean Harb, Xinlei Pan, Yan Wang, Xiangyu Chen, et al. Waymax: An accelerated, data-driven simulator for large-scale autonomous driving research. *Advances in Neural Information Processing Systems*, 36:7730–7742, 2023. 3

[25] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017. 22

[26] Lukas Höllein, Ang Cao, Andrew Owens, Justin Johnson, and Matthias Nießner. Text2room: Extracting textured 3d meshes from 2d text-to-image models. *arXiv preprint arXiv:2303.11989*, 2023. 3

[27] Mineui Hong, Minjae Kang, and Songhwai Oh. Diffused task-agnostic milestone planner. *Advances in Neural Information Processing Systems*, 36, 2024. 3

[28] Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, Kai Dang, Yang Fan, Yichang Zhang, An Yang, Rui Men, Fei Huang, Bo Zheng, Yibo Miao, Shanghaoran Quan, Yunlong Feng, Xingzhang Ren, Xuancheng Ren, Jingren Zhou, and Junyang Lin. Qwen2.5-coder technical report, 2024. 24

[29] Unnat Jain, Luca Weihs, Eric Kolve, Ali Farhadi, Svetlana Lazebnik, Aniruddha Kembhavi, and Alexander Schwing. A Cordial Sync: Going beyond marginal policies for multi-agent embodied tasks. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part V 16*, pages 471–490. Springer, 2020. 3

[30] Alex Kendall, Jeffrey Hawke, David Janz, Przemyslaw Mazur, Daniele Reda, John-Mark Allen, Vinh-Dieu Lam, Alex Bewley, and Amar Shah. Learning to drive in a day. *CoRR*, abs/1807.00412, 2018. 3

[31] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Matt Deitke, Kiana Ehsani, Daniel Gordon, Yuke Zhu, et al. Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*, 2017. 2, 3, 9, 20

[32] James J Kuffner and Steven M LaValle. Rrt-connect: An efficient approach to single-query path planning. In *Proceedings 2000 ICRA. Millennium conference. IEEE international conference on robotics and automation. Symposia proceedings (Cat. No. 00CH37065)*, volume 2, pages 995–1001. IEEE, 2000. 9

[33] Chengshu Li, Fei Xia, Roberto Martín-Martín, Michael Lingelbach, Sanjana Srivastava, Bokui Shen, Kent Elliott Vainio, Cem Gokmen, Gokul Dharan, Tanish Jain, et al. igibson 2.0: Object-centric simulation for robot learning of everyday household tasks. In *5th Annual Conference on Robot Learning*, 2021. 2, 3

[34] Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-Martín, Chen Wang, Gabrael Levine, Wensi Ai, Benjamin Martinez, et al. Behavior-1k: A human-centered, embodied ai benchmark with 1,000 everyday activities and realistic simulation. *arXiv preprint arXiv:2403.09227*, 2024. 2, 3

[35] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *arXiv preprint arXiv:2304.08485*, 2023. 3

[36] Minghua Liu, Chao Xu, Haian Jin, Linghao Chen, Mukund Varma T, Zexiang Xu, and Hao Su. One-2-3-45: Any single image to 3d mesh in 45 seconds without per-shape optimization. *Advances in Neural Information Processing Systems*, 36, 2024. 4

[37] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, et al. Isaac gym: High performance gpu based physics simulation for robot learning. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. 2

[38] Volodymyr Mnih. Asynchronous methods for deep reinforcement learning. *arXiv preprint arXiv:1602.01783*, 2016. 24

[39] David Nistér. An efficient solution to the five-point relative pose problem. *IEEE transactions on pattern analysis and machine intelligence*, 26(6):756–770, 2004. 17

[40] Joon Sung Park, Joseph C O'Brien, Carrie J Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. *arXiv preprint arXiv:2304.03442*, 2023. 3, 6, 18

[41] Despoina Paschalidou, Amlan Kar, Maria Shugrina, Karsten Kreis, Andreas Geiger, and Sanja Fidler. Atiss: Autoregressive transformers for indoor scene synthesis. *Advances in Neural Information Processing Systems*, 34:12013–12026, 2021. 3

[42] Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8494–8502, 2018. 2, 3

[43] Xavier Puig, Tianmin Shu, Shuang Li, Zilin Wang, Yuan-Hong Liao, Joshua B Tenenbaum, Sanja Fidler, and Antonio Torralba. Watch-and-help: A challenge for social perception and human-ai collaboration. In *International Conference on Learning Representations*, 2020. 2, 3

[44] Xavier Puig, Eric Undersander, Andrew Szot, Mikael Dallaire Cote, Tsung-Yen Yang, Ruslan Partsey, Ruta Desai, Alexander Clegg, Michal Hlavac, So Yeon Min, et al. Habitat 3.0: A co-habitat for humans, avatars, and robots. In *The Twelfth International Conference on Learning Representations*, 2023. 2, 23

[45] Tianhe Ren, Shilong Liu, Ailing Zeng, Jing Lin, Kunchang Li, He Cao, Jiayu Chen, Xinyu Huang, Yukang Chen, Feng Yan, Zhaoyang Zeng, Hao Zhang, Feng Li, Jie Yang, Hongyang Li, Qing Jiang, and Lei Zhang. Grounded sam: Assembling open-world models for diverse visual tasks, 2024. 17

[46] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2021. 4, 17

[47] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim GJ Rudner, Chia-Man Hung, Philip HS Torr, Jakob Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 2186–2188, 2019. 3

[48] Manolis Savva, Angel X. Chang, Alexey Dosovitskiy, Thomas Funkhouser, and Vladlen Koltun. MINOS: Multimodal indoor simulator for navigation in complex environments. *arXiv:1712.03931*, 2017. 2

[49] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. Habitat: A platform for embodied ai research. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9339–9347, 2019. 2, 3, 9, 20

[50] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 24

[51] Jonas Schult, Sam Tsai, Lukas Höllein, Bichen Wu, Jialiang Wang, Chih-Yao Ma, Kunpeng Li, Xiaofang Wang, Felix Wimbauer, Zijian He, et al. Controlroom3d: Room generation using semantic proxy rooms. *arXiv preprint arXiv:2312.05208*, 2023. 3

[52] Pratyusha Sharma, Antonio Torralba, and Jacob Andreas. Skill induction and planning with latent language. *arXiv preprint arXiv:2110.01517*, 2021. 3

[53] Volodymyr Shcherbyna, Linh Kästner, Diego Diaz, Huu Giang Nguyen, Maximilian Ho-Kyoung Schreff, Tim Lenz, Jonas Kreutz, Ahmed Martban, Huajian Zeng, and Harold Soh. Arena 4.0: A comprehensive ros2 development and benchmarking platform for human-centric navigation using generative-model-based environment generation. *arXiv preprint arXiv:2409.12471*, 2024. 3

[54] Bokui Shen, Fei Xia, Chengshu Li, Roberto Martín-Martín, Linxi Fan, Guanzhi Wang, Claudia Pérez-D'Arpino, Shyamal Buch, Sanjana Srivastava, Lyne Tchapmi, et al. igibson 1.0: A simulation environment for interactive tasks in large realistic scenes. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7520–7527. IEEE, 2021. 2, 3

[55] Stability AI. Stable diffusion 3 research paper. https://stability.ai/news/stable-diffusion-3-research-paper, 2024. Accessed: 2025-05-15. 4

[56] Peter Stone, Rodney Brooks, Erik Brynjolfsson, Ryan Calo, Oren Etzioni, Greg Hager, Julia Hirschberg, Shivaram Kalyanakrishnan, Ece Kamar, Sarit Kraus, et al. Artificial intelligence and life in 2030: the one hundred year study on artificial intelligence. *arXiv preprint arXiv:2211.06318*, 2022. 3

[57] Joseph Suarez, Yilun Du, Phillip Isola, and Igor Mordatch. Neural MMO: A massively multiagent game environment for training and evaluating intelligent agents. *arXiv preprint arXiv:1903.00784*, 2019. 3

[58] Theodore Sumers, Shunyu Yao, Karthik Narasimhan, and Thomas L Griffiths. Cognitive architectures for language agents. *arXiv preprint arXiv:2309.02427*, 2023. 3

[59] Jiaming Sun, Zehong Shen, Yuang Wang, Hujun Bao, and Xiaowei Zhou. Loftr: Detector-free local feature matching with transformers. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2021. 17

[60] Andrew Szot, Unnat Jain, Dhruv Batra, Zsolt Kira, Ruta Desai, and Akshara Rai. Adaptive coordination in social embodied rearrangement. In *International Conference on Machine Learning*, pages 33365–33380. PMLR, 2023. 3

[61] Jiapeng Tang, Yinyu Nie, Lev Markhasin, Angela Dai, Justus Thies, and Matthias Nießner. Diffuscene: Scene graph denoising diffusion probabilistic model for generative indoor scene synthesis. *arXiv preprint arXiv:2303.14207*, 2023. 3

[62] Stone Tao, Fanbo Xiang, Arth Shukla, Yuzhe Qin, Xander Hinrichsen, Xiaodi Yuan, Chen Bao, Xinsong Lin, Yulin Liu, Tse-kai Chan, et al. Maniskill3: Gpu parallelized robotics simulation and rendering for generalizable embodied ai. *arXiv preprint arXiv:2410.00425*, 2024. 2, 3, 9, 20

[63] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012. 2, 9

[64] Nathan Tsoi, Mohamed Hussein, Jeacy Espinoza, Xavier Ruiz, and Marynel Vázquez. Sean: Social environment for autonomous navigation. In *Proceedings of the 8th international conference on human-agent interaction*, pages 281–283, 2020. 3

[65] Nathan Tsoi, Alec Xiang, Peter Yu, Samuel S. Sohn, Greg Schwartz, Subashri Ramesh, Mohamed Hussein, Anjali W. Gupta, Mubbasir Kapadia, and Marynel Vázquez. Sean 2.0: Formalizing and generating social situations for robot navigation. *IEEE Robotics and Automation Letters*, pages 1–8, 2022. 2, 3

[66] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023. 3

[67] Hanqing Wang, Jiahe Chen, Wensi Huang, Qingwei Ben, Tai Wang, Boyu Mi, Tao Huang, Siheng Zhao, Yilun Chen, Sizhe Yang, Peizhou Cao, Wenye Yu, Zichao Ye, Jialun Li, Junfeng Long, ZiRui Wang, Huiling Wang, Ying Zhao, Zhongying Tu, Yu Qiao, Dahua Lin, and Pang Jiangmiao. Grutopia: Dream general robots in a city at scale. In *arXiv*, 2024. 2, 3, 5, 18

[68] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. A survey on large language model based autonomous agents. *arXiv preprint arXiv:2308.11432*, 2023. 3

[69] Yian Wang, Xiaowen Qiu, Jiageng Liu, Zhehuan Chen, Jiting Cai, Yufei Wang, Tsun-Hsuan Wang, Zhou Xian, and Chuang Gan. Architect: Generating vivid and interactive 3d scenes with hierarchical 2d inpainting. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. 5, 18

[70] Yufei Wang, Zhou Xian, Feng Chen, Tsun-Hsuan Wang, Yian Wang, Katerina Fragkiadaki, Zackory Erickson, David Held, and Chuang Gan. Robogen: Towards unleashing infinite data for automated robot learning via generative simulation. In *Forty-first International Conference on Machine Learning*, 2024. 3

[71] Zhilin Wang, Yu Ying Chiu, and Yu Cheung Chiu. Humanoid agents: Platform for simulating human-like generative agents. *arXiv preprint arXiv:2310.05418*, 2023. 3

[72] Zihao Wang, Shaofei Cai, Anji Liu, Xiaojian Ma, and Yitao Liang. Describe, explain, plan and select: Interactive planning with large language models enables open-world multi-task agents, 2023. 3

[73] Muning Wen, Jakub Kuba, Runji Lin, Weinan Zhang, Ying Wen, Jun Wang, and Yaodong Yang. Multi-agent reinforcement learning is a sequence modeling problem. *Advances in Neural Information Processing Systems*, 35:16509–16521, 2022. 3

[74] Wayne Wu, Honglin He, Yiran Wang, Chenda Duan, Jack He, Zhizheng Liu, Quanyi Li, and Bolei Zhou. Metaurban: A simulation platform for embodied ai in urban spaces. *CoRR*, 2024. 2

[75] Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, et al. The rise and potential of large language model based agents: A survey. *arXiv preprint arXiv:2309.07864*, 2023. 3

[76] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, et al. Sapien: A simulated part-based interactive environment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11097–11107, 2020. 2

[77] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, et al. Sapien: A simulated part-based interactive environment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11097–11107, 2020. 3

[78] Claudia Yan, Dipendra Misra, Andrew Bennnett, Aaron Walsman, Yonatan Bisk, and Yoav Artzi. Chalet: Cornell house agent learning environment. *arXiv preprint arXiv:1801.07357*, 2018. 3

[79] Yandan Yang, Baoxiong Jia, Peiyuan Zhi, and Siyuan Huang. Physcene: Physically interactable 3d scene synthesis for embodied ai. *arXiv preprint arXiv:2404.09465*, 2024. 3

[80] Wu Yi, Wu Yuxin, Gkioxari Georgia, and Tian Yuandong. Building generalizable agents with a realistic and rich 3d environment, 2018. 3

[81] Xianhao Yu, Jiaqi Fu, Renjia Deng, and Wenjuan Han. Mineland: Simulating large-scale multi-agent interactions with limited multimodal senses and physical needs. *arXiv preprint arXiv:2403.19267*, 2024. 3

[82] Hongxin Zhang, Weihua Du, Jiaming Shan, Qinhong Zhou, Yilun Du, Joshua B Tenenbaum, Tianmin Shu, and Chuang Gan. Building cooperative embodied agents modularly with large language models. In *The Twelfth International Conference on Learning Representations*, 2023. 3

[83] Yang Zhang, Shixin Yang, Chenjia Bai, Fei Wu, Xiu Li, Zhen Wang, and Xuelong Li. Towards efficient llm grounding for embodied multi-agent collaboration. *arXiv preprint arXiv:2405.14314*, 2024. 3

[84] Fangwei Zhong, Kui Wu, Churan Wang, Hao Chen, Hai Ci, Zhoujun Li, and Yizhou Wang. Unrealzoo: Enriching photo-realistic virtual worlds for embodied ai. *arXiv preprint arXiv:2412.20977*, 2024. 2

A 3D Scene Generation Details

In this section, we present the implementation details of the 3D scene generation pipeline. This pipeline takes three inputs—latitude, longitude, and range radius—to generate 3D community scenes for simulation, automatically outputting multiple textured 3D meshes, including buildings, terrain, roofs, and a JSON format of objects that can be loaded in the physics engine. The entire scene generation pipeline is implemented in Blender[4].

A.1 Mesh Preprocessing and Terrain Construction

Given the latitude, longitude, and range radius, we first retrieve the 3D tiles data within the specified range. The original 3D tiles data is in the Earth-Centered Earth-Fixed (ECEF) coordinate system, which is then converted to East-North-Up (ENU) coordinates, setting the mesh centroid as the origin by averaging the positions of all vertices. After this translation, we seamlessly join all tile meshes by merging each vertex near the boundary of one tile with the nearest vertex of an adjacent tile. With these preprocessing steps, we obtain a coordinate-aligned and integrated mesh for each scene.

To construct the corresponding terrain mesh, we retrieve OpenStreetMap (OSM) road and ground annotations within the specified area and align them with the mesh obtained from the previous step. By sampling latitude-longitude pairs along roads and ground surfaces and performing ray casting from these sampled points, we calculate the heightfield for each road and ground area. However, the heightfield can be noisy due to extraneous meshes, such as cars or other objects, in the 3D tiles data. To address this, we apply a rule-based filter to remove abnormal points from the heightfield. The terrain mesh is then constructed using bilinear interpolation on the cleaned heightfield.

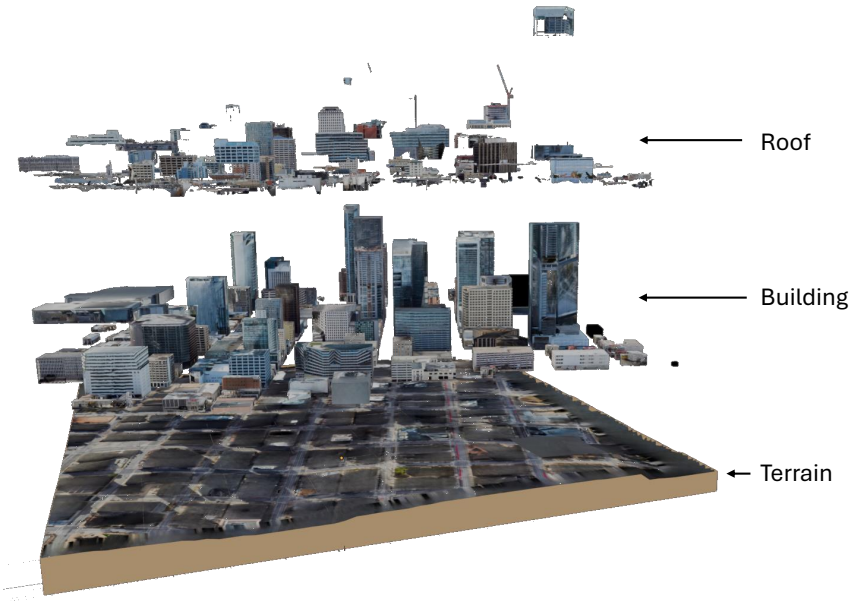


Figure 6: We decompose the outdoor 3D scene into three components - the terrain, buildings, and roofs. Each part is generated separately using different strategies to balance the visual appearance and geometry complexity.

A.2 Texture Transfer and Enhancement

The meshes in the 3D tiles data have sub-optimal geometry and noisy textures for simulation since they are reconstructed by photometric methods from aerial photos. To generate meshes suitable for simulators, we apply the *texture transfer* and *texture enhancement* step to the 3D tiles data.

[4] https://blender.org/

Texture Transfer To create topologically sound geometry for the simulator, we first construct geometries called *OSM geoms* using the *Simple 3D Buildings*[5] annotation from OpenStreetMap overpass API. These OSM geoms have significantly fewer vertices and faces compared to the 3D tiles while ensuring water-tightness and topological soundness. However, these geometries constructed by the rule-based method do not contain any texture information. We address this deficiency by baking the texture from 3D tiles to the OSM geoms using Blender. Specifically, we used the caged baking method to improve the texture transfer quality further.

Texture Enahancement Since the original texture does not have sufficient resolution for photo-realistic first-person-view rendering, we apply diffusion models such as StableDiffusion for inpainting and super-resolution [46].

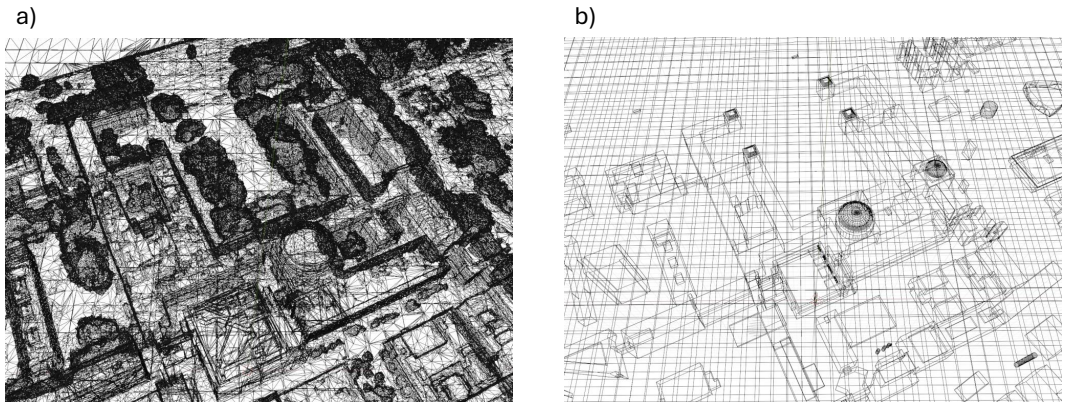


Figure 7: Comparing the scene geometry of a) raw Google 3D Tiles and b) our reconstructed scene at the Boston, MA. Our scene has much simpler geometry and reliable mesh topology, facilitating physical simulation at scale.

A.3 Street View Reprojection Details

To further enhance the realism of building and terrain texture, we utilize the street-view images from Google StreetView[6] and Mapillary[7] to fine-tune the scene texture. This process, called street-view reprojection, is composed of four major steps: 1) camera initiation, 2) view fine-tuning, 3) street-view inpainting, and 4) texture reprojection.

Camera Initilization In this step, we fetch all the street-view images in the range of 3D scenes. Using the provided metadata on longitude, latitude, orientation, and camera configuration, we instantiate cameras with corresponding intrinsic and extrinsic matrices in Blender.

View Fine-Tuning Since the image metadata are prone to sensor measurement noise, we perform an additional step of view fine-tuning to perform minor pose correction on the camera in Blender. For each camera placed in the initialization step, we first render the 3D scene mesh from the camera's perspective. The rendered image is then matched with the street-view image using LoFTR matching algorithm [59]. Using the matched keypoint pair, we are able to solve the pose difference between the street-view image and the rendering camera following the 5-point method [39]. The pose correction estimated by the 5-point method is then validated by comparing the norm of $\mathfrak{se}(3)$ pose vector with a pre-defined threshold.

Streetview Inpainting The street-view images often contain dynamic objects such as vehicles and pedestrians. These objects are intrinsically dynamic and do not have corresponding geometry on the reconstructed 3D scene. Therefore, we identify these objects using Language grounded Segment Anything (LangSAM) [45] and perform inpainting using StableDiffusion.

Texture Reprojection Finally, we project the inpainted street-view images to the 3D meshes using the corrected camera pose and *Texture Painting* feature from Blender. Since the street-view images

[5] https://wiki.openstreetmap.org/wiki/Simple_3D_Buildings
[6] https://www.google.com/streetview/
[7] https://www.mapillary.com

are taken under various lighting and weather conditions, the color distribution between images and 3D scene texture may differ significantly. To mitigate the gap between color distributions, we implement a color-correction routine that automatically aligns the color temperature and exposure of multiple images using histogram alignment. We also apply performance optimizations, including view-frustum clipping and greedy camera-mesh matching to speed up this process.

A.4 Indoor Scene Generation Pipeline

The Indoor Scene Generation Pipeline comprises two main stages to create detailed, realistic multi-room environments. Given building names in the target area—fetched from Google Maps and OpenStreetMap—as input, it outputs corresponding indoor scenes loadable in the simulator. In the retrieval stage, we query GRUTopia [67] for the most relevant indoor layout, but because GRUTopia scenes can be extremely complex, we employ the generative stage for all but the most frequently used scenes. In the generative stage, following [69], a diffusion-based inpainting model populates empty rooms with large objects, which are then detected and spatially positioned by vision models; subsequently, large-language models assist in selecting and placing suitable small objects onto or within the arranged large objects.

B Avatar Simulation

B.1 Avatar Models

We present examples of generated humanoid avatars in Figure 8. These demonstrates the capability of our method to create detailed and varied human-like avatars. Each skin model of characters includes 71 skeletal joints and can be adapted to animation sequences in SMPL-X and FBX formats. To reduce the computational load during animation playback in the Virtual Community, we optimized the skin models by applying Blender's Decimate Modifier tool, reducing the number of vertices in the 3D skin mesh by 90%.

B.2 Profile and Daily Plan Generation

An example character with social relationship networks and daily schedule generated is shown in Figure 9 (a). Given the scene-grounded characters and social relationship networks, we prompt the foundation models to generate the daily schedule for each agent, using a similar design to [40]. Differently, we generate the daily schedule in a structured manner directly with each activity represented with a start time, an ending time, an activity description, and the corresponding activity place, and consider the required commute time between adjacent activities that are happening in different places explicitly, due to the actual cost of navigating in an expansive 3D environment.

C Traffic Simulation

In this section, we present a detailed description of our traffic simulation implementation. The simulation pipeline includes two components: road network construction and traffic control. Together, these modules enable realistic and efficient urban traffic simulation.

C.1 Road Network

To implement traffic simulation, the first step is to construct an accurate and structured representation of the urban road network. Our system uses data obtained from the OSM API. Based on the raw OSM data, we build a comprehensive road information database that includes attributes such as road type, location, and width for each segment of the network.

For more advanced traffic simulation and control, we further convert the raw OSM data into the OpenDRIVE format. This format provides a highly structured and semantic-rich description of the road network, including road direction, geometry, lane configurations, and connectivity between roads. These features are essential for enabling precise vehicle navigation and traffic behavior modeling. Specifically, we use the OSM to OpenDrive converter provided by CARLA [17].

Figure 8: Some examples of generated avatars

C.2 Traffic Control

Once we have the road network, pedestrians and vehicles can be placed either manually or randomly on the map. To make their behaviors realistic and coherent, we implement a Traffic Manager module responsible for controlling and coordinating all traffic participants.

The main functions of the Traffic Manager include path planning, collision avoidance, and traffic flow regulation. It ensures that both vehicles and pedestrians follow reasonable movement patterns while maintaining safety and efficiency. Considering both realism and computational performance, we define a set of simplified traffic rules within the Traffic Manager:

- **Junction Access Control** If any pedestrian or vehicle is currently inside an junction, no other agent will enter until the junction is clear.
- **Direction Preference** When a vehicle reaches an intersection and has multiple directions to choose from, it selects the route with fewer vehicles to minimize congestion.
- **Pedestrian Behavior** Pedestrians are allowed to walk in both directions along sidewalks. When two pedestrians come too close, they adjust their positions to avoid collisions.
- **Lane Change under Congestion** In cases where a lane becomes congested, some vehicles are allowed to switch to adjacent lanes to maintain traffic flow.

Figure 9: An example of generated character and daily schedule.

D Robot Simulation

In the Community Robot Challenge, we employ a robot carrier and a mobile manipulator. Although Virtual Community also supports other robot types—including quadruped and humanoid robots—and can readily accommodate any robot platform thanks to its Genesis foundation, in this section we describe the simulation details for the four default robot types.

D.1 Mobile Manipulator

We adopt the Google Robot from the MuJoCo library and integrate it into Genesis as the default mobile manipulator. Following AI2-THOR [31], Habitat [49], and ManiSkill3 [62], we add one joint to control forward/backward motion and another joint to enable rotation about the z-axis at the base.

Figure 10: An example of simulating quadruped and humanoid robots in Virtual Community.

D.2 Quadruped and Humanoid Robots

The Unitree Go2 serves as our default quadruped, and the Unitree H1 as our default humanoid, shown in Figure 10. We utilize the according URDF file supported by Genesis [4], together with reinforcement-learning-trained locomotion policies provided by the Genesis team.

D.3 Robot Carrier

We use the Husky robot as the default carrier, importing its URDF file from Bullet[8]. The carrier has four degrees of freedom—one per wheel. We have modified its top surface so that any object landing on it is automatically attached to the carrier.

[8] https://github.com/bulletphysics/bullet3

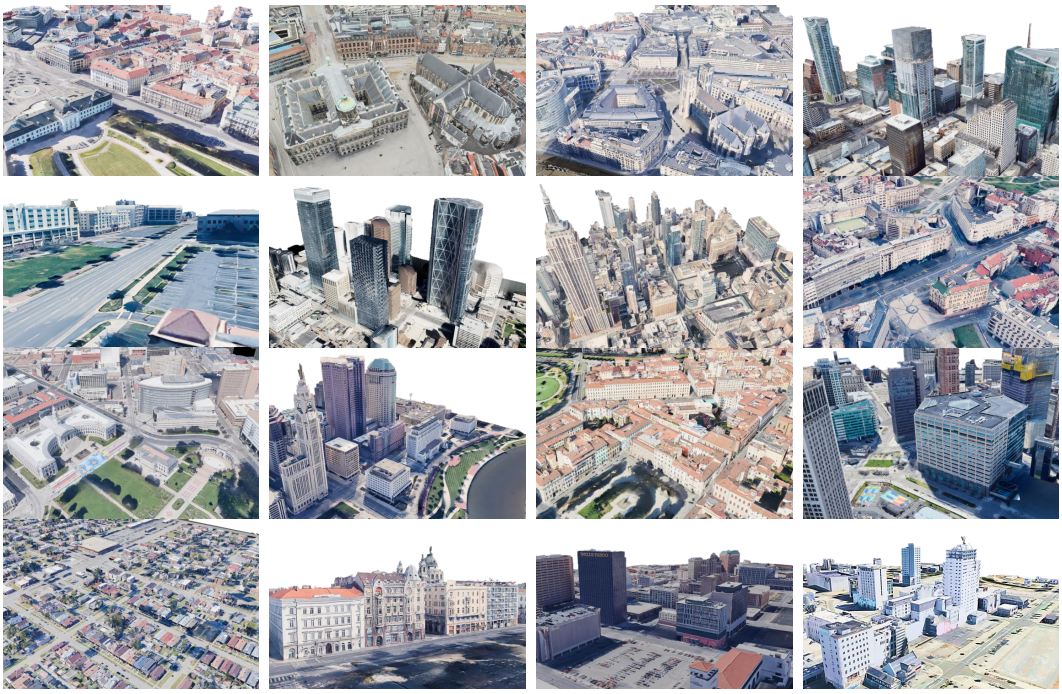


Figure 11: Large-scale scene rendered from the generated scenes in North America, Europe, and Oceania. Our method can generate high-quality scenes with an area of square miles.

Figure 12: Close-up view of the generated scenes. The resulting scene has clean geometry and realistic texture, which is essential for physical simulation.

E Benchmarking

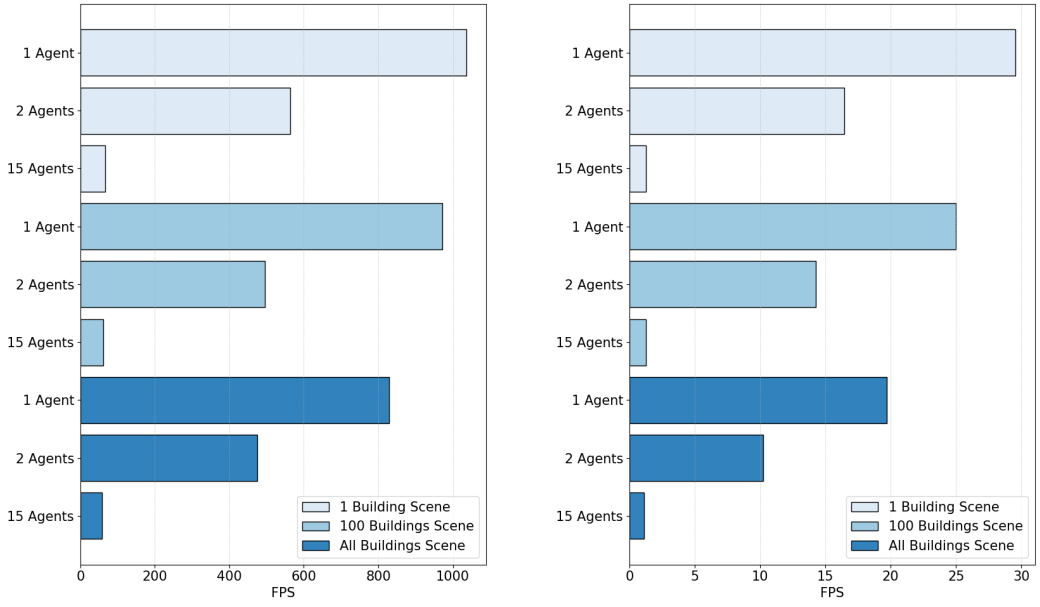
E.1 Scene Benchmarking

In this section, we provide both quantitative and qualitative summaries of the scenes generated for the Virtual Community. Currently, all outdoor 3D scenes have a size of 800m $\times$ 800m. We generated 35 diverse scenes covering 17 countries from North America, Europe, and Oceania. We show some of the generation results in Figure 11 and Figure 12.

To assess the quantitative quality of our generated scenes, we compare them with the original Google 3D Tiles data along two dimensions: visual fidelity and geometric complexity. Visual fidelity directly impacts the ego-view observations received by agents, whereas geometric complexity affects the difficulty of physics simulations. For visual fidelity, we compute the Fréchet Inception Distance (**FID**) [25] and Kernel Inception Distance (**KID**) [6] between our generated scenes (and the baseline scenes) and Google Street View images captured at identical camera positions and orientations. For geometric complexity, we measure the average number of mesh faces per scene—excluding roof faces, which are not involved in the physics simulation—and compare these values between our scenes and the baseline. We rendered 31k images per method to evaluate visual fidelity, and utilize the 3D meshes of all 35 generated scenes to measure geometric complexity. According to the results in Table 2, our generated scenes has significantly improved the visual effects and reduced the geometric complexity compared with the origianl data.

Table 2: We evaluate the generated scenes using Fréchet Inception Distance (FID) and Kernel Inception Distance (KID) for visual fidelity, and the average mesh face count for geometric complexity.

Methods	FID↓	KID ($\times 10^{-2}$)↓	Face Num. ($\times 10^{5}$)↓
3D Tiles	108.04	8.88 ± 0.66	20.94
Ours	**83.65**	**7.60 ± 0.63**	**3.76**



(a) RGB Setting Benchmarking Results

(b) Depth Setting Benchmarking Results

Figure 13: Simulation Speed Benchmarking under RGB and Depth Settings

E.2 Simulation Benchmarking

We benchmark the simulation speed of Virtual Community with the following settings:

- **RGB Setting**: The simulator provides avatar observations, including RGB signals, at 1 Hz, which encompasses 100 physics frames per second. We record the average physics frames per second (FPS) in this setting.
- **Depth Setting**: Similar to Habitat 3.0 [44], we adopt a depth-only configuration that renders a depth image at each physics step (at 100 Hz). In this setting, we also record the average physics frames per second (FPS).

We ran all experiments using a single process on an NVIDIA A100 GPU. Figure 13 presents the benchmarking results under various conditions. In each experiment, we load a fixed terrain background while varying the number of buildings. We also evaluate simulation speed as a function of the number of simulated avatar agents.

F Challenge Details

F.1 Avatar Observation Space

Virtual Community provides each agent with the following observations at each frame:

- **RGB**: Visual input with dimensions of 256×256 and 3 channels.
- **Depth**: Depth information represented as a 256×256 single-channel map.
- **Segmentation**: Segmentation information represented as a 256×256 single-channel map.
- **Pose**: A 6D vector containing the 3D location and a 3D facing vector in ENU coordinates.
- **Camera Extrinsics**: Parameters defining the camera's position and orientation.
- **Events**: Text messages sent from nearby agents using the *communicate* action.
- **Other States**: Includes current cash, accessible locations, and action status.

G Single Agent Task

Task Definition How to leverage public transit in a community to plan the daily commute route to save the most time and energy is a fundamental but also challenging task. We introduce the *community commute* task to study this open-world planning and navigation capability of embodied agents. In this task, an agent needs to commute between 4 - 8 different places given a daily schedule covering 2.5 km of routes on average. The agent can utilize available transit options, including buses with fixed routes and rental bikes along the roads. The bus is only available at several bus stops and the agent can only take a bus when the bus arrives. The bikes are available at given bike stations, and the agent also needs to return the bike to any bike station before the task finishes.

The *Community Commute* task covers 10 different daily schedules in each of 10 different scenes, making 100 test episodes in total. For each episode, we assess the results of all commutes in their daily plan over a single day. On average, each agent completes 5.5 commutes per episode.

G.1 Observation Spaces

Virtual Community provides each agent with the following observations at each frame:

- **RGB**: Visual input with dimensions of 256×256 and 3 channels.
- **Depth**: Depth information represented as a 256×256 single-channel map.
- **Pose**: A 6D vector containing the 3D location and a 3D facing vector in ENU coordinates.
- **Camera Extrinsics**: Parameters defining the camera's position and orientation.
- **Events**: Text messages sent from nearby agents using the *communicate* action.
- **Other States**: Includes current cash, accessible locations, and action status.

G.2 Action Spaces

For the *Commute* task, we restrict the action space to the following:

- **Walk**: Move forward by any distance.
- **Turn**: Rotate left or right by any angle.
- **Enter bus**: Board a bus. Upon execution, the agent is moved inside the bus and parented to it.
- **Exit bus**: Leave a bus. Upon execution, the agent is moved outside the bus and unparented from it.
- **Enter bike**: Mount and start riding a bike. Once executed, the agent's *walk* and *turn* actions are replaced with corresponding bike-riding actions.
- **Exit bike**: Dismount the bike and return to the ground.

Metrics A good commute plan should cost the least time, money, and energy, and avoid missing the schedule. We design the following metric for thorough evaluation.

• **Travel Time**: The average travel time in minutes taken on the route to finish a day's commute.

• **Travel Price**: The average cost for a day's commute.

• **Walk Distance**: The average distance in kilometers an agent walked in a day's commute, measures the energy cost.

• **Late Rate**: Percentage of commute that fails to arrive at the destination in time, measures the method's robustness.

G.3 Baselines

Rule-based Agent ignores the public transit options and always walks directly toward the target location on foot, representing traditional navigation agents.

LLM Agent converts the task information into a prompt and queries Large Langauge Model (we use Llama-3.1-8B-Instruct [19], Qwen-2.5-7B-Instruct [28], and GPT-4o [1]) to generate a commute plan directly, which may include multiple steps such as walking to a bus stop, taking the bus to a specific stop, and then walk to the final destination.

MCTS-based Planner MCTS Planner is based on Monte Carlo Tree Search and simulates different decisions. In our MCTS implementation, transitions from a parent node to its child represent high-level decisions, including:

- **Walking**: Moving to an adjacent position on the map.
- **Taking a bus**: Traveling to any of the N_{bus} bus stops from the nearest bus stop.
- **Taking a bike**: Traveling to any of the N_{bike} bike stations from the nearest bike station.

RL Planner In this section, we present the experimental results of reinforcement learning (RL) models in the *Commute* task.

We trained two RL models using PPO [50] and A2C [38]. The RL-based agents share the same observation and action spaces as described in Section F. The cumulative reward is designed as the sum of the following terms

- For each goal place reached, add 1000 to the reward
- Add the difference $d_0 - d_t$ to the reward, where d_0 is the initial distance to the goal place and d_t is the current distance.
- For each step taken while walking, add -1 to the reward. This encourages agents to opt for public transit system instead.
- For every unit of cash spent, add -1 to the reward.

- For each action performed, add -0.1 to the reward.

For both PPO and A2C algorithms, we set learning rates to 3×10^{-4} and trained for 10^6 steps.

Each parent node can have up to $1 + N_{\text{bus}} + N_{\text{bike}}$ child nodes, where 1 corresponds to walking, N_{bus} to bus stops, and N_{bike} to bike stations. This structure balances connectivity and the exploration of diverse transportation options.

In our MCTS framework, we use Upper Confidence Bound 1 (UCB1) for node selection. For simulation, the reward for each node is designed to evaluate the agent's progress towards the target while considering the total travel cost. Given the following parameters:

- $v_{\text{walk}}, v_{\text{bike}}, v_{\text{bus}}$: Estimated speeds for walking, biking, and taking the bus, respectively,
- d_{target}: Euclidean distance from the current position to the target,
- $d_{\text{walk}}, d_{\text{bike}}, d_{\text{bus}}$: Total Euclidean distances traveled by walking, biking, and using the bus from the root node to the current node.

The simulated reward R for a node is defined as:

$$R = -\left(\frac{d_{\text{walk}}}{v_{\text{walk}}} + \frac{d_{\text{bike}}}{v_{\text{bike}}} + \frac{d_{\text{bus}}}{v_{\text{bus}}}\right) - \alpha \cdot d_{\text{target}},$$

where the parameter α controls the trade-off between exploring closer nodes and exploiting paths with lower travel time. In our experiments, we take $\alpha = 1$.

Notably, unlike baseline agents described in the main paper, RL agents does not rely on a hierarchical decision-making framework. Instead, RL planners directly process observations from the environment to select an action from the action space, differentiating them from high-level decision-making planners such as MCTS and LLMs.

G.4 Results

Methods	Travel Time↓	Travel Price↓	Walk Distance↓	Late Rate↓
Rule	**41.68**	**0.00**	2.44	**10.43**
MCTS	54.33	7.50	1.62	20.24
LLMs				
Qwen	99.67	26.04	2.52	58.88
Llama	66.74	0.82	1.25	33.98
GPT-4o	78.20	20.68	**1.19**	35.72
RL				
PPO	81.96	1.29	4.03	43.50
A2C	97.23	1.66	3.36	44.54

Table 3: **Experiment results of *Commute* challenge.** All metrics are averaged across 10 characters and 10 scenes.

As shown in Table 3, The simplest rule-based agent demonstrates the best performance in terms of travel time, cost, and robustness, achieving the smallest late rate. However, it consumes nearly twice as much energy as the LLM agent powered by GPT-4o when considering walking distance. Both the traditional planning approach using MCTS agents and the advanced foundation model-based LLM agent struggle to effectively utilize the available public transit options. This inefficiency leads to longer commute times and higher late rates compared to the straightforward rule-based agent. Notably, while the LLM agent leverages public transit more frequently, its inability to accurately estimate the time required to reach transit stations—due to uncertainty in navigation—results in significantly increased commuting time. Similarly, planning-based methods fail under the complexity of predicting whether the agent can catch a bus, particularly when working with partially built maps of the environment. RL agents exibit overall longer travel times and higher late rates compared to other agents with low-level navigation planners. Futhermore, RL agents also fail to leverage the public transit system effectively, resulting in relatively lower travel costs but greater walk distances compared to LLM-based agents.

H Computational Resources

The computational cost of our experiments varies across different scenes. Most scene-generation experiments require approximately four hours and 50 GB of GPU memory on a single GPU, excluding the inpainting and upscaling steps, which can be parallelized. For the simulation experiments, we use a single NVIDIA A100, H100, A40, or L40S GPU for each run; memory usage remains below 10 GB for most scenes. Detailed runtime benchmarks are analyzed in Section E.2.

I Prompts for LLM Planners

We use the following prompt template in the daily plan generation for agents:

> Given my character description and known places, please help me plan tomorrow's schedule.
> My Character Description:
> $Character$
> List of places I know:
> $Places$
> Schedule format: The output should be a JSON object which is an array of activities for the character. Each activity should follow the following format:
> "type": "activity type, should be one of the following: 'commute', 'meal', 'sleep', 'main'",
> "activity": "activity description",
> "place": "name of the place where the activity takes place, should be in the list of the known places. Should be null for commute activities",
> "building": "name of the building the activity place belongs to, should be consistent as in the list of known places. Should be null for commute activities",
> "start_time": "HH:MM:SS",
> "end_time": "HH:MM:SS",
> Note: The schedule should be planned based on the character's description and known places. The place should be mentioned for each activity and must be included in the known places. Do not hallucinate places. Commute activities should be given enough time to finish and be inserted between all consecutive activities that do not share the same building so the agent can have time to commute to the correct building before the start of the activity, including commute to meal places. The schedule should start at 00:00:00 and end at 23:59:59, and covering the consecutive time of 24 hours with no gaps. The schedule should be planned in a way that the character can complete all the activities within the given time frame.
> Tomorrow is $Date$. My full schedule for tomorrow:

We use the following prompt template for LLM-based agent in the *commute* task.

> Given my character description, current time and schedule, and known transit system info, please help me make the best commute plan.
> My Character Description:
> $Character$
> Current Time:
> $Time$
> Current Schedule:
> $Schedule$
> Known Transit System Info:
> $Transit$
> Estimated Walking Time from Me to My Goal: $EstimatedTime1$
> Estimated Walking Time from Me to Each Transit Stop:
> $EstimatedTime2$
> Estimated Walking Time from Each Transit Stop to My Goal:
> $EstimatedTime3$
> Note: There are three types of transit options: take a bus, rent a shared bike, or walk. Each option comes with different time and cost. Shared bike has to be rented and returned at

bicycle stations. Please help me choose the best option based on my situation. Output the commute plan as a JSON array where each item is a step in the commute plan with the following format:
"goal_place": "name of the place where the character wants to go, could be a transit stop or a destination", "transit_type": "type of transit, could be 'bus', 'bike', or 'walk'"
Commute Plan:

We use the following prompt template for candidate agents to selection the next target voter in the *campaign* task.

Given my character description and all voter's information, as a election candidate, help me choose which voter to sway first.
My Character Description:
$Character$
Current Time:
$Time$
I'm now at this place:
$Place$
Voter List:
$Voters$
When the distance between a voter and me is lesser than 2, I can directly talk to him without moving. Also make sure that when I reach one voter, he or she should not be commuting. Now I can only choose $limit$ voters among them, please answer with one of the voters' name that I should sway first now, so that I can sway as many voters as possible to my side. Do not include other words.
The voter's name:

We use the following prompt template for candidate agents to generate the first message when communicating with the target voter in the *campaign* task.

Given my character description and a voter's information, as a election candidate, help me convince him or her to my side.
My Character Description:
$Character$
Voter Description:
$Voter$
Please answer with what I should say to convince him or her in natural language.
Hello!

We use the following prompt template for candidate agents to generate dialogue starting from the second message when communicating with the target voter in the *campaign* task.

Context:
I'm an election candidate trying to persuade a voter to vote for me in an election campaign. We are now in a conversation.
My Character Description:
$Character$
The Voter's Character Description:
$Voter$
Dialog History:
$Dialog$
Please answer with what I should say next to convince him or her to my side in natural language. Don't include other words.

We use the following prompt template for voter agents to communicate with the candidates in the *campaign* task.

Context:
An election candidate is trying to persuade me to vote for him or her in an election campaign.
$Additional$
My Character Description:
$Character$
Dialog History:
$Dialog$
Please answer with what I should say next to make better choice about whether I should vote for him in natural language. Don't include other words.

We use the following prompt template for voter agents to vote in the *campaign* task.

Given my character description, candidate list, and interaction history, please help me decide one candidate to vote for.
My Character Description:
$Character$
Candidate List:
$Candidates$
Interaction history:
$Interaction$
Please help me decide one candidate to vote for. Output a JSON object with the following format:
"candidate": "name of the candidate", "reason": "reason for choosing this candidate"

We use the following prompt template to generate tasks.

Given my information, the map information, and object library, please help me propose a task. I will have an agent help me complete this task.
$Info$
Below is the information about this task.
Task type: $TaskType$
Task description: $TaskDescription$
Note that you should make the task as reasonable as possible. For example, if I am going to commute from one place to another, then the 'source' and 'destination' in the json should be in accordance with my schedule.
Your answer should be in a json format like the following:
{
__json_dict__
}
Now give the json output. Don't include any other words. Especially don't include anything like "```json".

For LLM-based single agent baseline in the Community Assistant Tasks, we use the following prompt templates.

I'm $NAME$, an assistive robot in a virtual community designed to help people with daily tasks. I have six tasks to complete before $ENDTIME$. Given the tasks, location information, current state, and my previous actions, help me select the best available action to complete all tasks as efficiently as possible.
Tasks:
$TASKS$
Location Information:
$LOCATION_INFORMATION$
Current State:
$STATE$

Previous Actions:
$PREVIOUS_ACTIONS$
Available Actions:
$AVAILABLE_ACTIONS$
Constraints and Strategy: Avoid searching for objects in regions and prioritize direct targets like a specific place or agent. Avoid searching for objects in the same region for more than 5 minutes. After completing a task, action with {'type': 'task_complete', 'arg1': 'i'} immediately. I have two arms, but each arm can only hold one object. I can only pick up an object if that arm is free. Before moving to a task destination, ensure that any required objects are already picked up. I should be holding the target objects before starting the following task. Tasks do not have to be completed in order, and completing part of a task is allowed. Focus on actions that maximize overall task progress and completion within the time limit.
Output a JSON object with the following format:
{
"action": "full dictionary of the best available actions",
"reason": "Explain why this action is the best choice given the context."
}

For CoELA, we used the following prompt templates for the planning module:

I'm $NAME$, an assistive robot in a virtual community designed to help people with daily tasks. There's another assistive robot $OPPO_NAME$ in the community, I need to cooperate with him to get tasks done efficiently. We have six tasks to complete before $ENDTIME$. Given the tasks, location information, current state, my previous actions and our conversation history, help me select the best available action to complete all tasks as efficiently as possible.
Tasks:
$TASKS$
Location Information:
$LOCATION_INFORMATION$
Current State:
$STATE$
Previous Actions:
$PREVIOUS_ACTIONS$
Conversation History:
$Conversation_History$
Available Actions:
$AVAILABLE_ACTIONS$
Constraints and Strategy: Avoid searching for objects in regions and prioritize direct targets like a specific place or agent. Avoid searching for objects in the same region for more than 5 minutes. After completing a task, action with 'type': 'task_complete', 'arg1': 'i' immediately. I have two arms, but each arm can only hold one object. I can only pick up an object if that arm is free. Before moving to a task destination, ensure that any required objects are already picked up. I should be holding the target objects before starting the following task. Tasks do not have to be completed in order, and completing part of a task is allowed. Focus on actions that maximize overall task progress and completion within the time limit.
Output a JSON object with the following format:
{
"action": "full dictionary of the best available actions",
"reason": "Explain why this action is the best choice given the context."
}

For CoELA, we used the following prompt templates for the communication module:

I'm $NAME$, an assistive robot in a virtual community designed to help people with daily tasks. There's another assistive robot $OPPO_NAME$ in the community, I need to cooperate with him to get tasks done efficiently. We have six tasks to complete before $ENDTIME$.

Given the tasks, location information, current state, my previous actions and progress and our conversation history, help me generate a short message to send to $OPPO_NAME$ to complete all tasks as efficiently as possible.
Tasks:
$TASKS$
Location Information:
$LOCATION_INFORMATION$
Current State:
$STATE$
Previous Actions:
$PREVIOUS_ACTIONS$
Conversation History:
$Conversation_History$
Progress:
$PROGRESS$
Constraints and Strategy: I should communicate with $OPPO_NAME$ to coordinate our actions and share information about the tasks and objects. Searching for objects in regions is much more difficult than direct targets, like a specific place or agent, so it's wise to prioritize easier or more direct targets. If search for a too long time, I will be out of time. I have two arms, but each arm can only hold one object. I can only pick up an object if that arm is free. Before moving to a task destination, ensure that any required objects are already picked up. I should be holding the target objects before starting the following task. Tasks do not have to be completed in order, and completing part of a task is allowed. Focus on actions that maximize overall task progress and completion within the time limit.
Output a JSON object with the following format:
{
"message": "a short message of what I should say to $OPPO_NAME$, null if the conversation should be ended now.",
"reason": "reason for generating this message"
}