Artificial Intelligence Review
https://doi.org/10.1007/s10462-022-10253-x

# A review of platforms for simulating embodied agents in 3D virtual environments

**Deepti Prit Kaur**[1] · **Narinder Pal Singh**[2] · **Bonny Banerjee**[3]

Accepted: 12 August 2022


**Abstract**
The unprecedented rise in research interest in artificial intelligence (AI) and related areas, such as computer vision, machine learning, robotics, and cognitive science, during the last decade has fuelled the development of software platforms that can simulate embodied agents in 3D virtual environments. A simulator that closely mimics the physics of a real-world environment with embodied agents can allow open-ended experimentation, and can circumvent the need for real-world data collection, which is time-consuming, expensive, and in some cases, impossible without privacy invasion, thereby playing a significant role in progressing AI research. In this article, we review 22 simulation platforms reported in the literature. We classify them based on visual environment and physics. We present a comparison of these simulators based on their properties and functionalities from a user's perspective. While no simulator is better than the others in all respects, a few stand out based on a rubric that encompasses the simulators' properties, functionalities, availability and support. This review will guide users to choose the appropriate simulator for their application and provide a baseline to researchers for developing state-of-the-art simulators.

**Keywords** Simulator · Embodied agent · Virtual environment · Perception · Action · Interaction · Navigation · Metaverse · Human digital twin

✉ Bonny Banerjee
bbnerjee@memphis.edu

Deepti Prit Kaur
deeptiprit.kaur@chitkara.edu.in

Narinder Pal Singh
narinder.singh@chitkara.edu.in

[1] Chitkara University Institute of Engineering and Technology, Chitkara University, Rajpura, Punjab 140401, India

[2] Chitkara School of Art and Design, Chitkara University, Rajpura, Punjab 140401, India

[3] Institute for Intelligent Systems, and Department of Electrical & Computer Engineering, University of Memphis, Memphis, TN 38152, USA

Published online: 10 September 2022

## 1 Introduction

The notion of embodiment has been extensively studied for understanding cognition (e.g., Wilson 2002; Smith and Gasser 2005; Pfeifer et al. 2008; Fischer and Zwaan 2008; Shapiro 2019), and for making technological advances in robotics (e.g., Brooks et al. 1998; Pfeifer et al. 2007; Brooks 2018; Deng et al. 2019), vision (e.g., Baruah and Banerjee 2020a, b; Chaplot et al. 2021; Baruah et al. 2022), speech (e.g., Najnin and Banerjee 2017; Baruah and Banerjee 2022), language (e.g. Das et al. 2018), and other areas in artificial intelligence (AI). Embodied interaction allows an agent to derive supervision from the constraints imposed by the regularities of its real-world physical environment (Brooks et al. 1998; Chaplot et al. 2021).

A necessity in the study of embodied AI is the availability of large amounts of rich data involving human subjects in their natural habitat as opposed to in a controlled setting. As observed by Banerjee et al. (2021), collecting and using such real-world data poses significant challenges, such as risk of confidentiality breach and privacy invasion due to the need to simultaneous record an individual and his environment using audiovisual sensors, low-level data collection and processing issues, limited resolution and coverage of sensors, lack of motivation and discomfort to wear sensors for extended periods of time, and requirement of considerable time and expertise for annotating the recorded data. Unsurprisingly, the availability of audiovisual monitoring data recorded from individuals in their natural habitat is scarce. This bottleneck in AI research has been one of the strongest motivators behind the development of software platforms for simulating embodied agents in 3D virtual environments (see Fig. 1).

**What is a simulator and why is it useful?** A simulation software (or simulator) is a computer program that allows a user to observe the operation of a system, and in many cases interact with the system, without the physical existence of the system. A simulation is often judged by its fidelity to a real-world system. An embodied AI simulator is useful only if it closely mimics the physics of interaction between an embodied agent and its real-world environment. Simulation software platforms have been developed and used extensively in many domains such as aviation, healthcare, warfare, transportation, robotics, sensor networks, Internet of Things, meteorology, and biology. Some software simulators

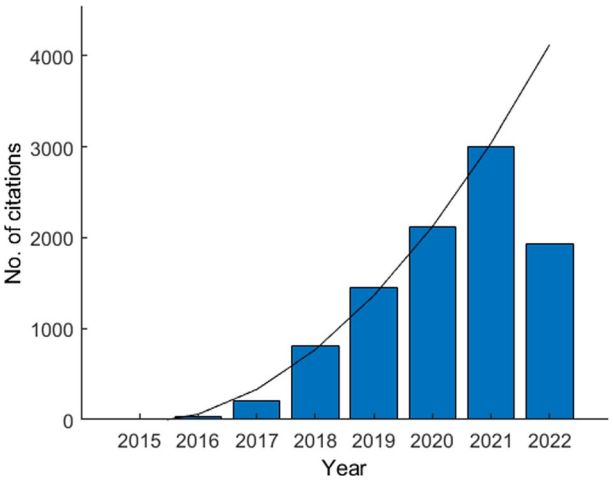


**Fig. 1** Total citation count, as obtained on August 8, 2022 from Google Scholar, during the years 2015 through 2022 for the 22 simulation platforms reviewed in this article. This graph shows an exponential growth in interest in this research area

model the human body in addition to its environment such as patient simulators and workplace environment simulators.

Simulators provide a host of advantages. Simulation is perhaps the only way to verify our understanding of how a system, in this case embodied agents in their environments, work. Simulator code and simulated data can be openly shared and reused. Simulators can generate unlimited amounts of data. Further, simulators can be used for safe and affordable training of AI students, modelers, and data scientists by facilitating simulation of critical situations. In particular, the ability to simulate critical scenarios at will allows efficient and effective training. In the absence of simulators, one would be able to learn how to deal with such scenarios either theoretically or have to wait until such a situation occurred in reality which is not desirable.

In collaborative engineering, simulators facilitate unbiased comparison and benchmarking of progress. Upon successful development and testing through simulations, an approach can be implemented in real-world physical platforms. In embodied agent research, such as mobile robotics, simulators allow policy evaluation with respect to benchmarks for reproducibility and progress monitoring (Savva et al. 2019).

High-fidelity simulation platforms constitute the building blocks for a metaverse. An embodied agent in a metaverse has the potential to be the digital twin of a human, blurring the distinction between virtuality and reality. This will allow experimentation with almost any aspect of life, at micro, mezzo and macro levels, before implementation in the real world. The path towards achieving such goal seems certain with the unprecedented progress in development of embodied agent simulators.

**Contributions of this paper.** In a recent chapter, Nikolenko (2021) reviewed datasets and simulations for outdoor and indoor environments, robotics, unmanned aerial vehicles, and computer games. We review 22 software platforms for simulating embodied agents in 3D virtual environments reported in the literature. Unlike Nikolenko (2021), we classify the simulators based on visual environment and physics, and score them based on a rubric of selected properties and functionalities from a user's perspective. While no simulator is better than the others in all respects, a few stand out. We discuss the uniqueness of each simulator. To the best of our knowledge, this is the first comparison of simulation platforms of its kind, and will benefit both users and researchers alike.

## 2 Background

This section briefly explains a few terms that are used extensively in the field of AI and would play a key role in understanding the rest of this article. Details of each term are available in the corresponding references.

- *Agent.* An entity that exists in an environment, and perceives it using sensors and acts on it using actuators is an agent (Russell and Norvig 2020). This article is concerned only with agents implementable in software. Such agents have been extensively reported in our earlier work (Banerjee and Chandrasekaran 2010a, 2010b; Najnin and Banerjee 2017; Baruah and Banerjee 2020a, b; Baruah et al. 2022) as well as in others'.
- *Agent program* (Russell and Norvig 2020). The perceptual input to an agent at any instant is called a *percept*. The history of everything an agent has ever perceived comprises its *percept sequence*. The mapping from a percept sequence to an action is called

the *agent function*. Internally in an artificial agent, the agent function is implemented by an *agent program*. See Fig. 2.1 in Russell and Norvig (2020).

- *Environment.* The environment of an agent constitutes the part of the world on which the agent can act and from which it can perceive. Formally, an environment is a function that generates sensory signals as a consequence of an agent's actions on the current environment. The environment might be static or dynamic, deterministic or stochastic, fully or partially observable, and single-agent or multiagent. Refer to Sect. 2.3.2 of Russell and Norvig (2020) for discussions on these properties of an environment. In general, an agent's environment includes, in addition to other things, its own body and other agents.
- *Perception.* The mechanism using which an agent interprets its sensory signals from the environment is called *perception* (Han et al. 2016). Formally, perception is a mapping from the raw sensory signals to a distribution over the agent's current states.
- *Action.* The mechanism using which an agent samples and sometimes alters a selected part of its environment is called *action*. Formally, an action may be defined as a mapping from a distribution over the agent's current states to a distribution over the agent's future states.
- *State space.* The set of all states reachable from an initial state by any sequence of actions constitutes the state space of a problem (Russell and Norvig 2020). The sequence of states traversed due to a sequence of actions constitutes a *path* in the state space. The state space may be discrete or continuous. A discrete state space has distinct states (as in chess playing). A continuous state space does not have well-defined boundaries between states (as in taxi driving). A continuous state space is infinite while a discrete one may be finite or infinite.
- *Rational agent.* The field of AI is mostly concerned with rational agents. Given the agent's internal knowledge and its percept sequence at any instant, a rational agent selects an action that maximizes its performance measure in an expected sense (Russell and Norvig 2020).
- *Multiagent interaction.* An environment is multiagent if an agent's behavior maximizes a performance measure whose value is influenced by another agent's behavior (Russell and Norvig 2020). The interaction between agents in such an environment might be of different types, such as competitive, cooperative, collaborative, following, and so on.
- *Agent types.* The principles underlying most intelligent systems are embodied in three types of agents: reflex, goal-based, and utility-based agents (Russell and Norvig 2020).
- *Embodied agent.* An agent that can use its body to sense the environment and execute actions in the environment is called an embodied agent. A stricter definition of embodied agent necessitates that the agent can sense its own body via a mechanism called *proprioception* (Han et al. 2016; Baruah and Banerjee 2020a, b; Baruah et al. 2022). In most cases, the relaxed definition is used and so will be in this article. For the feasibility and importance of embodiment, refer to Wilson (2002), Pfeifer et al. (2008), Brooks (2018) and Shapiro (2019).
- *Sensors.* Based on their function, sensors can be of two types: ones that are part of the agent's body and help the agent to perceive its environment (e.g., cameras as eyes of a robot), and the others that are used to monitor the agent's physical and physiological behavior. The latter class of sensors can be of two types: those that are worn or implanted in the agent's body (a.k.a. wearable), and those that are not worn but present in the agent's environment (a.k.a. ambient).
- *Physics engine.* To enable a realistic experience for a user, a physics engine is adopted to simulate the interaction between an agent and its environment. For example, objects

picked up and tossed should manifest physical motions due to gravity and collision just as in the real world (Song et al. 2008).

- *First- and third-person perspective in virtual environment*. The transposition of users' viewpoint in a virtual environment can be in first- or third-person perspective (Gorisse et al. 2017). It provides a sense of presence and a sense of embodiment towards the virtual body in an environment.
- *Human–computer interaction.* The field of HCI focuses on improving efficiency and effectiveness of interfaces between man and machine through the design of hardware and software. Augmented and virtual reality (AR/VR) experiences are enhanced due to efficient and natural ways for a user to interact with a virtual or real environment (Kim 2015).
- *Contextual interaction.* When the focus is on user interaction allowing creation and exploration of virtual environments, through a particular context of interaction, the system is called *contextually intelligent* (Walczak et al. 2018).
- *Scene graph.* A hierarchical organization of a 3D environment into objects and regions that can be manipulated programmatically (Savva et al. 2019).

## 3 Selection criteria

This section explains the criteria based on which the 22 simulators are selected for review.

### 3.1 Research questions

This review has been conducted to answer the following research questions in the context of utilization of agent simulators for AI applications:

(1) Is it possible to simulate, in a realistic way, embodied agents in 3D environments that can perceive, act (including interact with the environment), reason, and learn? The environment may consist of other agents and human users.
(2) Is it possible to develop a simulator where users can program the agent and configure the environment?

### 3.2 Eligibility criteria

The inclusion and exclusion criteria used in this review are as follows.

#### 3.2.1 Inclusion criteria

(1) Simulators must be able to simulate 3D environment.
(2) Simulators must be open-source with availability of either source code or standalone application.
(3) Simulators developed and implemented during the years 2016 through 2021.
(4) Simulators must be able to simulate agents with the ability to perceive in at least one modality (e.g., visual) and execute at least one type of action (e.g., navigation).
(5) Simulators must be able to simulate embodied agent(s).

(6) Simulators must be able to simulate agent's interaction with its environment or with other agents.

#### 3.2.2 Exclusion criteria

Any system without the availability of downloadable code/software, or simulation results, or a website with user manual is excluded.

### 3.3 Sources of information

The sources of information are the journals and various IEEE and ACM conference proceedings in the areas of AI, agents and multiagent systems, computer vision, and intelligent systems. Also, arXiv preprints in these areas are included (arXiv is a digital open-access collection of electronic preprints, approved for posting after moderation but without peer review).

### 3.4 Search

The search for various research articles was done using the phrase "embodied AI simulators" and related keywords.

### 3.5 Study selection

The publications on embodied AI simulators during the years 2016 through 2021 are selected.

### 3.6 Data collection process

First two authors searched the mentioned databases and identified various simulators as per the inclusion/exclusion criteria mentioned above. The third author reviewed the simulators based on desired parameters.

### 3.7 Quality assessment

It is key to identify and evaluate only high-quality simulators that strive to answer the research questions. A quality assessment was applied by excluding all simulators that did not have publicly-available code/software, simulation results, or a website with user manual.

### 3.8 Summary measures

This review leads to two outcomes:

(1) Classification and taxonomy of the simulators based on visual environment, physics, and learning environment.

(2) Comparison and ranking of the simulators based on their functionalities (as described in their publications and manuals), and availability and support which include availability of downloadable software, instructions stating how to install and use the software, and a contact email for users to report difficulties.

## 4 Comparison of simulation platforms

In this section, 22 simulation platforms are compared based on their functionalities from a user's perspective. The list of selected platforms includes Malmo (Johnson et al. 2016), VizDoom (Kempka et al. 2016), DeepMind Lab (Beattie et al. 2016), CAD$^2$RL (Sadeghi and Levine 2016), HoME (Brodeur et al. 2017), Cognitive Mapper and Planner (CMP; Gupta et al. 2017), CARLA (Dosovitskiy et al. 2017), AI2-THOR (Kolve et al 2017), PyBullet (Caumans and Bai 2017), CHALET (Yan et al. 2018), MINOS (Savva et al. 2018), Matterport3D (Anderson et al. 2018), Gibson (Xia et al. 2018), VirtualHome (Puig et al. 2018), House3D (Wu et al. 2018), Obstacle Tower (Juliani et al. 2019), Habitat (Savva et al. 2019), VRGym (Xie et al. 2019), VRKitchen (Gao et al. 2019), Arena (Song et al. 2019), ThreeDWorld (Gan et al. 2020), and SAPIEN (Xiang et al. 2020). Gazebo (Koenig and Howard 2004) and MuJoCo (Todorov et al. 2012) are also discussed as many of these 22 simulators are built on them. A timeline of these simulators (software or publication) is shown in Fig. 2.

To answer the research questions stated in Sect. 3, the relevant literature on these simulation platforms is reviewed. The simulation software are tested to understand how realistic the physics is when an agent performs a given task, when there is interaction between dynamic objects and the agents, and when there should be contextual interactions in the scene. As software is the most important component of the simulator afforded to the users, the programming language used for development of the platform is included in the review. Also, the shape, structure and flexibility of the agent's body, type of camera (first-person versus third-person perspective) used in the virtual environment, allowance for interaction with a user external to the platform (as is offered in interactive video games), and various standout features in terms of applications are studied. All these 22 platforms can simulate embodied agents but differ with respect to the 3D scene data used, the embodied agents' actions, interactions and tasks they can simulate, and the evaluation protocols followed by them (Savva et al. 2019). The classification of these simulators based on visual environment and simulation physics is shown in Fig. 3.

### 4.1 Classification of simulation platforms based on visual environment

Based on the nature of the visual environments, these platforms are classified into two broad categories. One is based on 3D dataset where the scene data used by these simulators can be from Matterport3D (Chang et al. 2017), Gibson (Xia et al. 2018), SUNCG (Song et al. 2017), Replica (Straub et al. 2019), or S3DIS (Armeni et al. 2016). The other is based on artist-authored environment, which is a configurable photorealistic 3D environment as per the user's specifications, and can be created within the simulators.

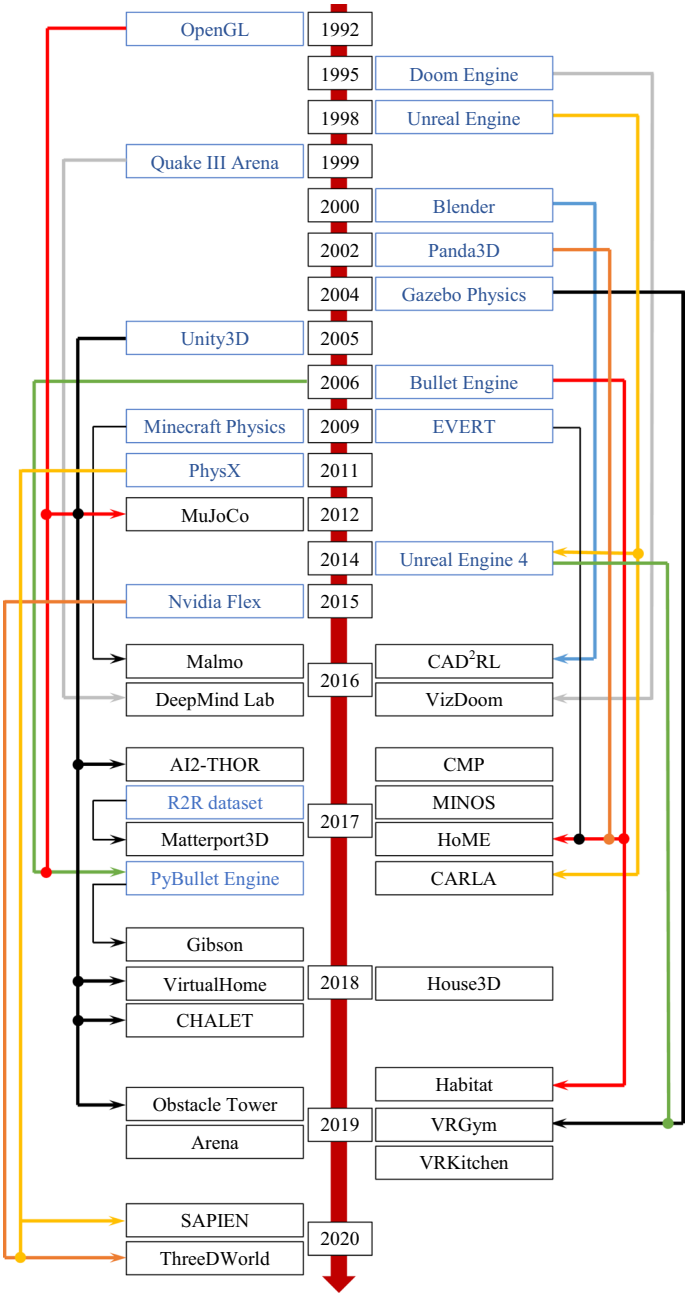


**Fig. 2** Timeline of the release of simulators (shown in black rectangles) and related engines and datasets (shown in blue rectangles). The arrows indicate dependencies, i.e. the system from which an arrow starts is used by a system at which the arrow ends. Best viewed in color. (Color figure online)

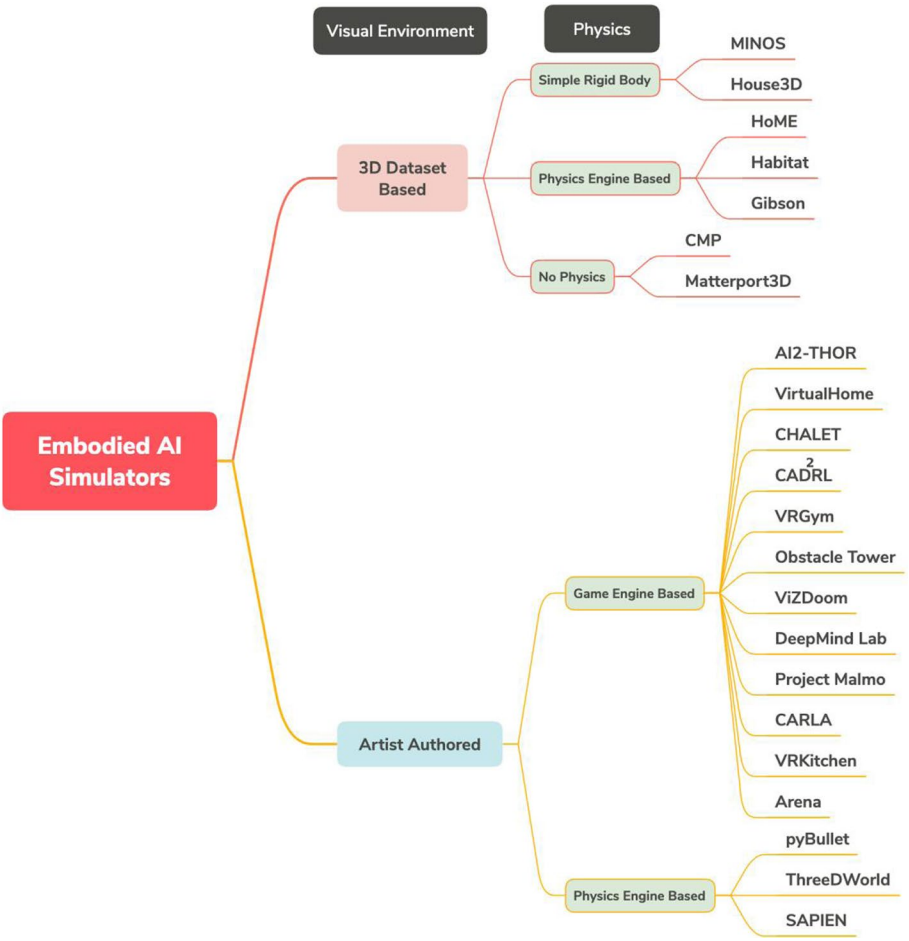


**Fig. 3** A taxonomy of embodied AI simulators based on visual environment and simulation physics

### 4.1.1 Visual environments based on 3D datasets

The advantage of using 3D datasets over synthetic environments is that all observations, such as every coffee mug, pot plant and wallpaper texture (pixel intensities), are realistic and unique as they are generated from natural images of real scenes. This kind of variation and complexity in visual observations is difficult to achieve with a restricted range of 3D objects (assets) and textures. Thus, real-world data retains visual and linguistic richness more than synthetic environments, thereby increasing the likelihood of successfully transferring trained agents to real applications, and supporting the use of 3D datasets for simulating environments (Chang et al. 2017).

Matterport3D (Chang et al. 2017) dataset comprises of 10,800 panoramic views generated from 194,400 RGB-D images of 90 building-scale scenes. The panoramic viewpoints are distributed across the entire walkable floor plan of each scene. The average separation between the viewpoints is 2.25 m. Each panoramic view comprises of 18 RGB-D images captured from a 3D position at the height of an individual. Each image is annotated with

six degrees of freedom camera pose. Collectively, the images capture the sphere except the poles. Textured 3D meshes, globally aligned and annotated with class and instance segmentations of regions (rooms) and objects, are included in this dataset. Habitat, MINOS, Matterport3D and Gibson use this dataset.

The SUNCG dataset (Song et al. 2017) contains realistic manually-created furniture and room layouts of more than 45,000 3D indoor scenes, each semantically annotated at the object level. HoME, House3D and MINOS utilize this dataset.

Gibson's underlying database is collected from real indoor spaces of 572 buildings, using 3D scanning and reconstruction. The buildings consist of 1447 floors over an area of 211 $km^2$. Other than Gibson's own simulator, Habitat uses this dataset.

The Replica dataset (Straub et al. 2019) includes 18 highly photorealistic 3D indoor scene reconstructions comprising of mesh, high-resolution high-dynamic-range (HDR) textures, per-primitive semantic class information, and other features at room and building scale. Replica enables machine learning (ML) research based on visually, geometrically, and semantically realistic generative models of the world. Embodied agents can be developed using Replica to carry out semantic scene segmentation, geometric inference, egocentric vision tasks, navigation, question answering, and instruction following. Habitat utilizes this dataset, where all supported 3D datasets, including synthetic (e.g., SUNCG) and real-world reconstructions (e.g., Matterport3D), are represented in a hierarchical scene graph.

Stanford large-scale 3D Indoor Spaces (S3DIS) dataset (a.k.a. 2D–3D-S dataset) (Armeni et al. 2016) contains over 70,000 RGB images, collected from six large-scale indoor areas covering an area of 6000 $m^2$ in three buildings. The images are associated with geometric and semantic annotations of 3D meshes and point clouds, camera angles, and depth. Information in 2D, 2.5D and 3D modalities is included. This dataset, utilized by the CMP simulator, allows for the creation of joint and cross-modal learning models, as well as unsupervised methods exploiting the regularities in large-scale indoor areas.

#### 4.1.2 Artist-authored visual environments

For artist-authored environments, the scenes are designed manually by 3D artists from reference photos, thus making such simulators free of the bias that exists in automatically generated scenes (Kolve et al. 2017). The environment may also include actionable objects, such as in AI2-THOR, whose states can be altered. In some simulators, such as VirtualHome, sequences of atomic actions and interactions, extractable directly from videos or natural language descriptions, are used in high-level representation of complex tasks.

A challenging domain can be created for training and evaluation of autonomous agents, involving language, vision, and planning for tasks in a dynamic environment. Such environment is used in CHALET. $CAD^2RL$ also uses manually designed collection of 3D indoor environments built using Blender, an open-source 3D modeling suite, forming the basis of a simulated training setup. VRGym provides tasks in realistic scenarios and enables automated data tracking during the execution of a task by agents (humans and robots). In CARLA's simulation platform, environmental conditions and sensors can be specified flexibly, while it supplies open digital assets (urban layouts, buildings, and vehicles).

Obstacle Tower employs a high-fidelity environment comprising of around 100 floors, where an agent starts from the zeroth floor. This 3D environment is generated procedurally and operates in third-person perspective with the provision to recognize and manipulate observations, actions, and reward functions. VizDoom, like many other environments, allows selection of resolution and rendering options.

The rich graphics and lifelike physics of DeepMind Lab comprise a 3D action space useful for generating computer vision datasets. Malmo offers a dynamic, organized and rich environment with which agents interact via a natural sensorimotor loop. Arena provides a general evaluation platform and building toolkit for multiagent intelligence. VRKitchen provides simulation of various cooking activities through human-like agents in a virtual kitchen environment.

PyBullet offers customized environment. ThreeDWorld enables multiple agents to interact in the simulation environment, and also allows humans to interact with virtual objects. SAPIEN offers physics-rich simulated realistic environment for robot–object interactions.

### 4.2 Classification of simulation platforms based on physics

The simulators can be categorized based on their implementation of physics governing the consequences of the agents' actions. Some simulators do not employ any physics for simulating an agent's movements and actions. However, there are simulators with simple rigid body physics governing the agent's motion for collision detection in the simulated environment (Savva et al. 2018). Alternatively, the agent can be subjected to physics and space constraints by integrating the system with a physics engine. The agents are free to perform any mobility task, provided the constraints are satisfied (Xia et al. 2018). The physics can also be implemented using the game engine used for implementing the simulator. In that case, the framework is able to provide application programming interfaces (APIs) for the computation of reward and feedback signals for learning, and extract information regarding the state and location of objects in the environment (Yan et al. 2018). Generally, the methods for rendering 2D and 3D scenes are part of *graphics engines*, such as Open Scene Graph (Wang and Qian 2012). They support graphic drawings on computer display screens, and include implementation of shaders, materials, lights and camera. Such engines are built on top of low-level APIs, such as OpenGL for drawing existing geometry. The physics used in simulators can be further categorized into physics engine-based (e.g., PyBullet, MuJoCo) and game engine-based (e.g., Unity3D, Doom, Unreal).

The objective of a *physics engine* software is to simulate the physical reality of objects, their motions, and the world. The simulation starts with a list of the objects to be simulated and certain configurations (can change physics world gravity, collision settings, physics material properties). Each step advances the simulation by a fraction of a second. The simulation outcomes are used to update the graphics rendered on the screen. A physics engine can assist developers to construct physics-based models and simulations of environmental objects. However, during the process of handling users' inputs (mouse, keyboard, or other kinds of inputs), the developers still need to combine components, such as graphics and physics, in order to complete the simulation. Therefore, system designers have started developing their simulators based on commercial game engines (Kang et al. 2011). Physics engines also contribute towards more realistic motion generation in a simulation environment. Examples of physics engines include PyBullet (Caumans and Bai 2017) and MuJoCo (Todorov et al. 2012) among others.

The software frameworks commonly utilized to develop games are referred to as *game engines*. These game engines usually consist of a graphics engine and a physics engine. The simulators, built on game engines, make use of physics available in the game engine for carrying out physics-related tasks by the agents in the simulation environment. The major utilization of these simulators is done for training the embodied agents for navigation in a 3D virtual environment. Typically, a game engine can support rendering of 2D and 3D


**Fig. 4** CMP environment (Gupta et al. 2017)

**Fig. 5** Matterport3D environment (Anderson et al. 2018)

graphics using graphics engine, and physics simulation and collision detection using physics engine. It also has functions for developing interactions with scripting, making game objects intelligent using AI, networking, threading, memory management, etc. Examples include Unreal Engine (Busby et al. 2010) and Unity3D (Juliani et al 2018) among others.

### 4.2.1 Simulators implementing no physics

The CMP (ref. Figure 4) is built on a unified mapping and planning architecture. Its spatial memory allows it to plan in a partially-observable environment. CMP has been used to perform geometric tasks (positioning of the goal in coordinate frame of robots), semantic tasks (by providing one-hot vector indicating the object category), and visualizations (by analyzing the mapper representation and the value maps learned by their networks).

In Matterport3D (ref. Figure 5), a recurrent neural network-based architecture is used to model an agent for vision and language navigation. Although this simulator trades off between reproducibility, visual realism and interactivity, it is capable of providing a

**Fig. 6** House3D environment (Wu et al. 2018)


**Fig. 7** MINOS environment (Savva et al. 2018)

visually-realistic framework for investigating the Vision-and-Language Navigation (VLN) useful for robotics.

### 4.2.2 Simulators implementing simple rigid body physics

House3D (ref. Figure 6) utilizes simple rigid body physics for collision detection of agents in the simulating environment. A small overhead is added to the rendering due to the default simple physics (Wu et al. 2018). House3D offers 45,622 human-designed 3D scenes of realistic single-room studios to multi-story houses, all equipped with a broad collection of annotated 3D objects, textures and scene layouts, as well as ML methods. It uses concept-driven navigation, and different levels of augmentation for better generalization in the 3D environment. With the simulation of several visual and semantic tasks, agents can navigate through high-level semantic concepts, and the environment resembles the real world in terms of richness of content and structure. Despite providing customizable environment for agent training, it lacks in object interactions and multiagent system support.

MINOS concentrates on navigation-only environments. In MINOS (ref. Figure 7), the agent actions are governed by simple rigid body physics. It employs Matterport3D dataset, thereby inheriting the strengths of Room-to-Room tasks, making its success quantifiable. It can be used to evaluate the ability to follow instructions in natural language for navigation in previously unseen real images at building-scale. MINOS with SUNCG dataset is designed to be a multimodal simulation platform. It supports the development of

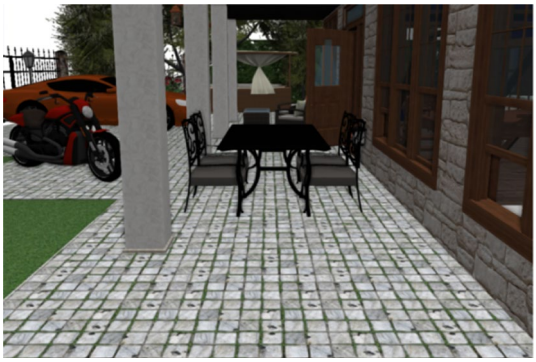

**Fig. 8** HoME environment (Brodeur et al. 2017)


**Fig. 9** Gibson environment (Xia et al. 2018)

multisensory models for achieving goal-directed navigation in customizable indoor environments. However, such environments lack photorealism, object interactions, and support for multiagent systems.

### 4.2.3 Simulators implementing physics using game engine or physics engine

HoME facilitates the simulation of several visual and semantic tasks for agents, and offers diverse customizable environment for agent training. Yet there is lack of object interactions and support for multiagent systems. HoME (ref. Figure 8) employs high-fidelity 3D visual renderings based on Panda3D, and 3D acoustic renderings based on EVERT (Laine et al. 2009). The physics simulation is Bullet-based. It handles collisions, gravity, agent–object interaction, and other functionalities such as learning from multiple sources in a realistic way. It is the first platform to support high-fidelity audio, which allows experimentation on different kinds of tasks. HoME manages acoustic ray-tracing and object-based 3D geometry tracing in real-time, implemented using EVERT. It also allows multiple audio input and output sources. Stereo sounds can be artificially generated from environment audio sources. It supports distance-dependent sound attenuation. HoME semantic engine provides information about properties of objects, such as color, category, material, size, and location. It also provides image segmentation. In this environment, agents can practice teamwork and can communicate with each other to maximize shared reward.

Gibson (ref. Figure 9) uses neural networks for view synthesis, which is useful for training in navigation and locomotion tasks. However, there is no dynamic content in the form

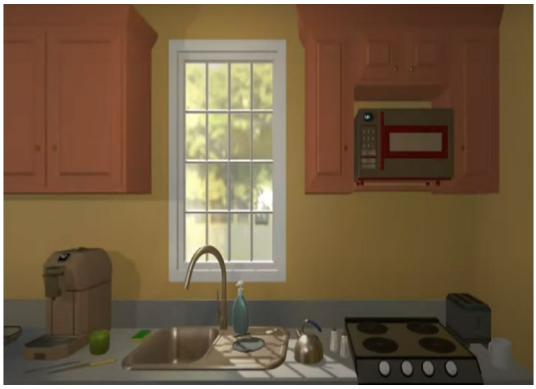

**Fig. 10** AI2-THOR environment (Kolve et al. 2017)

**Fig. 11** VirtualHome environment (Puig et al. 2018)

of additional moving objects or support for manipulation. ROS robotic agents, unified robotic description formats (URDFs), and MuJoCo XML formats can be used in the simulation environment. This simulator is capable of fixing the artifacts and generating a realistic image based on geometric point cloud rendering and physics based on PyBullet engine to support rigid and soft body simulation with discrete and continuous collision detection. Throughout learning, it is critical for an agent to be subjected to space and physics constraints in the form of collision, gravity, and friction. PyBullet's built-in efficient collision handling mechanism is utilized to record an agent's interactions, such as its collision with physical impediments. The Gibson environment has been used for development of real-world perception in active agents. Gibson allows integration with synthetic objects to enable inclusion and manipulation of dynamic content (e.g., other moving objects). However, in the resulting environment, material properties may not align with the physics simulator.

AI2-THOR, VirtualHome and CHALET use first-person perspective for user's viewpoint. VirtualHome can also be customized for third-person perspective. AI2-THOR (ref. Figure 10) provides an environment to improve learning by interaction, planning, vision, question answering, object detection, segmentation, and learning models of cognition. It offers a Python API to interact with Unity3D, a professional game development engine that provides functionalities such as navigation, applying forces, object interaction, and physics modeling. AI2-THOR is unique in that the states of its actionable objects can be altered by the agents. The trained model can be transferred to real robots after fine-tuning.

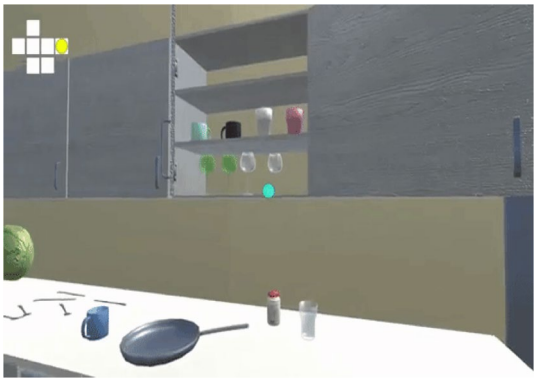

**Fig. 12** CHALET environment (Yan et al. 2018)


**Fig. 13** Obstacle Tower environment (Juliani et al. 2019)

VirtualHome (ref. Figure 11) automatically converts high-level text instructions or video input to programs that "drive" an artificial agent to execute tasks in a simulated household environment. The simulator allows the creation of a large activity video dataset with rich ground-truth, enabling training and testing of video understanding models. In this environment, path finding is built on the NavMeshAgent Unity3d Component (Juliani et al. 2018), with inverse kinematics provided by RootMotion Final IK for realistic motions while interacting with objects. VirtualHome can train agents to perform tasks using vision and reinforcement learning. Using the simulator, one can create large activity video datasets with high-fidelity details, which facilitates the training and testing of video understanding models.

CHALET (ref. Figure 12) allows creation of new houses from a set of rooms, objects, and house configurations. It is implemented in Unity3D. Its simulations are supported by multiple operating systems (Linux, MacOS, Windows, Android, iOS, WebGL). Unity provides a built-in physics engine, and can integrate with AR and VR devices. CHALET specializes in object manipulation and complex task environments, and provides a realistic testbed for processes such as language study for instruction following, visual reasoning, and question answering.

Obstacle Tower environment (ref. Figure 13) utilizes the Unity-based platform and ML-Agents Toolkit (Juliani et al. 2018). It runs on multiple operating systems (Mac, Windows,

**Fig. 14** VRGym environment (Xie et al. 2019)

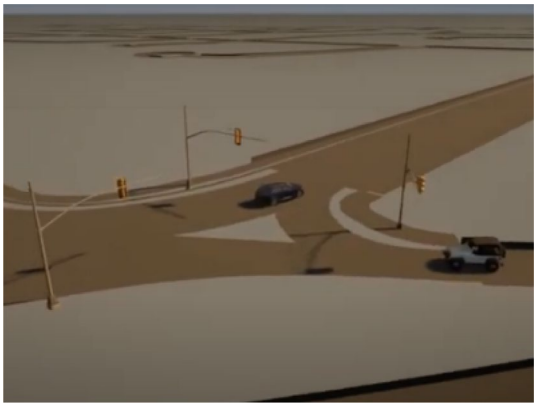

**Fig. 15** CARLA environment (Dosovitskiy et al. 2017)

Linux), and is controlled through the OpenAI Gym interface for easy integration with deep reinforcement learning frameworks (Brockman et al. 2016). Agents in Obstacle Tower learn to control themselves at a low level and plan at a high level for an unseen environment. This environment is built to focus on problems in vision, control, planning, and generalization.

VRGym (ref. Figure 14) uses Unreal Engine 4 (UE4) for real-time physics-based simulations. It is a virtual testbed focused on the development and training of physical and interactive agents used in robotics, ML, and cognitive science. Human–robot interaction is also possible using VRGym, as it can represent human embodiment in the form of a virtual avatar through a variety of hardware setups dedicated for body and manipulation sensing. It uses a VR stereo view for human and robot camera and provides external human interactions and contextual intelligence. It can represent in real-time a physical human agent as an avatar in the simulated virtual environment.

CARLA (CAR Learning to Act) (ref. Figure 15) is used for research, development, training, and validation for autonomous urban driving. It uses two- and four-wheeled vehicles for navigation. CARLA is implemented on UE4 as an open-source layer for flexibility and realism in rendering and physics simulation. It provides state-of-the-art rendering, physical realism, basic non-player character (NPC) logic, and interoperable plugins.

**Fig. 16** VizDoom environment (Kempka et al. 2016)

VizDoom (ref. Figure 16) employs the video-game "Doom", and uses convolutional neural networks. It allows development of bots that can exhibit human-like behavior and play the classical first-person shooter video game using the screen buffer. VizDoom is useful for research in vision-based reinforcement learning.

DeepMind Lab (ref. Figure 17) is used in research and development of AI/ML systems and concentrates on navigation-only environments. It provides high-fidelity 3D visuals involving science fiction and game-like physics. One can create a variety of environments, tasks, and cognitive tests through DeepMind Lab's creative task development.

Habitat's scene graph lends itself nicely to integration with physics engines in order to control the state of objects and agents in the scene graph. Agents in Habitat (ref. Figure 18) are virtual robots capable of working in a self-organized manner and transferring their learned skills to reality. They contribute to a photorealistic and efficient 3D simulation. The simulator offers language understanding to map queries and instructions into actions, and navigation and search in complicated dynamic environments. Instead of low-level control and simulation of robot manipulations, Habitat focuses on high-level navigation and interaction, and facilitates the investigation and implementation of large-scale embodied AI training regimes.

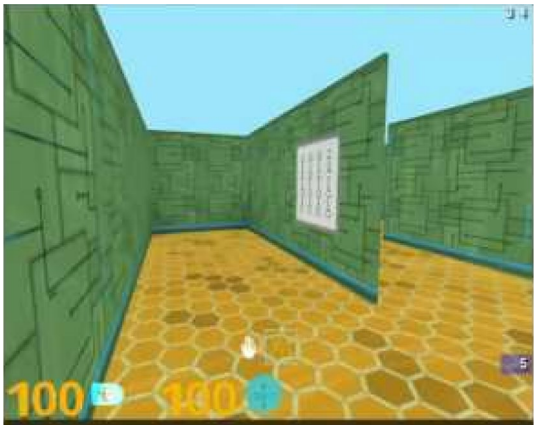


**Fig. 17** DeepMind Lab environment (Beattie et al. 2016)

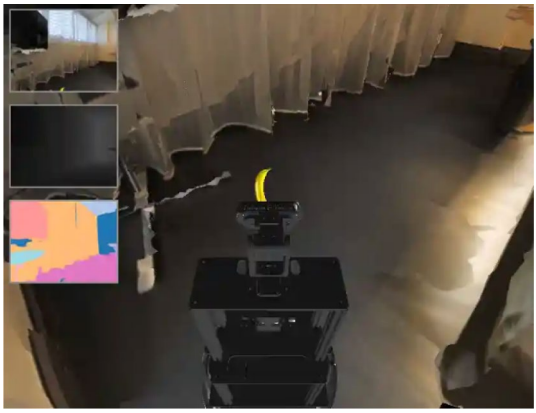

**Fig. 18** Habitat environment (Savva et al. 2019)

**Fig. 19** CAD$^2$RL environment (Sadeghi and Levine 2016)

CAD$^2$RL (ref. Figure 19) is trained using 3D CAD models, and uses first-person viewpoint for the user. It can simulate collision-free flight in real-world indoor environments using reinforcement learning. In CAD$^2$RL, deep neural networks can be trained with simulated monocular RGB images to learn policies for obstacle avoidance and hallway following.

Malmo (ref. Figure 20) provides game-based learning with crafting, building, surviving, etc. Malmo is based on Minecraft game, and is designed to support experimentation in AI, especially artificial general intelligence (AGI). It can support research in robotics, computer vision, reinforcement learning, planning, multiagent systems, and related areas, by providing a rich, structured and dynamic environment. It also supports real-time interactions and multiagent tasks involving humans.

The VRKitchen environment is used to carry out simulations of kitchen activities such as cutting, peeling, and so on. The snapshot in Fig. 21 shows a humanoid agent making a sandwich. VRKitchen supports toolkit implementations for agent training and collecting human demonstrations in AI systems, and will (in the future) include different types of agent learning for evaluations in the simulation environment.

**Fig. 20** Malmo environment (Johnson et al. 2016)

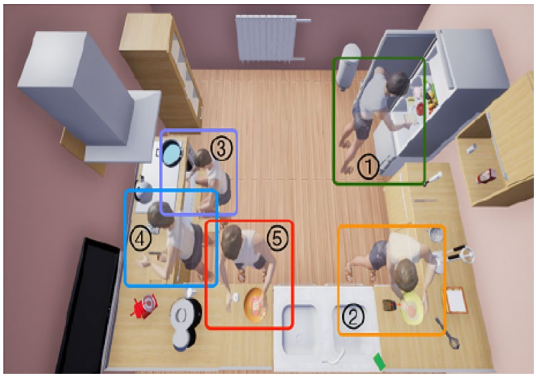


**Fig. 21** VRKitchen environment (Gao et al. 2019)

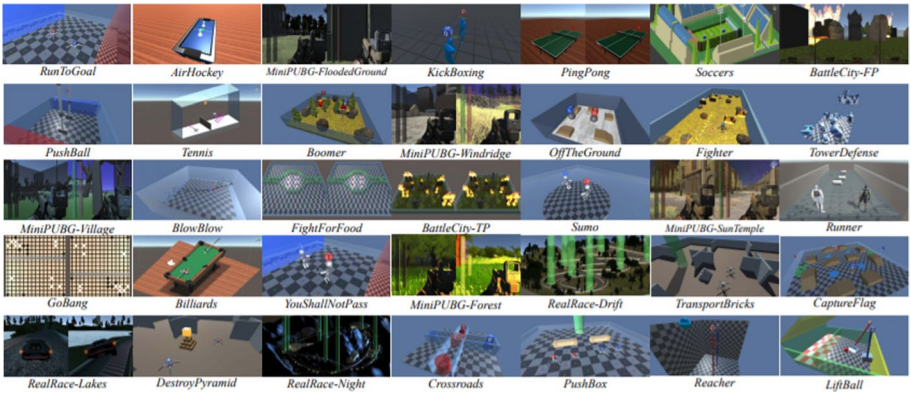


**Fig. 22** Game set of Arena (Song et al. 2019)

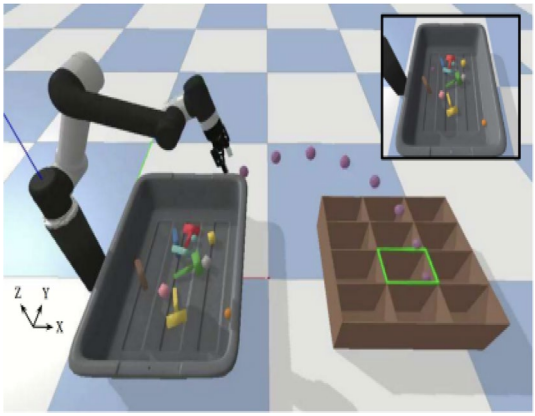


**Fig. 23** PyBullet Environment (Caumans and Bai 2017)

Figure 22 shows the Game set of Arena simulator. Arena provides a building toolkit, and a research and evaluation platform for multiagent systems.

PyBullet, SAPIEN and ThreeDWorld are physics engine-based simulators. PyBullet (ref. Figure 23) is used in robotics and reinforcement learning. It can load articulated bodies from several file formats, such as URDF, simulation description format (SDF), and others. PyBullet is based on Bullet and has built-in functionality for spring-dampers at hinge joints. These engines help the developers to construct models and simulate their operations; yet there is a need to integrate graphics and other components with these physics engines.

SAPIEN (ref. Figure 24) allows tasks related to robot perception and interaction. It provides a unified framework of state-of-the-art physical simulators integrated with modern engines for graphics rendering and user-friendly robot interfaces. However, it lacks in transferable and efficient reinforcement learning approaches.

In ThreeDWorld (ref. Figure 25), human–VR device integrations are available that allows direct observation and manipulation of objects in the simulated environment. It provides a high-fidelity, multimodal platform for interactive physical simulation, and is

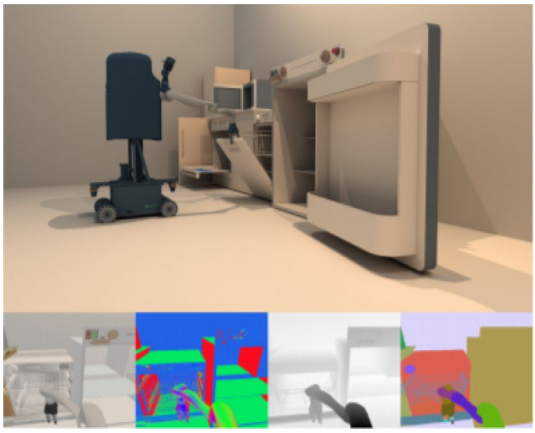

**Fig. 24** Robot–object interaction in SAPIEN (Xiang et al. 2020)

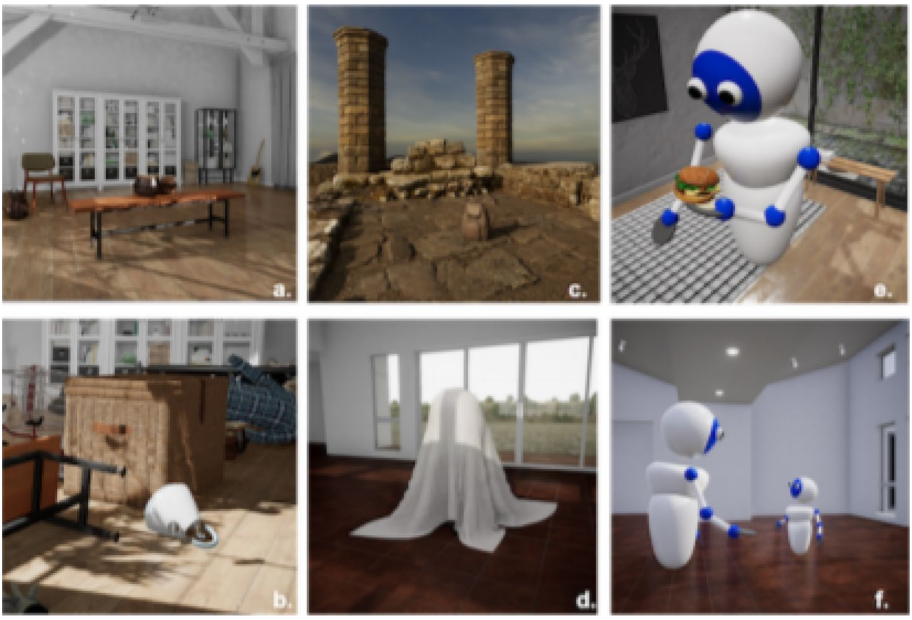


**Fig. 25** ThreeDWorld Environment (Gan et al. 2020)

capable of bridging human and AI behavior. However, only a small number of ThreeDWorld objects are modifiable through user interactions.

Presence of challenging simulation environments has contributed to the rapid pace of AI research. These environments typically offer goal-driven tasks, ranging from simple board games to complex video games. A benefit of these game environments is that they can be utilized to methodically evaluate the performance of AI/ML models in comparison to each other as well as to humans. Learning environments in agent simulation platforms are of two types: task-driven or game-driven.

**4.2.3.1 Task-driven learning** Task-driven learning refers to learning in agents from their experience by performing specified tasks in custom environments. A user can set agents' goals and sequence of actions to achieve those goals within given constraints. AI2-THOR uses target-driven navigation in its simulated environment. CARLA also allows goal-directed navigation tasks for its agents. In CAD$^2$RL, the task is indoor robot navigation. They use two baselines for performance evaluation of the agents: straight controller, and left, right and straight (LRS) controller for collision detection. The performance of agents in CHALET is evaluated by comparing their behaviors with annotated demonstration created using a sequence of states and actions. VRGym and VirtualHome also provide a set of household activities or maze-navigation tasks for training and evaluating the autonomous agents. VRKitchen implements a VR system integrated with AI, and allows human teachers to train the agents using demonstrations. ThreeDWorld is primarily developed for physics simulations, yet it can be used for task-based agent learning. A transport challenge is developed with ThreeDWorld where a robot behaves as an embodied agent and solves complex tasks (ref. http://tdw-transport.csail.mit.edu/). Similarly, PyBullet also allows custom task-based learning, where an agent can learn quadruped locomotion using rewards.

**4.2.3.2 Game-driven learning** In game-driven learning, the environment is game-like and an agent learns from its history to decide the best tactics for winning or survival. In such systems, the agents learn to play obeying the rules of a game within the given environmental constraints. A classic example of game-driven learning for agents is the Atari 2600 game console, which was used by Mnih et al. (2015) to test deep Q-networks and outperformed human expert players. Obstacle Tower, DeepMind Lab, Malmo, Arena, and VizDoom utilize game-driven learning for the agents in the simulation environment. These simulators are inspired by the Arcade Learning Environment based on Atari 2600, one of the most extensively used benchmarks for reinforcement learning. An agent in Obstacle Tower is evaluated based on its performance in unknown instances of the environment. In VizDoom, the game can wait for the bot's decisions, or learning in agents can occur by observing a human playing. The rewards are customizable. For DeepMind Lab, there are tasks such as maze navigation, collecting fruits, traversing difficult passages and avoiding falling off the cliffs, moving between platforms by bouncing through space using launch pads, and various neuroscience-inspired tasks. All these simulators support pixels-to-actions autonomous learning agents.

In task-driven environments, evaluation strategy is decided by the user. In game environments, agent evaluation is based on the rules of the game, agent's achievements, and number of completed levels. Figure 26 shows the categorization of simulation platforms based on agent learning environment.

A comparison of simulators developed using 3D dataset for visual environment is presented in Table 1. Table 2 presents a comparison of the game engine-based simulation platforms for artist-authored visual environment. Table 3 presents a comparison of physics engine-based simulation platforms. All three tables use the same set of criteria for comparison.

MuJoCo (Todorov et al. 2012) and Gazebo (Koenig and Howard 2004) are physics and robot simulators respectively, used by many embodied AI simulators as their basis. MuJoCo (ref. Figure 27) is utilized for testing, implementation, and validation of control mechanisms, which are deployed on physical robots thereafter. C, C++, and Python have been used for programming the environment for MuJoCo. It has a built-in implementation of hinge proportional–derivative controller that uses implicit damping. MuJoCo is distinguished by the fact that its primitive joint types (slide, hinge, ball, free joint) may be composed into more complicated joints without the need for intermediate dummy bodies.

Gazebo (ref. Figure 28) is based on Gazebo physics engine. It can integrate with robot operating system that supports Python and C++. It has a configurable environment with the functionality to recreate complex worlds. A dynamic 3D multi-robot environment can be created. Gazebo uses the Open Dynamics Engine (www.ode.org), a widely used physics engine for simulating the dynamics and kinematics observed in articulated rigid bodies.

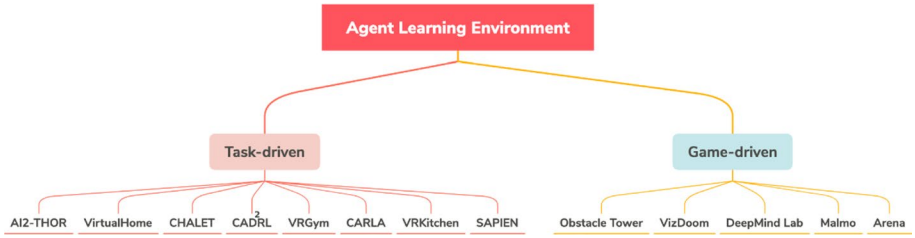


**Fig. 26** Classification of artist-authored simulators based on agent learning environment

**Table 1** A comparison of simulation platforms that are based on 3D dataset environment

| Simulator | HoME (Brodeur et al. 2017) | Habitat (Savva et al. 2019) | MINOS (Savva et al. 2018) | House3D (Wu et al. 2018) | Matterport3D (Anderson et al. 2018) | Gibson (Xia et al. 2018) | CMP (Gupta et al. 2017) |
|---|---|---|---|---|---|---|---|
| What perceptual modalities are available to the agent? | Visual, Audio, Semantic | Visual, Goal-oriented GPS + Compass, Semantic for Matterport3D (Slow), Collision | Visual, Surface normals, Touch (contact forces), Semantic segmentation | Visual, Semantic segmentation | Visual, Natural Language Processing | Visual, Surface normals, Semantics | Visual, Semantic |
| What actions are available to the agent? | Navigation, Interaction | Navigation, turn left, turn right, move forward, and stop | Goal-directed navigation (PointGoal, ObjectGoal, RoomGoal), Basic movement and look actions | Navigation (Goal-oriented using RoomNav) | Navigation (Room-to-Room dataset) | Navigation | Navigation, Semantic |
| Are multiagent interactions programmable? | Yes (can communicate to maximize a shared reward and solve the task) | No | No | No information | No information | No information | No information |
| Are wearable sensors available in the system? | No | Contact sensors | Audio, force, four contact-force sensors attached to agent | No | No | Supports all sensors available on robot agents | No information |
| Are ambient sensors available in the system? | Camera (RGB + Depth), Microphone | Camera (RGB + Depth) | Camera (RGB + Depth) | Camera (RGB + Depth) | Camera (RGB) | Camera (RGB + Depth) | Camera (RGB + Depth) |

**Table 1** (continued)

| Simulator | HoME (Brodeur et al. 2017) | Habitat (Savva et al. 2019) | MINOS (Savva et al. 2018) | House3D (Wu et al. 2018) | Matterport3D (Anderson et al. 2018) | Gibson (Xia et al. 2018) | CMP (Gupta et al. 2017) |
|---|---|---|---|---|---|---|---|
| Is the environment fixed or user-configurable? | Configurable, SUNCG dataset (Song et al. 2017) | Configurable, Matterport3D (Chang et al. 2017), Gibson (Xia et al. 2018), Replica dataset (Straub et al. 2019) | Configurable, SUNCG dataset (Song et al. 2017), Matterport3D dataset (Chang et al. 2017) | Configurable, SUNCG dataset (Song et al. 2017) | Configurable, Matterport3D dataset (Chang et al. 2017), Room-to-Room dataset | Configurable, Matterport3D (Chang et al. 2017), Gibson dataset (Xia et al. 2018) | Stanford large-scale 3D Indoor Spaces (S3DIS) dataset (Armeni et al. 2016) |
| Physics engine | Bullet | Bullet | Simple Rigid Body (Collision) | Simple Rigid Body (Collision) | No information | PyBullet | No information |
| Agent's body shape | Bullet Physics, capsule and spherical shape with physics properties radius, height, mass | Cylindrical primitive shape with diameter 0.2 m and height 1.5 m | Cylinder proxy geometry, return non-sensor measurements (shortest path, distance and direction to goal) | Camera-based agent | Camera-based agent | ROS robotic agents, allows importing arbitrary agents using URDFs. A variety of agents, along with a humanoid and an ant from Roboschool, a drone, a husky car, a Jackrabbot, and a minitaur, are also included as entrance points. MuJoCo XML agent models are used | Cylinder (discrete agent navigation) |
| Navigation state space | No information | Continuous | Continuous, discrete | Continuous, discrete | Continuous, discrete | Continuous | Discrete |

**Table 1** (continued)

| Simulator | | HoME (Brodeur et al. 2017) | Habitat (Savva et al. 2019) | MINOS (Savva et al. 2018) | House3D (Wu et al. 2018) | Matterport3D (Anderson et al. 2018) | Gibson (Xia et al. 2018) | CMP (Gupta et al. 2017) |
|---|---|---|---|---|---|---|---|---|
| Camera type | | No information | First person | Grayscale | First person | First person | First person | First person |
| External human interaction | | No | Humans-as-agents | No | No | No | No | No |
| Inter-active scene object | Approx. count | No information | No | No | No | No | No | No |
| | Action-able proper-ties | No information | No | No | No | No | No | No |
| | Material proper-ties | No information | No | No | No | No | No | No |
| | Contex-tual interac-tions | No information | No | No | No | No | No | No |
| Development language | | Python | C++ backend with Python API | WebGL Server, Python and web client API | Python | C++, using OpenGL, Python bindings also avail-able | Python | Python |

"User-specific" means the user is free to design and configure the relevant features that are not already provided or built in the simulator

**Table 2** A comparison of simulation platforms that are based on artist-authored environment and game engines

| Simulator | AI2-THOR (Kolve et al. 2017) | VirtualHome (Puig et al. 2018) | CHALET (Yan et al. 2018) | CAD$^2$RL (Sadeghi and Levine 2016) | VRGym (Xie et al. 2019) | Obstacle Tower (Juliani et al. 2019) |
|---|---|---|---|---|---|---|
| What perceptual modalities are available to the agent? | Visual (object detection predefined) | Visual | Sense (ray casting) | Vision (monocular) | Visual, Contact (Gazebo plugin) | Visual |
| What actions are available to the agent? | Navigation | Navigation (shortest-path planning, obstacle avoidance) | Navigation, interactions | Navigation | Navigation, interactions | Navigation, actions (idle, run, jump, etc.) |
| Are multiagent interactions programmable? | Yes | Yes | No | No | Yes (user-specific) | No |
| Are wearable sensors available in the system? | No | No | No | No | Gazebo sensors (camera, OpenNI Kinect, GPU laser, bumper) | No |
| Are ambient sensors available in the system? | Camera, (RGB + D) | Camera (RGB + D) segmentation, surface normals) | No | No | Camera (RGB + D), laser | Camera (RGB) |
| Is the environment fixed or user-configurable? | Configurable, photo-realistic, more than 120 house layouts | Configurable, photo-realistic, 7 default house layouts | Configurable, 10 default house layouts, custom layout using 58 rooms | Configurable, user-specific | Configurable, user-specific | Procedurally generated (100 floors with set of rooms) |
| Physics engine | Unity3D | Unity3D | Unity3D | No | Gazebo Physics and UE4 | Unity3D |

**Table 2** (continued)

| Simulator | AI2-THOR (Kolve et al. 2017) | VirtualHome (Puig et al. 2018) | CHALET (Yan et al. 2018) | CAD$^2$RL (Sadeghi and Levine 2016) | VRGym (Xie et al. 2019) | Obstacle Tower (Juliani et al. 2019) |
|---|---|---|---|---|---|---|
| Agent's body shape | Capsule shape (radius 0.2 m, height 1.8 m) | Humanoid character | FPS camera | FPS camera | Custom | Humanoid character |
| Navigation state space | Continuous, discrete | Continuous | Continuous | Continuous | Continuous | Continuous |
| Camera type | First person | First, third person, custom camera | First person | First person | VR stereo view for human, robot camera | Third person |
| External human interaction | No | No | No | No | Yes, using Oculus VR headset and Microsoft Kinect | Control the agent using a keyboard |

**Table 2** (continued)

| Simulator | | AI2-THOR (Kolve et al. 2017) | VirtualHome (Puig et al. 2018) | CHALET (Yan et al. 2018) | CAD$^2$RL (Sadeghi and Levine 2016) | VRGym (Xie et al. 2019) | Obstacle Tower (Juliani et al. 2019) |
|---|---|---|---|---|---|---|---|
| Inter-active Scene Object | Approx. count | 125 Objects | No information | 150 Object types | No | Custom, high-fidelity hand integration for many objects | No |
| | Actionable properties | Open, pickup, move, toggle, receptacle, fill, slice, cook, break, dirty, used up | Available on interactive objects (static, interact, grab), approx. 35 actions | 71 Kinds of objects can be manipulated.: 60 picked and placed, 6 opened and closed, 5 change state | No | Many actionable properties | No |
| | Material properties | Abstract temperature, change temp, mass, salient materials, inherited | No | No | No | No information | No |
| | Contextual interactions | Some objects are able to contextually affect the state or behavior of other objects | Yes | No | No | Yes | Sometimes agent has to collect key to open the door |

**Table 2** (continued)

| Simulator | AI2-THOR (Kolve et al. 2017) | VirtualHome (Puig et al. 2018) | CHALET (Yan et al. 2018) | CAD$^2$RL (Sadeghi and Levine 2016) | VRGym (Xie et al. 2019) | Obstacle Tower (Juliani et al. 2019) |
|---|---|---|---|---|---|---|
| Development language | Python | Python, High level instructions | C# | No information | Python, C++ | Python |
| Simulator | VizDoom (Kempka et al. 2016) | DeepMind Lab (Beattie et al. 2016) | Malmo (Johnson et al. 2016) | CARLA (Dosovitskiy et al. 2017) | VRKitchen (Gao et al. 2019) | Arena (Song et al. 2019) |
| What perceptual modalities are available to the agent? | Visual, Non-visual (health, ammunition) | Visual | Visual | Visual (RGB + Depth), Semantic segmentation | Visual (object detection predefined) | Visual |
| What actions are available to the agent? | Navigation, shoot | Navigation, training (collect rewards, explore environment), crouch, jump, look | Navigation, interactions (crafting, building, surviving, etc.) | Actions available to control vehicle | Navigation (path planning, collision detection) | Navigation, game-specific actions |
| Are multiagent interactions programmable? | Multiplayer game capabilities (agent vs. agent, agent vs. human) | Yes (interaction with built-in bots) | Yes (user-specific) | No (vehicles can detect other vehicles for motion control) | No | Yes (set of 100 agents/teams, user-specific interactions) |

**Table 2** (continued)

| Simulator | VizDoom (Kempka et al. 2016) | DeepMind Lab (Beattie et al. 2016) | Malmo (Johnson et al. 2016) | CARLA (Dosovitskiy et al. 2017) | VRKitchen (Gao et al. 2019) | Arena (Song et al. 2019) |
|---|---|---|---|---|---|---|
| Are wearable sensors available in the system? | Non-visual | Velocity | No | LIDARs, cameras, depth sensors, GPS, detectors (obstacle, collision, lane), GNSS, IMU, radar, RSS | No | No |
| Are ambient sensors available in the system? | Camera (RGB + D) | Camera (RGB + D) | No | No | Camera, (RGB + D, segmentation) | Camera (RGB), LiDAR |
| Is the environment fixed or user-configurable? | Environment is based on Doom First-person shooter (FPS) video game | Stairway to Melon Level, Nav Maze Level, Laser Tag Space Bounce Level | Dynamic and open, supports infinitely-varied environments | Configurable, various maps for urban environments with weather control and a blueprint library | Configurable, 16 photo-realistic kitchen scenes | Configurable, game-specific |
| Physics engine | Doom Engine | Quake III Arena | Minecraft Physics Engine | Unreal Engine | Unreal Engine 4 | Unity3D Engine |
| Agent's body shape | FPS camera | FPS camera | Minecraft characters | Two- and four-wheeled vehicles with advance physics | Humanoid character | Custom |
| Navigation state space | Continuous | Continuous | Continuous, discrete | Articulated | Continuous | Continuous, discrete |

**Table 2** (continued)

| Simulator | | VizDoom (Kempka et al. 2016) | DeepMind Lab (Beattie et al. 2016) | Malmo (Johnson et al. 2016) | CARLA (Dosovitskiy et al. 2017) | VRKitchen (Gao et al. 2019) | Arena (Song et al. 2019) |
|---|---|---|---|---|---|---|---|
| Camera type | | First person | First person | First, third person | Free camera, third person, custom | Third person | First, third person |
| External human interaction | | Learning from human demonstration | Human input controls for testing | Humans can play a Malmo mission as agents to collect training data for imitation learning | No | Yes | Yes |
| Inter-active Scene Object | Approx. count | Based on experiment selection | Based on experiment selection | Based on Minecraft sandbox video game (endless terrain, crafting tools and objects, create structures) | No | 55 Objects | Game-specific |

**Table 2** (continued)

| Simulator | | VizDoom (Kempka et al. 2016) | DeepMind Lab (Beattie et al. 2016) | Malmo (Johnson et al. 2016) | CARLA (Dosovitskiy et al. 2017) | VRKitchen (Gao et al. 2019) | Arena (Song et al. 2019) |
|---|---|---|---|---|---|---|---|
| | Actionable properties | Shooting the monster, collecting medkits, surviving in acidic environment | Gathering fruits, navigation through a maze, laser-tagging of inbuilt bots | Based on Minecraft game | No | Kitchen-based actions (cutting, opening a can, peeling, pouring, etc.) | Game-specific |
| | Material properties | Customizable resolution and rendering parameters | Randomly generated at start of each episode | Based on Minecraft game | No | No information | No |
| | Contextual interactions | No Information | No Information | Based on Minecraft game | No | Some objects are able to contextually affect the state or behavior of other objects | Game-specific |
| Development language | | C++ | API in C and Lua, Python bindings | Python, C++, C#, Java, Lua | Python, C++ | Python | Python |

**Table 3** A comparison of physics engine-based simulation platforms

| Comparison criteria | Physics engine-based simulators | | | Physics simulator | Robot simulator |
|---|---|---|---|---|---|
| | PyBullet (Caumans and Bai 2017) | SAPIEN (Xiang et al. 2020) | ThreeDWorld (Gan et al. 2020) | MuJoCo (Todorov et al. 2012) | Gazebo (Koenig and Howard 2004) |
| What perceptual modalities are available to the agent? | Custom physics-related perception can be developed | Visual, movable object detection | Visual, audio | Custom physics related perception can be developed | Visual (RGB + Depth), contact (with Gazebo plugin) |
| What actions are available to the agent? | Customizable | Not available (custom can be developed) | Not available (custom can be developed) | Customizable | Customizable |
| Are multiagent interactions programmable? | Yes | Interaction with environment only | Yes | Yes | Yes |
| Are wearable sensors available in the system? | No | No | No | IMU, sensors for touch, force–torque, velocity and force, joint and tendon position, and actuator position, magnetometers | Robot agent Gazebo sensors (camera, laser, OpenNI Kinect, GPU laser, bumper) |
| Are ambient sensors available in the system? | No | Camera (RGB + Depth) | Camera (RGB), Microphone | No | Camera (RGB + Depth), Laser |
| Is the environment fixed or user-configurable? | Configurable, OpenGL for visualization | Configurable, User-specific | Configurable, User-specific | Configurable, OpenGL or Unity3D for visualization | Configurable, User-specific |

**Table 3** (continued)

| Comparison criteria | | Physics engine-based simulators | | | Physics simulator | Robot simulator |
|---|---|---|---|---|---|---|
| | | PyBullet (Caumans and Bai 2017) | SAPIEN (Xiang et al. 2020) | ThreeDWorld (Gan et al. 2020) | MuJoCo (Todorov et al. 2012) | Gazebo (Koenig and Howard 2004) |
| Physics Engine | | Bullet | PhysX and ROS control interface | PhysX and NVIDIA Flex | MuJoCo | Gazebo Physics |
| Agent's body shape | | Custom | Custom, robots, rigid body, articulated objects | Custom | Custom | Robots, including PR2, iRobot Create, Pioneer2 DX and TurtleBot or custom using SDF |
| Navigation state space | | Continuous | No information | Continuous | Continuous | Articulated |
| Camera type | | Custom | Custom | Custom | Custom | Robot camera, custom |
| External human interaction | | No | No | Yes | No | No |
| Inter-active Scene Object | Approx. count | No | Custom | Custom | No | No |
| | Actionable properties | No | No | No | No | No |
| | Material properties | No | No | No | No | No |
| | Contextual interactions | No | No | No | No | No |
| Development language | | Python | Python, C++ | Python | C, C++, Python | Can integrate with ROS that supports Python and C++ |

**Fig. 27** MuJoCo environment (Todorov et al. 2012)

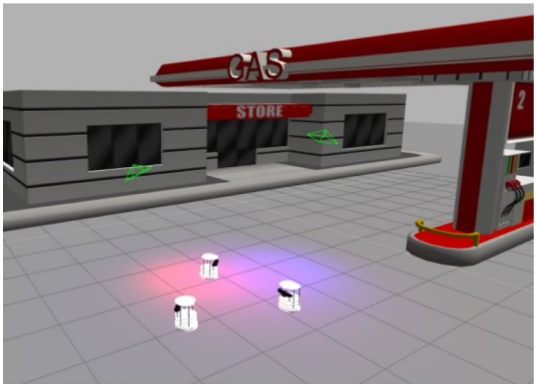


**Fig. 28** Gazebo environment (Koenig and Howard 2004)

Gazebo is an outdoor robot simulator with limited fidelity. Still, it has features such as programmable objects and multi-level image extrusion that make Gazebo useful in many AI applications.

Table 4 summarizes the applications of reviewed simulators in various domains, and includes their current state (active or inactive), and URL for download and support.

## 5 Discussion: which simulation platform is the best?

Each simulator is unique in its own way, so ranking them is a challenge. This section ranks the reviewed simulators based on a rubric.

Most of the reviewed systems allow simulation of an embodied reinforcement learning agent in an environment that rewards when a predefined goal is reached. However, there is seldom any information regarding the number of agents that can be simulated simultaneously. The kind of agent that can be simulated (e.g., imitation learning agent versus reinforcement learning agent, as in CARLA) is either in-built or user-specific.

The criteria for selecting a platform for embodied AI tasks must depend on the trade-offs between available features, generality of platforms, and ease of use. It must be ensured that the chosen platform is capable of accommodating the essence of perceptually realistic

**Table 4** Applications of the reviewed simulators

| Simulator [Current State] | Dominant Application Areas [Download and support URL] |
|---|---|
| HoME [Inactive since 2018] | Physics, semantics and interaction with objects and agents in a realistic setting, instruction following, visual question answering, agent and oracle conversation to solve complex tasks, sound source localization with multichannel microphones equipped by agents, multiagent communication https://github.com/pswietojanski/home-platform |
| Habitat [Active] | Enables navigation training of embodied AI agents, PointGoal navigation https://aihabitat.org/, https://github.com/facebookresearch/habitat-sim |
| MINOS [Inactive since 2019] | Studies the performance of learning-based navigation agents in cluttered indoor environments, facilitates the implementation of multisensory models for goal-directed movement in complicated indoor situations http://minosworld.org, https://github.com/minosworld/minos |
| House3D [Inactive since 2020] | Comprises of thousands of indoor sceneries, facilitates a range of activities such as 3D navigation, visual perception, linguistic grounding, idea acquisition, embodied question answering, object and scene understanding http://github.com/facebookresearch/House3D |
| Matterport3D [Active] | Facilitates the creation of AI agents that engage with real-world 3D scenes utilizing visual data, can perform language-based navigation. Useful for providing virtual tours https://bringmeaspoon.org, https://github.com/peteanderson80/Matterport3DSimulator |
| Gibson [Active] | Supports development of real-world perception for active agents, obstacle avoidance planning, long-distance navigation, and visual stair climbing http://gibson.vision/, https://github.com/StanfordVL/GibsonEnv |
| CMP [Inactive since 2019, deprecated] | Cognitive mapping and visual navigation https://github.com/agiantwhale/cognitive-mapping-agent |
| AI2-THOR [Active] | Used to learn navigation and interaction by AI agents, allows investigation of differences in appearance and control https://ai2thor.allenai.org, https://github.com/allenai/ai2thor |
| VirtualHome [Active] | Simulates complex household activities via programs. Allows complex interactions with the environment, produce movies of human actions; also contains a knowledge base with instructions for a wide range of tasks http://virtual-home.org/, https://github.com/xavierpuigf/virtualhome |
| CHALET [Inactive since 2019] | 3D house simulator, allows for a wide variety of object manipulation, navigation, interactive question answering and instruction following https://github.com/lil-lab/chalet, https://lil.nlp.cornell.edu/resources/CHALET/ |
| CAD$^2$RL [Not released] | Supports real-world collision-free indoor flying simulation https://fsadeghi.github.io/CAD2RL/ |
| VRGym [Not released] | VR testbed for realistic human–robot interaction, can record human interactions and delicate manipulations, support multiple robots, allow research in human–robot interaction, and offer development tools for training cutting-edge ML algorithms https://xuxie1031.github.io/projects/VRGym/VRGymProj.html |
| Obstacle Tower [Active] | A game where an agent can play and learn in a procedurally generated environment https://github.com/Unity-Technologies/obstacle-tower-env |

**Table 4** (continued)

| Simulator [Current State] | Dominant Application Areas [Download and support URL] |
|---|---|
| VizDoom [Active] | Vision-based reinforcement learning by bots for move-and-shoot task and maze navigation problem. Enables experimentation with various learning paradigms such as reinforcement learning, apprenticeship learning, and learning by demonstration through multiple modes of operation http://vizdoom.cs.put.edu.pl/, https://github.com/mwydmuch/ViZDoom |
| DeepMind Lab [Active] | Useful for reinforcement learning, unsupervised auxiliary tasks, study navigation, planning, strategy, creative task development, 3D vision from raw pixel inputs, fine motor dexterity https://github.com/deepmind/lab |
| Malmo [Active] | Enables AI experimentation and research using Minecraft, and assists agents in sensing and acting inside the Minecraft environment. Activities include strolling around, hunting for treasures, constructing a tower with a group of colleagues, etc https://www.microsoft.com/en-us/research/project/project-malmo/, https://github.com/microsoft/malmo |
| CARLA [Active] | Aids in the democratization of autonomous driving research and development by providing a platform that can be readily used and configured https://carla.org/, https://github.com/carla-simulator/carla |
| VRKitchen [Active] | Enables embodied agents to perform and learn challenging tasks requiring a wide variety of fine-grained object operations in a realistic setting https://github.com/xfgao/VRKitchen |
| Arena [Active] | A multiagent intelligence evaluation platform with a collection of various games that allows for the expansion of new social models based on GUI-configurable social trees https://sites.google.com/view/arena-unity/, https://github.com/YuhangSong/Arena-BuildingToolkit |
| PyBullet [Active] | Includes Python package for simulating physics, robotics, and deep reinforcement learning. Provides inverse dynamics computation, collision detection, forward dynamics simulation, forward and inverse kinematics and ray intersection queries https://pybullet.org/wordpress/, Forum: https://pybullet.org/Bullet/phpBB3, https://github.com/bulletphysics/bullet3, https://pypi.org/project/pybullet/ |
| SAPIEN [Active] | Allows for a variety of robotic vision and interaction tasks that need deep component-level knowledge, part identification, and motion characteristic recognition, as well as the demonstration of robotic interaction tasks utilizing heuristic techniques and reinforcement learning. Provides multiple rendering modalities including optical flow depth map, normal map, ray tracing, and active light https://sapien.ucsd.edu/, https://github.com/haosulab/SAPIEN-Release |
| ThreeDWorld [Active] | Supports environmental audio and reverberation, human interactions with VR devices. Enables attention studies in humans and neural networks. Includes multiagent interactions, physical dynamics prediction, multimodal physical scene understanding, models that learn like a child http://threedworld.org/, https://github.com/threedworld-mit/tdw |
| MuJoCo [Active] | Fast and accurate physics engine for constrained systems. Specifically designed for multiple joint dynamical systems in contact-rich behaviors http://www.mujoco.org/, https://www.roboti.us/ |

**Table 4** (continued)

| Simulator [Current State] | Dominant Application Areas [Download and support URL] |
|---|---|
| Gazebo [Active] | Capable of effectively and efficiently simulating robot populations in complex indoor and outdoor contexts, perform regression testing, train AI systems, design robots, rapidly test algorithms http://gazebosim.org/, https://github.com/osrf/gazebo, Forum: https://answers.gazebosim.org/questions |

**Table 5** Rubrics for grading the simulators

| Criteria | Score |
|---|---|
| How many embodied agents can be simulated simultaneously? | 0: No agent can be simulated<br>0.5: Only one agent can be simulated<br>1: Multiple agents can be simulated simultaneously |
| Is the visual environment configurable? | 0.3: Fixed<br>0.65: Partially configurable<br>1: Fully configurable |
| How many modalities are available? | 0.3: Visual only<br>0.65: Visual + semantic<br>1: Visual + semantic + any other |
| How many actions are available? | 0: No action<br>0.5: Navigation only<br>1: Navigation and other actions |
| What types of agent interactions are available? | 0: No interaction<br>0.5: Interactions with the environment only<br>1: Interactions with the environment and other agents |
| How realistic is the physics in the system? | 0: No physics<br>0.5: Simple rigid body physics<br>1: Physics based on physics-engine or game-engine |
| Availability and support for simulator in the form of a webpage (e.g., in GitHub) containing instructions/manual stating how to install and use the software, and a contact email for users to report issues | 0: Software is not available for download<br>0.5: Software is available for download but no support is available<br>1: Software is available for download and support is available |

environment and the physics needed for achieving the agent tasks, so that the learned models are ready to be transferred to the real world. Table 5 presents the rubrics for assessment of the simulators; their scores are presented in Table 6.

## 6 Conclusions

There has been an exponential rise in research interest in software platforms for simulating embodied agents in 3D virtual environments. Platforms that simulate with high fidelity to the real world the interaction among embodied agents and with their environments are highly desirable. We reviewed 22 simulators reported in the literature since the year 2016. We classified them based on visual environment and physics, and compared them based on their properties and functionalities from a user's perspective. We scored the simulators using a rubric containing seven factors encompassing their properties, functionalities,

**Table 6** Assessment of simulators based on our rubric. The simulators are sorted by their total score

| Simulator (year of release) | Number of agents | Configurable environment | Number of modalities | Number of actions | Agent interactions | Physics-based interactions | Support | Total score (max. 7.0) |
|---|---|---|---|---|---|---|---|---|
| HoME (2017) | 1 | 0.65 | 1 | 1 | 1 | 1 | 1 | 6.65 |
| Malmo (2016) | 1 | 1 | 0.3 | 1 | 1 | 1 | 1 | 6.3 |
| Arena (2019) | 1 | 0.65 | 0.3 | 1 | 1 | 1 | 1 | 5.95 |
| VirtualHome (2018) | 1 | 0.65 | 0.3 | 1 | 1 | 1 | 1 | 5.95 |
| DeepMind Lab (2016) | 1 | 0.65 | 0.3 | 1 | 1 | 1 | 1 | 5.95 |
| Gibson (2018) | 0.5[a] | 0.65 | 1 | 1 | 0.5[a] | 1 | 1 | 5.65 |
| VizDoom (2016) | 1 | 0.3 | 0.3 | 1 | 1 | 1 | 1 | 5.6 |
| AI2-THOR (2017) | 1 | 0.65 | 0.3 | 0.5 | 1 | 1 | 1 | 5.45 |
| SAPIEN (2020) | 1 | 1 | 0.3[b] | 0 | 1 | 1 | 1 | 5.3 |
| ThreeDWorld (2020) | 1 | 1 | 0.3[c] | 0 | 1 | 1 | 1 | 5.3 |
| VRGym (2019) | 1 | 1 | 0.3[d] | 1 | 1 | 1 | 0 | 5.3 |
| PyBullet (2017) | 1 | 1 | 0.3 | 0 | 1 | 1 | 1 | 5.3 |
| CARLA (2017) | 0.5 | 1 | 0.65 | 0.5 | 0.5 | 1 | 1 | 5.15 |
| CHALET (2018) | 0.5 | 0.65 | 0.3[e] | 1 | 0.5 | 1 | 1 | 4.95 |
| Habitat (2019) | 0.5 | 0.65 | 1 | 0.5 | 0 | 1 | 1 | 4.65 |
| Obstacle Tower (2019) | 0.5 | 0.65 | 0.3 | 0.5 | 0.5 | 1 | 1 | 4.45 |
| VRKitchen (2019) | 0.5 | 0.65 | 0.3 | 0.5 | 0.5 | 1 | 1 | 4.45 |
| MINOS (2018) | 0.5 | 0.65 | 1 | 0.5 | 0 | 0.5 | 1 | 4.15 |
| House3D (2018) | 0.5[a] | 0.65 | 1 | 0.5 | 0 | 0.5 | 1 | 4.15 |
| CMP (2017) | 0.5[a] | 0.65 | 0.65 | 1 | 0 | 0 | 1 | 3.8 |
| Matterport3D (2018) | 0.5[a] | 0.65 | 0.65 | 0.5 | 0 | 0 | 1 | 3.3 |
| CAD$^2$RL (2016) | 0.5 | 1 | 0.3 | 0.5 | 0.5 | 0 | 0 | 2.8 |

[a]No information regarding multiple agent simulation

[b]In addition to visual perception, it can also detect movable object

[c]Audio perception is available in addition to visual perception

[d]In addition to visual perception, it can also detect contact with another object

[e]It does not have visual perception, but senses an object using ray casting

availability and support. HoME, Malmo, Arena, VirtualHome, and DeepMind Lab take the top five spots closely followed by some of the others. We acknowledge that no simulator is better than the others in all respects and discuss the uniqueness of each. This review, the first of its kind, will guide users to choose the appropriate simulator for their application and provide a baseline to researchers for developing state-of-the-art simulators.

## References


Anderson P, Wu Q, Teney D et al (2018) Vision-and-Language Navigation: interpreting visually-grounded navigation instructions in real environments. In: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp 3674–3683

Armeni I, Sener O, Zamir AR, Jiang H, Brilakis I, Fischer M, Savarese S (2016) 3D semantic parsing of largescale indoor spaces. In: CVPR, 2016

Banerjee B, Chandrasekaran B (2010a) A constraint satisfaction framework for executing perceptions and actions in diagrammatic reasoning. J Artif Intell Res 373–427

Banerjee B, Chandrasekaran B (2010b) A spatial search framework for executing perceptions and actions in diagrammatic reasoning. In: International conference on theory and application of diagrams, 2010b. Springer, Berlin, pp 144–159

Banerjee B et al (2021) Synthesizing skeletal motion and physiological signals as a function of a virtual human's actions and emotions. In: SIAM international conference on data mining, 2021, pp 684–692

Baruah M, Banerjee B, Nagar AK (2022) An attention-based predictive agent for static and dynamic environments. IEEE Access 10:17310–17317. https://doi.org/10.1109/ACCESS.2022.3149585

Baruah M, Banerjee B (2020a) The perception–action loop in a predictive agent. In: Annual meeting of the Cognitive Science Society, 2020, pp 1171–1177

Baruah M, Banerjee B (2020b) A multimodal predictive agent model for human interaction generation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops, 2020b

Baruah M, Banerjee B (2022) Speech emotion recognition via generation using an attention-based variational recurrent neural network. In: INTERSPEECH, 2022, Incheon, Korea

Beattie C, Leibo J Z, Teplyashin D et al (2016) DeepMind Lab. arXiv preprint arXiv:1612.03801

Blender Community (nd) Blender: Open Source 3D modeling suit. http://www.blender.org

Brockman G, Cheung V, Pettersson L, Schneider J, Schulman J, Tang J, Zaremba W (2016) OpenAI gym. arXiv:1606.01540

Brodeur S, Perez E, Anand A et al (2017) HoME: a household multimodal environment. arXiv preprint arXiv:1711.11017

Brooks RA (2018) Intelligence without reason. In: Steels L, Brooks RA (eds) The artificial life route to artificial intelligence: building embodied, situated agents. Routledge, London, pp 25–81

Brooks RA, Breazeal C, Marjanović M, Scassellati B, Williamson MM (1998) The Cog project: building a humanoid robot. In: International workshop on computation for metaphors, analogy, and agents, April 1998. Springer, Berlin, pp 52–87

Busby J, Parrish Z, Wilson J (2010) Mastering unreal technology. Sams Publishing, Indianapolis

Chang A, Dai A, Funkhouser T, Halber M, Niessner M, Savva M, Song S, Zeng A, Zhang Y (2017) Matterport3D: learning from RGB-D data in indoor environments. In: International conference on 3D vision (3DV), 2017

Chaplot DS, Dalal M, Gupta S, Malik J, Salakhutdinov RR (2021) SEAL: self-supervised embodied active learning using exploration and 3D consistency. In: Advances in neural information processing systems, vol 34

Coumans E, Bai Y (2017) PyBullet, a Python module for physics simulation in robotics, games and machine learning

Coumans E, Bai Y (2016) PyBullet, a Python module for physics simulation for games, robotics and machine learning

Das A, Datta S, Gkioxari G, Lee S, Parikh D, Batra D (2018) Embodied question answering. In: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp 1–10

Deng E, Mutlu B, Mataric M (2019) Embodiment in socially interactive robots. arXiv preprint arXiv:1912.00312

Dosovitskiy A, Ros G, Codevilla F, Lopez A, Koltun V (2017) CARLA: an open urban driving simulator. arXiv preprint arXiv:1711.03938

Fischer MH, Zwaan RA (2008) Embodied language: a review of the role of the motor system in language comprehension. Q J Exp Psychol 61(6):825–850
Gan C, Schwartz J, Alter S et al (2020). ThreeDWorld: a platform for interactive multi-modal physical simulation. arXiv preprint arXiv:2007.04954
Gao X, Gong R, Shu T, Xie X, Wang S, Zhu SC (2019) VRKitchen: an interactive 3D virtual environment for task-oriented learning. arXiv preprint arXiv:1903.05757
Gorisse G, Christmann O, Amato EA, Richir S (2017) First- and third-person perspectives in immersive virtual environments: presence and performance analysis of embodied users. Front Robot AI 4:33
Gupta S, Davidson J, Levine S, Sukthankar R, Malik J (2017) Cognitive mapping and planning for visual navigation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp 2616–2625
Han J, Waddington G, Adams R, Anson J, Liu Y (2016) Assessing proprioception: a critical review of methods. J Sport Health Sci 5(1):80–90
Johnson M, Hofmann K, Hutton T, Bignell D (2016) The Malmo platform for artificial intelligence experimentation. In: IJCAI, 2016, pp 4246–4247
Juliani A, Khalifa A, Berges VP et al (2019) Obstacle Tower: a generalization challenge in vision, control, and planning. arXiv preprint arXiv:1902.01378
Juliani A, Berges V, Vckay E, Gao Y, Henry H, Mattar M, Lange D (2018) Unity: a general platform for intelligent agents. arXiv:1809.02627
Kang SC, Juang JR, Hung W (2011) Using game engine for physics-based simulation—a forklift. J Inf Technol Constr 16:3–22
Kempka M, Wydmuch M, Runc G, Toczek J, Jaśkowski W (2016) VizDoom: a doom-based AI research platform for visual reinforcement learning. In: IEEE conference on computational intelligence and games, September 2016, pp 1–8.
Kim G (2015) Human–computer interaction. Auerbach Publications, Boca Raton
Koenig N, Howard A (2004) Design and use paradigms for Gazebo, an open-source multi-robot simulator. In: IEEE/RSJ international conference on intelligent robots and systems, September 2004, vol 3, pp 2149–2154
Kolve E, Mottaghi R, Han W et al (2017) AI2-THOR: an interactive 3D environment for visual AI. arXiv preprint arXiv:1712.05474
Laine S, Siltanen S, Lokki T, Savioja L (2009) Accelerated beam tracing algorithm. Appl Acoust 70(1):172–181
Mnih V, Kavukcuoglu K, Silver D et al (2015) Human-level control through deep reinforcement learning. Nature 518:529 EP
Najnin S, Banerjee B (2017) A predictive coding framework for a developmental agent: speech motor skill acquisition and speech production. Speech Commun 92:24–41
Nikolenko SI (2021) Synthetic simulated environments. In: Synthetic data for deep learning. Springer optimization and its applications, vol 174. Springer, Cham. https://doi.org/10.1007/978-3-030-75178-4_7
Pfeifer R, Lungarella M, Iida F (2007) Self-organization, embodiment, and biologically inspired robotics. Science 318(5853):1088–1093
Pfeifer R, Lungarella M, Sporns O (2008) The synthetic approach to embodied cognition: a primer. In: Handbook of Cognitive science. Elsevier, Amsterdam, pp 121–137
Puig X, Ra K, Boben M, Li J, Wang T, Fidler S, Torralba A (2018) VirtualHome: simulating household activities via programs. In: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp 8494–8502
Russell S, Norvig P (2020) Artificial intelligence: a modern approach, 4th edn. Pearson, Hoboken
Sadeghi F, Levine S (2016) CAD$^2$RL: real single-image flight without a single real image. arXiv preprint arXiv:1611.04201
Savva M, Kadian A, Maksymets O et al (2019) Habitat: a platform for embodied AI research. In: Proceedings of the IEEE international conference on computer vision, 2019, pp 9339–9347
Savva M, Chang AX, Dosovitskiy A, Funkhouser T, Koltun V (2017) MINOS: multimodal indoor simulator for navigation in complex environments. arXiv preprint arXiv:1712.03931
Shapiro L (2019) Embodied cognition. Routledge, London
Smith L, Gasser M (2005) The development of embodied cognition: six lessons from babies. Artif Life 11(1–2):13–29
Song P, Yu H, Winkler S (2008) Vision-based 3D finger interactions for mixed reality games with physics simulation. In: Proceedings of the 7th ACM SIGGRAPH international conference on virtual-reality continuum and its applications in industry, December 2008, pp 1–6
Song S, Yu F, Zeng A, Chang AX, Savva M, Funkhouser T (2017) Semantic scene completion from a single depth image. In: CVPR, 2017

Song Y, Wojcicki A, Lukasiewicz T et al (2020) Arena: a general evaluation platform and building toolkit for multiagent intelligence. In: Proceedings of the AAAI conference on artificial intelligence, April 2020, vol 34, No. 05, pp 7253–7260.

Straub J, Whelan T, Ma L et al (2019) The Replica dataset: a digital replica of indoor spaces. arXiv:1906.05797

Todorov E, Erez T, Tassa Y (2012) MuJoCo: a physics engine for model-based control. In: IEEE/RSJ international conference on intelligent robots and systems, October 2012, pp 5026–5033

Walczak K, Sokolowski J, Dziekoński J (2018) Configurable virtual reality store with contextual interaction interface. In: 2018 11th International conference on human system interaction (HSI), July 2018. IEEE, pp 28–34

Wang R, Qian X (2012) OpenSceneGraph 3 Cookbook. Packt Publishers Ltd., Birmingham

Wilson M (2002) Six views of embodied cognition. Psychonom Bull Rev 9(4):625–636

Wu Y, Wu Y, Gkioxari G, Tian Y (2018) Building generalizable agents with a realistic and rich 3D environment. arXiv preprint arXiv:1801.02209

Xia F, Zamir AR, He Z, Sax A, Malik J, Savarese S (2018) Gibson env: real-world perception for embodied agents. In: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp 9068–9079

Xiang F, Qin Y, Mo K et al (2020) SAPIEN: a simulated part-based interactive environment. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2020, pp 11097–11107

Xie X, Liu H, Zhang Z et al (2019) VRGym: a virtual testbed for physical and interactive AI. In: Proceedings of the ACM Turing celebration conference-China, May 2019, pp 1–6

Yan C, Misra D, Bennnett A, Walsman A, Bisk Y, Artzi Y (2018) CHALET: Cornell house agent learning environment. arXiv preprint arXiv:1801.07357