BeSAFE v2 - Habitat 3.0 on Meta Quest3 via Unreal Engine.

Haonan Chen haonchen@ethz.ch
Shenghao Zhang zshenghao@ethz.ch
Yingzhe Liu yingzliu@ethz.ch
Tianhao Chen chentia@ethz.ch

Abstract

In this project, we aim to address the limitations of synthetic environments, which often lack realism for robot training. Leveraging Habitat 3.0's [2] features, particularly its capability to navigate through an environment with a Meta Quest 3 headset, we utilize Unreal Engine's stunning visual rendering to create a more immersive and realistic simulation.

Client: https://gitlab.ethz.ch/zshenghao/besafeunrealclient

1. Introduction

In recent years, embodied AI has undergone significant advancements. Unlike internet AI, which primarily processes and utilizes information curated from online sources [3], embodied AI is designed to generate actions for embodied agents, such as robots, operating within physical environments. Achieving this objective necessitates capabilities in perception, long-term planning, motion control, interaction, and related areas.

Embodied AI can be applied to a wide range of fields, including autonomous driving, where vehicles must perceive and navigate complex environments; assistive robotics, which provides support for individuals with disabilities by performing tasks like fetching objects or aiding mobility; and disaster response, where robots operate in hazardous environments to locate survivors or assess structural damages. It also plays a crucial role in warehouse automation, optimizing logistics through efficient movement of goods, and in healthcare, where robots assist in surgeries, patient care, or therapeutic activities. Furthermore, embodied AI contributes to immersive experiences in augmented reality (AR) and virtual reality (VR) systems, enabling realistic interactions within virtual environments.

There are many advanced simulation platforms designed for embodied AI training(e.g. Overcooked-AI [1] designed for coordination with humans with grid-like 2D environments). However, Most synthetic environments used to train robots either lack realism [3] or are very simplistic, such as Issac Sim.

In this project, we aim to address synthetic environment limitations by using Unreal Engine, a professional game engine. Specifically, we immigrate the simulation environment from Habitat simulator to the Unreal Engine, developing the user interface for both personal computers and the VR equipment (Meta Quest 3). Furthermore, We compare the rendering quality of the original simulator and our Unreal program on Meta Quest3. The result shows that the rendered images are more realistic than those rendered on the Habitat simulator. A user study is also implemented to investigate how well the integration of Habitat 3.0 and VR equipment would provide an immersive experience for both scientists and users, facilitating more precise evaluations of robotic interactions and task performance.

2. Background

2.1. Habitat 3.0

Habitat 3.0 is a simulation platform for studying collaborative human-robot tasks in home environments [2].

Human-in-the-loop (HITL) infrastructure Habitat3.0 develops an infrastructure that enables real human interaction with simulated robots via mouse and keyboard input or a Virtual Reality (VR) interface. In this framework, the user can interact with the model-driven robot in a simulated environment and give accurate evaluation while the researchers are able to collect data during that process.

As mentioned in [2], there are multiple advantages of using this HITL infrastructure. For instance, the client part can be implemented on other platforms without worrying much about the performance issue, as most logic is hosted and calculated on the server end. Furthermore, the framework provides functionalities for recording and replaying of HITL episodes to support data reproducibility.

Collaborative tasks Habitat3.0 presents two collaborative human-robot interaction tasks(i.e. *Social Navigation* and *Social Arrangement*), and a suite of baselines for each.

Social Navigation, as shown in 1, refers to the action of a robot finding and following a humanoid while maintaining a safe distance. In social navigation, the robot's objective is independent of the humanoid's goal,

The second task, *Social Rearrangement*, entails collaboration between a robot and a humanoid avatar to organize a set of objects by moving them from their initial positions to designated locations, simulating a house-cleaning scenario.

Particularly, we will use the model pretrained on *Social Navigation* task to guide the robot's movement in our project.

Figure 1. Overview of the *Social Navigation* task. The blue line indicates the human path and the pink one indicates the robot path. The robot path is divided into two phases: *Find Phase* to find the human and *Follow Phase* to avoid collision.

2.2. Unreal Engine

Unreal Engine (UE) is a state-of-the-art game engine widely used for creating real-time 3D applications, including video games, virtual reality experiences, and simulations. Developed by Epic Games, Unreal Engine provides a robust and versatile platform equipped with advanced rendering, physics simulation, and AI tools. Its highly optimized architecture supports photorealistic visual fidelity, dynamic lighting, and intricate environmental detail, making it a preferred choice for academic research in fields such as computer graphics, simulation, and virtual environments. Additionally, its scripting capabilities and Blueprint visual programming system offer researchers and developers a flexible framework for prototyping and implementing complex systems.

In our project, UE program will be the client end of our framework. It plays a role in rendering high-quality images with basic guidance such as instruction interfaces and pickable object highlights.

2.3. Virtual Reality

Virtual Reality (VR) is an advanced technological medium that creates immersive and interactive simulations of environments, allowing users to experience and engage with digital worlds in real time. By leveraging devices such as motion sensors, VR enables the replication of sensory experiences to simulate presence within virtual settings. It also serves as a tool for exploring human perception, behavior, and interaction in controlled and customizable scenarios in academic research.

We adopt Meta Quest 3 as our VR equipment.The Meta Quest 3 is a cutting-edge standalone VR headset developed by Reality Labs of Meta. The headset features advanced mixed reality capabilities, allowing users to seamlessly transition between virtual and real-world environments through high-resolution passthrough technology. Besides, the platform of Meta Quest 3 is fully supported by UE and thus the packaging process is relatively smoother.

3. Method

In this project, we developed a habitat-based application utilizing Unreal Engine, with the aim of delivering a photorealistic simulation environment accessible on both PC and VR platforms. Our contributions are enumerated as follows to ensure a seamless and interactive user experience.

1. Applied a client-server architecture for Unreal Engine and Habitat communication.

2. Immigrated the Habitat scene into Unreal Engine and deployed the application on the Meta Quest 3 device.

3. Explored the rendering ability on Meta Quest Device and made full use of the rendering ability to output a more realistic scene compared to the Habitat simulator.

4. Utilized the power of Habitat for user control and human-robot interaction tasks, including social navigation and random pickup tasks.

3.1. Architecture

The architectural design of our application adheres to a client-server framework, wherein the Unreal Engine serves as the front-end interface, providing environmental simulation through its advanced rendering capabilities. Concurrently, Habitat functions as a backend server, responsible for processing user input and facilitating interaction. This is further augmented by the integration of an embodied AI model that governs the actions of the robot.

The architecture overview is depicted in 2. Unreal Engine is specifically responsible for the task of rendering. The Habitat scene is imported into Unreal Engine to ensure that both the Habitat server and the Unreal Engine client possess identical information about the scene. Upon application initialization, a connection is established between Unreal Engine and Habitat. Concurrently, Habitat transmits creation instructions via a JSON file, detailing all objects within the scene. Upon receipt, Unreal Engine automatically loads the respective assets and positions the objects appropriately within the scene.

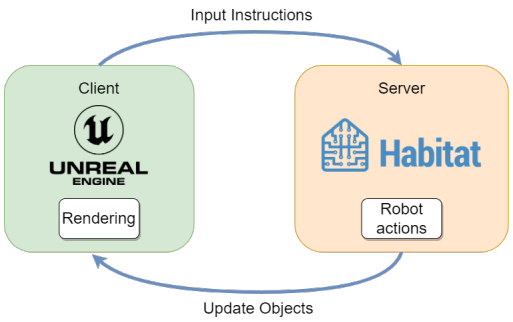

Figure 2. Client-server architecture

During application execution, there is continuous bidirectional communication between the Habitat server and Unreal Engine, occurring at regular intervals. To balance the volume of network requests and the real-time updates, the network frame rate is set to 10 fps, indicating ten instructions are dispatched per second. The messages from Habitat convey transformations of updated objects, enabling Unreal Engine to adjust the positions of relocated items for each frame. Conversely, Unreal Engine receives user inputs either through the keyboard or VR controllers, encoding them into keycodes comprehensible by Habitat. Based on this user input, the server executes the corresponding actions.

3.2. Server

The server side hosts the Habitat 3.0 simulator. It utilizes the HITL infrastructure to allow human-robot interactions in VR and defines collaborative tasks that shape the themes of the interactions. In particular, our application deploys models pretrained on the *Social Navigation* task mentioned in section B.1 and the *Random Pick* task in which the robot wanders randomly around the scene and picks up objects on its way. As it processes all computationally heavy parts, Habitat 3.0 receives user input from the client, updates the scene resulting from human and robot actions, and sends instructions to the client periodically to synchronize the scenes on both sides. The server runs on a machine with the Ubuntu operating system and communicates with the client using websockets.

Processing user input In our application, the user is able to explore an indoor scene, interact with the robot, and throw objects, all done in the Quest 3 headset. For this purpose, the server side receives JSON files containing essential information about the player and controller input to enable this user experience. These include the position as well as the rotation of the headset and the controllers. In addition, lists of controller buttons pressed, held, and released at the time of transmission are there for user interaction with movable objects. An example shows later:

Listing 1. User information and input JSON

```
"avatar": {
    "root": {
        "position": [-1.1,1.3,-4.7],
        "rotation": [0.1,0.8,0.9,0.0]
    },
    "hands": [{
        "position": [-1.0,1.3,-5.1],
        "rotation": [0.1,0.0,0.9,0.0]
    },
    {
        "position": [-0.8,1.3,-5.0],
        "rotation": [0.1,0.0,0.9,0.0]
    }]
},
"input": {
    "buttonHeld":[3],
    "buttonUp":[0,1],
    "buttonDown":[]
}
```

which entails the aforementioned content. Here, buttons 0 and 1 indicate the shoulder button and the side button of the left controller, and buttons 2 and 3 indicate the right controller counterparts. We encountered some problems as the input mapping provided by Habitat contains conflicts: the number keys 0, 1, 2, and 3 also serve as the keys to change scenes. Therefore, we changed the mapping to unused keys and fixed other bugs in order for the input part to run correctly.

The player moves in the scene with the left joystick and looks around with the right stick. Besides these, the player can also move the body freely in the real world, and the transformation of the headset and the controllers are reflected in-game as this comes as a feature of Unreal Engine's VR capabilities. Inside the message, the position and rotation are transformed to Habitat's coordinate system. With all the information, Habitat proceeds to represent the player in the simulator.

Habitat visualizes the player with simple shapes as shown in 3, where the four lines around the human avatar's head are the view port of the VR headset, and the two other lines each represent a controller. Although the avatar exists on the server, it is not visible on the client as the client is in first person. The avatar on the server is there for the robot to recognize.

For the user to grab movable objects, the server first checks in the message whether the grab keys (0, 1, 2, and 3) are pressed. If so, it queries the minimum distance between all movable objects and the pressed controller. If the distance is within a threshold, the corresponding object is successfully grabbed. When the grab button is released, the server applies calculated velocity in the direction of movement of the controller to perform the throwing action of

Figure 3. Player grabbing an object on server

the object. Movable objects are highlighted with circles to guide the player. The user can also grab objects already grabbed by the robot.

3.3. Client

Unreal Engine, a powerful game engine, is designed to offer a photorealistic simulation environment conducive to embodied AI training. In this application, Unreal Engine functions as the client and manages various tasks such as scene migration, scene rendering, user input, and VR deployment. Within the Habitat framework, Unity serves as the standard tool for VR device deployment; however, it does not achieve the same level of realism as Unreal Engine. During development, the Unity application serves as a baseline for our Unreal Engine application.

Scene Migration Habitat provides datasets that include a variety of indoor environments intended for embodied AI training purposes. We have transferred one specific scene from Habitat to Unreal Engine. Within Habitat, scenes are encapsulated as episodes in JSON format, encompassing comprehensive details about all objects present in the scene. As highlighted in the architecture discussion, Habitat dispatches a JSON file to the client during the initialization phase. The structure of this JSON file adheres to the following format,

Listing 2. Scene initialization JSON

```
"creations": [{
    "instanceKey": 0,
    "creation": {
        "filepath": "data/fpss/stages
           /103997424_171030444.glb",
        "isStatic": true,
        "isRGBD": true,
        "isSemantic": true,
        "isTextureSemantic": false,
        "lightSetupKey": "",
        "rigId": -1
    }
}]
```

which contains multiple attributes associated with the scene's items. The Unreal client retrieves the creation JSON to manage it correctly. To be more precise, the client loads the asset using the designated file path and then constructs an actor that matches the given properties. The instance key serves as a mapping index for straightforward actor access. Note that this creation process does not alter the actual transformations of these items. The actual location adjustment occurs during the initial update command, as per the subsequent JSON format,

Listing 3. Scene update JSON

```
"stateUpdates": [{
    "instanceKey": 0,
    "state": {
        "absTransform": {
            "translation": [
                0.0,
                0.0,
                0.0
            ],
            "rotation": [
                1.0,
                0.0,
                0.0,
                0.0
            ]
        },
        "semanticId": 0
    }
}]
```

The update command will attempt to relocate the objects to their appropriate positions and orientations.

During the migration, we face numerous challenging issues. Firstly, Habitat saves all meshes in the GLB format, causing unforeseen complications like missing materials and deployment issues. We resolve this by manually converting the GLB format to FBX, which is fully compatible with Unreal Engine. Secondly, a discrepancy exists between the coordinate systems of Unreal Engine and Habitat. Habitat uses a right-hand coordinate system, where the x-axis is left, the y-axis is up, and the z-axis is backward. Conversely, Unreal Engine operates on a left-hand coordinate system, where the X, Y, Z axes denote forward, left, and upward directions, respectively. To convert the translation and rotation using the correct coordinate system, translation and rotation received from the Habitat server will be

modified with the following rules

$$\text{Translation: } (X, Y, Z) \Rightarrow (-Z, X, Y)$$
$$\text{Rotation: } (X, Y, Z, W) \Rightarrow (Z, -X, Y, W)$$

Receiving User's Input As described in the server section, user inputs are managed by the Habitat server, with the Unreal Engine client responsible for capturing data from keyboards or controllers. Using the Enhanced Input System in Unreal Engine, input keys and buttons are mapped to Habitat keycodes. Messages are periodically dispatched to the server for further processing.

Scene Rendering Unreal Engine's most significant contribution to this application lies in its photorealistic renderer. The integration of a deferred shading pipeline and physically based materials in Unreal Engine enables more realistic rendering results. By default, nearly all lights are absent in the Habitat scene, leaving objects illuminated only by a simple ambient light, resulting in a flat and unconvincing appearance. To improve scene realism, we place additional lights in the scene. Introducing lighting allows us to cast shadows that enhance visual quality. While personal computers can deliver significantly better results, VR device limitations prevent the support of dynamic shadows. Typically, shadows need precomputation before runtime, restricting shadowed objects to a stationary state, which is undesirable because our application updates object positions during execution.

Another VR device challenge is performance. To create a precise simulation environment, Habitat offers a high-resolution scene with highly detailed meshes. Statistically, the scene is made up of over 6 million triangles, making it exceedingly difficult for Meta Quest 3 to fully load all detailed assets during initialization, causing issues such as missing textures and stretched meshes. To address this, we preprocess the assets by reducing mesh faces, cutting the triangle count to around 1 million, which suits our application.

4. Experiment

4.1. Qualitative Results

In this section, we demonstrate that the Meta Quest device's Unreal Engine output surpasses the results of the Habitat simulator. The Habitat simulator's scene is lacking in realistic lighting, while Unreal Engine benefits from more lighting effects. Although not perfect because of VR device compatibility limitations, it outperforms the original Habitat scene, as shown in the demonstration Fig. 4.

(a) Image rendered by habitat simulator.

(b) Image rendered by Unreal on Quest3.

Figure 4. Comparison on rendering quality

4.2. User Study

To further evaluate our results, a user study was conducted during the demo session. Participants with varying levels of familiarity with the system were asked to complete a questionnaire to assess their experience. The questionnaire was designed to evaluate three main aspects: graphics quality, ease of control, and human-robot interaction.

4.2.1 Questionnaire

The questionnaire consists of eight questions:

1. How do you rate the rendering quality in Habitat 3.0 Simulator?
2. How do you rate the rendering quality in Unreal Engine on PC?
3. How do you rate the rendering quality in on VR Headset?
4. How do you rate the easiness of control?
5. How do you rate the smoothness (e.g. frame rate) of this demo on VR?
6. Did you experience motion sickness from playing this demo? If so, what do you think is the main reason that caused the dizziness?
7. Do you agree that it is more immersive to play this demo on VR compared to PC with a monitor? If so, do you think this immersiveness can better help researchers train the robot?
8. Are there any more activities you would like to perform with the robot?

Questions 1–5 used a Likert scale, where "1" indicated "Very Bad" and "5" indicated "Very Good." Question 6 was a binary question with "Yes" or "No" options. If participants selected "Yes," they were asked to elaborate on the reasons for their discomfort. Questions 7 and 8 were open-ended, allowing participants to provide qualitative feedback.

4.2.2 Results

During the demo session, we obtained a total of 7 responses from participants. The results of Questions 1-5 are shown in Table 1.

Question No.	Rating					Avg. / SD
	1	2	3	4	5	
1	0	1	1	1	4	4.14 / 1.12
2	0	0	0	1	5	4.83 / 0.37
3	0	0	0	1	4	4.80 / 0.40
4	0	1	1	1	4	4.14 / 1.12
5	0	0	1	1	3	4.40 / 0.80

Table 1. **Results of Questions 1–5**. This table presents the frequency of ratings (1–5) given by participants for each question, along with the calculated mean (Avg.) and standard deviation (SD).

For Question 6, 3 participants (43%) answered "Yes" to experiencing motion sickness. All of these participants identified the cause as a mismatch between motion in the VR headset and the real world. 4 participants (57%) answered "No" to experiencing motion sickness.

For Question 7, all 7 participants agreed that it is more immersive on VR, and that it can better help researchers train the robot. However, one participant noted that the enhanced immersion is conditional on resolving the motion sickness issue. This highlights the need to address user comfort to fully leverage the advantages of VR in training scenarios.

For Question 8, 3 participants expressed interest in asking the robot to fetch a specific object, and 1 participant suggested the ability to talk with the robot.

4.3. Discussion

The results in Section 4.1, along with the responses to Questions 1–3 of the questionnaire, demonstrate a significant improvement in rendering quality from the Habitat simulator to the Unreal Engine, highlighting the effectiveness of introducing advanced lighting techniques. However, a noticeable decline in rendering quality was observed in VR due to device limitations. To address this, the integration of support for dynamic shadows is recommended for future development. The responses of Question 5 show that most of the participants were satisfied with the smoothness, which indicates that the frame rate on the VR device is sufficient.

In our project, the use of buttons and sticks on the controller led to two significant issues. The first was difficulty in control, as indicated by the responses to Question 4. The second was motion sickness, as highlighted in Question 6, where nearly half of the participants reported experiencing it. Utilizing motion sensors to allow users to control the humanoid through their own movements could address these issues. This approach is more intuitive for users who are not familiar with controllers and may help eliminate the mismatch between motion in VR and the real world, thereby improving the overall user experience.

The desire for more sophisticated interactions with the robot, as highlighted in Question 8, underscores the need for further development. Currently, Habitat 3.0 includes only pretrained models designed for simple social tasks. Expanding the system to train robots capable of handling more complex tasks is an important direction for future work.

5. Conclusion

In this project, we successfully migrated the Habitat 3.0 simulator to Unreal Engine, significantly enhancing the photorealism of synthetic environments for embodied AI training. Additionally, we developed a VR application to provide users with a more immersive experience. While challenges such as VR rendering constraints and user experience issues (e.g., motion sickness) remain, this project highlights the potential for creating more realistic and engaging virtual environments. Future efforts will focus on improving user experience, refining control mechanisms, and extending robot training capabilities to tackle more complex tasks.

References

[1] Micah Carroll, Rohin Shah, Mark K. Ho, Thomas L. Griffiths, Sanjit A. Seshia, Pieter Abbeel, and Anca Dragan. On the utility of learning about humans for human-ai coordination, 2020. 1

[2] Xavi Puig, Eric Undersander, Andrew Szot, Mikael Dallaire Cote, Ruslan Partsey, Jimmy Yang, Ruta Desai, Alexander William Clegg, Michal Hlavac, Tiffany Min, Theo Gervet, Vladimir Vondrus, Vincent-Pierre Berges, John Turner, Oleksandr Maksymets, Zsolt Kira, Mrinal Kalakrishnan, Jitendra Malik, Devendra Singh Chaplot, Unnat Jain, Dhruv Batra, Akshara Rai, and Roozbeh Mottaghi. Habitat 3.0: A co-habitat for humans, avatars and robots, 2023. 1

[3] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A Platform for Embodied AI Research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019. 1