Proceedings of the 13th Conference on Language Resources and Evaluation (LREC 2022), pages 1529–1541
Marseille, 20-25 June 2022

The VoxWorld Platform for Multimodal Embodied Agents

Nikhil Krishnaswamy*, William Pickard*, Brittany Cates*, Nathaniel Blanchard*, and James Pustejovsky†

*Colorado State University
Fort Collins, CO, USA
{nkrishna,william.pickard,brittany.cates, nathaniel.blanchard}@colostate.edu

†Brandeis University
Waltham, MA, USA
jamesp@brandeis.edu

Abstract

We present a five-year retrospective on the development of the VoxWorld platform, first introduced as a multimodal platform for modeling motion language, that has evolved into a platform for rapidly building and deploying embodied agents with contextual and situational awareness, capable of interacting with humans in multiple modalities, and exploring their environments. In particular, we discuss the evolution from the theoretical underpinnings of the VoxML modeling language to a platform that accommodates both neural and symbolic inputs to build agents capable of multimodal interaction and hybrid reasoning. We focus on three distinct agent implementations and the functionality needed to accommodate all of them: Diana, a virtual collaborative agent; Kirby, a mobile robot; and BabyBAW, an agent who self-guides its own exploration of the world.

Keywords: multimodality, multimodal interaction, situated grounding, embodied agent, modeling platform, simulation

1. Introduction

Multimodal language understanding has been a topic of intense interest in the natural language processing community over at least the last half a decade. The role that perception plays in human language understanding has long been clear to much of the community and many efforts have attempted to build similar functionality into computer systems (Pylyshyn, 1978; Waltz, 1978; Rehm and Goecke, 2000; Rohlfing et al., 2003; Pecher and Zwaan, 2005; Mooney, 2008; Scherer et al., 2012; Krishnamurthy and Kollar, 2013; Miller and Johnson-Laird, 2013). There has also been interest in context-aware language understanding systems, and the role that situatedness and embodiment play in such processes. This is an even harder problem than merging computer vision and NLP into a single system, and perhaps because of this, interest has remained high. In 2016, at the LREC conference, we introduced a modeling language **VoxML** (Pustejovsky and Krishnaswamy, 2016), intended to capture the semantics necessary to construct 3D visualizations of concepts denoted by linguistic expressions; as well as **VoxSim**, later that year at COLING 2016 (Krishnaswamy and Pustejovsky, 2016), a software system that generated such visualizations from the VoxML semantics.

Interest in this approach has persisted through the community (Cohen, 2017; Quick and Morrison, 2017; Fischer et al., 2018; Abrami et al., 2020; Bonn et al., 2020; Henlein et al., 2020; Rodrigues et al., 2020; Tamari et al., 2020; Kozierok et al., 2021; Richard-Bollans, 2021), and out of our initial goal to model visualized and situated object and event semantics grew an additional line of research: creating intelligent agent behaviors capable of performing *reasoning* over such semantics (Krishnaswamy et al., 2017; Narayana et al., 2018). VoxML and VoxSim became the platform on which to build such agent behaviors.

However, intelligent agents can have many forms and many purposes. The process of developing such agents in time made clear what constituted the key components of a *platform* for rapidly developing multimodal embodied agents, what was essential and what was extraneous, and how to develop and sustain such a platform for continued use. In this paper we present a retrospective of the evolution of the VoxML/VoxSim platform, now collectively known as **VoxWorld**, discuss our sometimes surprising conclusions regarding the development of VoxWorld as a distinct resource, and demonstrate how we now facilitate development of new interactive agents that synthesize our work with various other contributions, representations, and pipelines from the NLP/CL and AI communities.

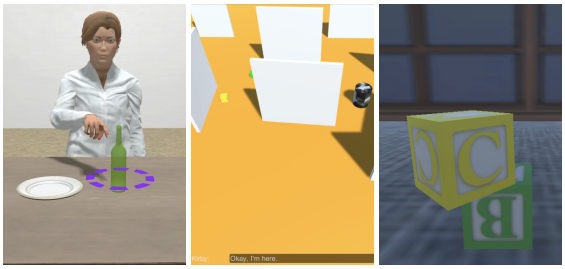

Figure 1: 3 VoxWorld agents

2. Related Work

Earlier work in "agent architectures" detailed the arrangement of conceptual components, e.g., facts, goals, and plans, for particular types of intelligent control system. A canonical example in NLP is TRIPS (Ferguson et al., 1998). Other cognitive architectures for intelligent agents include SOAR (Laird, 2019), ACT-R (Anderson et al., 1997), and other Belief-Desire-Intention (BDI) agents (Rao et al., 1995).

Explicit representation has largely fallen out of favor in

modern AI, and earlier agent architectures have therefore adapted to integrate deep learning inputs. Brooks (1991) presaged this move, but notably viewed the shortcomings of symbolic AI as including its lack of capturing either situatedness and embodiment with regard to robotics; the robotics community has, in turn, provided many treatments for robots that can plan and reason about actions, language, and social behaviors (Breazeal et al., 2004; Dzifcak et al., 2009; Tellex et al., 2020), with platforms for robotic development (e.g., Thrun et al. (2000), Schermerhorn et al. (2006), Rusu et al. (2008)).

Finally, along with the GPUs that facilitated the successes of deep learning came sophisticated graphics engines that allow developers, often with roots in the video gaming community, to develop photo-realistic simulators in which to develop and test intelligent agents and their ability to learn. Much of this work comes from the deep reinforcement learning community, where simulated environments are used for navigation, game-playing, and problem solving via deep RL (Kempka et al., 2016; Kolve et al., 2017; Savva et al., 2017; Juliani et al., 2018; Yan et al., 2018; Savva et al., 2019). These environmental platforms are not developed specifically to focus on communication, underspecification resolution, language grounding, or concept acquisition, though they may be used for these.

One particular suite of work of note is the work done by the EASE Collaborative Research Center at the University of Bremen. VoxWorld and the EASE ecosystem are similar in some of the tasks that agents are being trained to do, e.g., moving objects around a virtual environment to achieve certain goals (Kazhoyan and Beetz, 2019). There are also compatibilities in information representation techniques, for example the semantics of expressing locations, relationships of objects (Pomarlan and Bateman, 2020), and affordances (Beßler et al., 2020). However, the EASE work is more geared toward teaching robots to perform human tasks than to receiving direction from a human on a cooperative task. The EASE work on hand tracking and motion interpretation (Ramirez-Amaro et al., 2019; Rosskamp et al., 2020) is distinct from yet complementary with VoxWorld's use of gesture, gaze, etc., for communication.

In VoxWorld, we provide a common architecture that is compatible with both explicit logical and implicit subsymbolic representations, making it a robust platform for situated, embodied, neurosymbolic AI. VoxWorld is unique compared to other game engine-based environments for the following reason: while in other environments, the constraints on and results of object interactions are often coded into the system ad hoc as needed for a particular scenario, VoxWorld runs on a rigorous composition of object, event, relation, attribute, and function semantics at runtime, where the results of actions in the world flow naturally from the semantics of objects over which they are enacted, enabling generalizable situated reasoning.

To our knowledge, VoxWorld is the first platform to bring the above areas together, with a core yet extensible architecture driven by VoxML, in a 3D simulated world that enables multimodal interaction with, and observation of, agents in real or virtual environments.

3. What Makes an Agent?

Discussion of agents necessarily raises the question: what makes any given system an agent? Before the emergence of AI, many in philosophy held the view that action is to be explained in terms of the intentionality associated with an act. The concepts of "agency" and "agent" were mostly used to refer to the "exercise of the capacity to perform intentional actions" (Anscombe, 1957; Davidson, 1963). In psychology, agency entails intentionality (Dennett, 1987) and it cannot be meaningfully argued that computational "agents" as currently known have intentionality.

"Agents" in AI avoid this criticism by eschewing direct comparisons to human intelligence. There is little argument in the AI community about whether intelligent agents display "real" intelligence or not. They are simply systems that "can be viewed as perceiving [its] environment through sensors and acting upon that environment through actuators" (Russell and Norvig, 2002).

These actuators can be articulators like limbs, as in robots, or simply routines that effect change upon the world, as in an automated thermostat or flight-booking system. Personal digital assistants are perhaps the most common intelligent agents in use; Siri, Alexa, etc., live in our pockets or our offices, where they retrieve information or execute specific tasks on command.

Agents can also, in the context of virtual environments, exist entirely in simulation, where the only world they directly affect is one of data structures and pixels. Virtual worlds may be one of the most interesting and fruitful places to study agent behaviors, because it is much simpler in a virtual world to create an agent that is *situated*. A necessary condition on an situated agent is that it have an epistemic point of view associated with it, from which it can observe the world, and this has been an object of previous study (Pustejovsky and Krishnaswamy, 2019). Once the epistemic condition is imposed, the rendering of the virtual world from that point of view becomes a *mode of presentation* of the agent's understanding of the situation as encoded within the virtual environment. Thus the rendering serves as a rough analogue for human perception, by allowing an observer to perceive what the agent does.

Shared perception is a critical component of human communication (Kuhl, 1998; Pustejovsky and Krishnaswamy, 2020). When co-situated in a space together, humans make some tacit assumptions about what the other people are aware of and how they may behave, but part of those behaviors may require the agent to not only have a point of view as above, but also an *embodiment*. Alexa cannot see what you are pointing at, and

neither can she point herself. Thus embodied agents add new dimensions to human/agent interactions compared to voice- or text-only conversational agents (Allouch et al., 2021).

To act or manipulate within their world (real or virtual), agents must be equipped with the appropriate output, such as actuators or inverse kinematic solvers to move their joints, and text-to-speech engines, for which there are many available solutions. If an agent can interact with the world, allowing humans to interact with it requires the inverse functionality; it must be able to do things like recognize and interpret inputs in multiple modalities like gesture, speech, gaze, and action, and solving these problems has driven the development of VoxWorld as a platform.

In Sec. 4 of this paper, we provide an overview of the VoxWorld platform and detail the core components that we have developed to facilitate creation of agents that act in the world and interact with people. In Sec. 5, we describe 3 specific agent implementations (Fig. 1): Diana, a virtual collaborative agent; Kirby, a mobile robot; and BabyBAW, an agent who self-guides its own exploration of the world, and discuss specific considerations for each that informed the development of VoxWorld. We conclude with making VoxWorld available as a public resource.

4. VoxWorld Platform Overview

At the core of VoxWorld are the VoxML modeling language and its real-time interpreter, VoxSim. VoxSim is built on the Unity game engine making it a clear companion to other Unity-based frameworks (e.g., Juliani et al. (2018), see Sec. 5.3). VoxML models contextual and common-sense information about objects and events that is otherwise difficult to capture in unimodal corpora, e.g., *balls roll because they are round*. VoxML is particularly apt for capturing information about *habitats* (Pustejovsky, 2013) and *affordances* (Gibson, 1977). Fig. 2 shows the affordance structure for a [[BLOCK]] *voxeme*, or the visual instantiation of a block object. In implementation, VoxML is saved in XML format in a directory accessible to a VoxWorld-based agent implementation.

$$\left[\begin{array}{ll} \textbf{block} & \\ \text{AFFORD\_STR} = & \left[\begin{array}{ll} \text{A}_1 = & H_{[2]} \rightarrow [put(x, y, on([1]))] \\ & support([1], y) \\ \text{A}_2 = & H_{[2]} \rightarrow [grasp(x, [1])] \\ & hold(x, [1]) \\ \text{A}_3 = & H_{[2]} \rightarrow [lift(x, [1])] \\ & hold(x, [1]) \\ \text{A}_4 = & H_{[2]} \rightarrow [ungrasp(x, [1])] \\ & release(x, [1]) \end{array}\right] \end{array}\right]$$

Figure 2: VoxML affordance substructure for a [[BLOCK]] voxeme. The reentrancy index [1] refers to another attribute of the block, here its semantic "head" or physical extent. [2] refers to a habitat constraint.

Krishnaswamy (2017) presents the realization of events in simulation as composed *primitives*, e.g., [[MOVE]], [[TURN-TO]], [[TURN-ABOUT-AXIS]], [[GRASP]], [[HOLD]], [[UNGRASP]], and [[RELEASE]]. Because these event programs are necessarily underspecified (i.e., "move to where?"), the complete operationalization of an event relies on parameters that are inferred from the composition of the arguments with the program (see Sec. 4.1).

VoxSim provides a mechanism to do this composition in real time, and accommodates symbolic and logical methods such as qualitative calculi (e.g., Allen (1983), Randell et al. (1992), Gatsoulis et al. (2016)), or machine learning for automated inference (e.g., Brockman et al. (2016), Raffin et al. (2019)). Such 3rd-party resources are easily integrated via TCP sockets, REST connections, or direct integration through Unity plugins. Fig. 3 shows the high-level architecture of VoxWorld, absent any specific agent implementation.

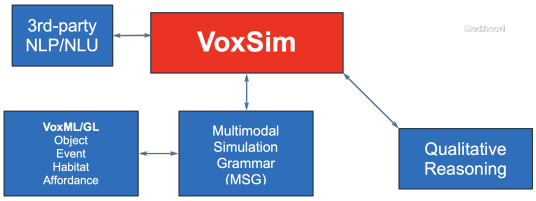

Figure 3: VoxWorld Platform Architecture

The agent then becomes an executor of events, which can involve manipulating objects, moving about the world, or even participating in dialogue. Event execution is handled by the *event manager*, which composes, interprets, grounds, directs, and monitors events frame by frame, testing for completion and satisfaction.

4.1. Action and Event Composition

The composition of events is driven by the semantics of VoxML. VoxML was designed to capture event semantics in a form that can be realized as a minimal model in a dynamic logic (Pustejovsky and Moszkowicz, 2011) while also providing space to draw more specific information from the environment itself where necessary a la Brooks (1991). This section details how the theoretical foundations of VoxML were translated into a robust working system that accommodates arbitrary events given a well-formed representation, and the evolution of event compositionality in implementation from our initial efforts in Krishnaswamy and Pustejovsky (2016) to the present.

Initially in VoxSim, all input was required to be typed English in the imperative mood, which was part-of-speech tagged, dependency-parsed, and transformed into a simple predicate-logic format denoting an action and its arguments, e.g., $put(the(black(block)), on(the(white(block))))$. This format still forms the core of VoxSim processing, but where previously, each individual predicate in the parsed input had to be operationalized directly in C# for execution in Unity, event composition has since evolved to function directly from the VoxML semantics, which breaks down complex predicates into compositions of the aforementioned primitives. In the process, VoxML specifications of events have been

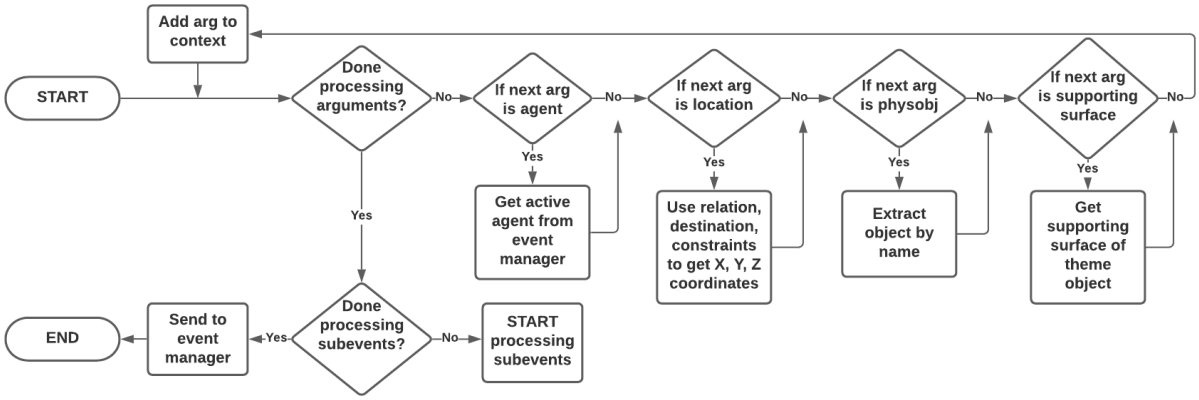

Figure 4: Flow of control when composing a program.

changed from strongly-typed feature structures to more of a duck-typed language like Python, wrapped in an attribute-value matrix (AVM)-like data structure. This makes development in VoxWorld more tractable for NLP researchers, who are likely to be more familiar with Python than C#.

Composing events, and interpreting and grounding predicates more generally, makes extensive use of *reflection*, a process that allows managed code to read its own metadata, and therefore find assemblies and execute methods by invoking them by name rather than as declared variables. Reflection also allows us to, rather than invoking a specific hard-coded name, invoke other methods that handle the interpretation of events using both VoxML semantics and the current state of the world, that in turn invoke handler methods that ground objects, attributes, relations, and functions to the current context. Fig. 4 shows a high-level diagram of the flow of control when composing an event program from VoxML semantics, including where information is incorporated from the world.

For example, a [[PUT]] voxeme may have the following subevent structure, with arguments x, the agent, y, the object, and z, the destination, and conditional operators $while$ and if (Fig. 5):

$$\left[\begin{array}{l}\textbf{put} \\ \text{BODY} = \left[\begin{array}{l}\text{E}_1 = grasp(x,y) \\ \text{E}_2 = while(hold(x,y) \wedge \neg at(y,z)) \\ \qquad \rightarrow move(x,y,z,\textbf{P()},[loc(y),z,y]) \\ \text{E}_3 = if(at(y,z) \rightarrow ungrasp(x,y)\end{array}\right]\end{array}\right]$$

Figure 5: Subevent structure of [[PUT]].

E_2 invokes the [[MOVE]] primitive such that agent x executes an instruction to move y to z. P() is a pointer to an optional function. The semantics of [[MOVE]] in simulation assumes this to be a path planner, that takes the subsequent array (the location of y, the destination, and y itself) as additional arguments. If no planner is specified, the object is moved directly in a straight line without regard for obstacles. VoxSim comes with an A* path planner available for use, but developers may supply their own through a C# class or any other endpoints that can supply a FIFO or indexable data structure of 3D waypoints.

4.2. Reasoning about Results and States

For an agent to reason about the world it inhabits, it must be able to track relations and changes of state that come to exist as the result of acting upon the environment. For instance, if "put the black block on the white block" is parsed and executed according to the process outlined in Sec. 4.1, the result of the event changes the state of the environment such that *white block* supports *black block*. This change of state is recorded in the affordances of [[BLOCK]] (see Fig. 2).

VoxWorld comes with a basic level of compositional spatial reasoning capabilities, based on various qualitative reasoning calculi (e.g., Randell et al. (1992), Balbiani et al. (1998), Moratz et al. (2002)), and through the external endpoint connections, additional reasoning or inference clients can be integrated. These may be either symbolic or machine learning-driven, and the output of any reasoner may be interpreted in terms of VoxML semantics as in Fig. 6.

$$\left[\begin{array}{l}\textbf{in\_front} \\ \text{TYPE} = \left[\begin{array}{l}\text{CLASS} = \textbf{config} \\ \text{VALUE} = RCC8.EC \\ \text{ARGS} = \left[\begin{array}{l}\text{A}_1 = \textbf{x:physobj} \\ \text{A}_2 = \textbf{y:physobj}\end{array}\right] \\ \text{CONSTR} = Z(x) > Z(y)\end{array}\right]\end{array}\right]$$

Figure 6: Type structure of [[IN_FRONT]].

When building embodied agents that can both query and interpret relational predicates, relations in situated environments pose a unique problem. Formally, relations are first-order functions over multiple arguments that return a boolean, e.g., [*is white block in front of black block?*] $\rightarrow$ {TRUE, FALSE}. In affordance and relation composition, VoxWorld treats relations with multiple arguments as a testable proposition like this, but in a situated context with linguistic grounding, relations also have another interpretation as locations or regions, exemplified by the nested predicate $in\_front(the(x))$. In this case, we treat the relational predicate as demanding interpretation as a *causal result* of satisfying the propositional equivalent, and return an element $\mathcal{R} \in \mathbb{R}^3$ denoted by a configurational relation, therefore $in\_front(x)$ becomes $IN(x, \mathcal{R})$, which can be left as a testable proposition for any to-be-specified

object x. Force dynamic relations like [[SUPPORT]] likewise trigger operations in the Unity physics engine. Using the same VoxML semantics for both types of relational computations allows a VoxWorld agent to easily perform both execution ("put the white block in front of the black block") and recognition ("where is the black block?").

This also makes changing the frame of reference simple: the inequality in the CONSTR of Fig. 6 just needs to be flipped to accommodate the opposite perspective. This can been done by directly mutating the encoding in the VoxML library, or though symbolic or machine learning methods triggered at runtime through dialogue, for example, if a human directs an agent to "put the white block in front of the black block" followed by "no, on the other side." Thereafter the agent can use the frame of reference adopted by its human partner.

4.3. Blackboard Architecture and PDA

Managing the inputs and dialogue state for even a unimodal agent is no trivial task. In speech alone, humans may jump back and forth across established context; managing multimodal inputs is an order of magnitude more complex as each modality may prompt a different kind of state update simultaneously. A particular point of user frustration with interactive agents comes when the agent falls into a rigid turn-taking pattern, and does not permit interruption or redirection (Kozierok et al., 2021). Human communication is asynchronous; we use multiple modalities in conjunction, and keep track of interlocutors' states while continuing the interaction. Therefore robust interactive agents should also have some of this capacity.

To illustrate, consider the following scenario (see Fig. 8). The agent should be able to take initiative to act upon partial information (e.g., "put the yellow block there," followed by pointing), and so it may start by picking up a yellow block in the environment and moving it in the direction of deixis even if the destination is not certain. At any point, though, the human partner may change their mind or decide that the agent needs to be corrected (e.g., "no, on the white one").

There is no doubt that a person would be able to follow these instructions, and an interactive agent should support the same. Therefore in the process of developing various interactive agents and their respective dialogues, we incorporated two relatively old but powerful ideas from early AI.

The first is a blackboard. Proposed in the 1970s for the Hearsay NLU system (Erman et al., 1980), blackboard architectures facilitate asynchronous updates from arbitrary knowledge sources that are managed by a control shell. Our blackboard was originally developed specifically for the Diana agent (Sec. 5.1) but has now been incorporated into VoxWorld directly as a convenient general-purpose data structure for managing third-party inputs. It is a strongly-typed key-value store in a modified singleton pattern where the control shell allows member functions of subclasses of the blackboard's `ModuleBase` class to subscribe to any or all keys on the blackboard and execute upon changes to the associated values (Strout, 2020). Fig. 7 shows the blackboard integration with knowledge sources used by the Diana agent (Sec. 5.1). Similar endpoints are used for the Kirby agent. The "Event and Dialogue Management" box connects to the rest of VoxSim (the purple box) via the event manager. Orange boxes denote agent outputs. Red boxes denote custom recognizers. Green boxes denote 3rd-party recognizers.

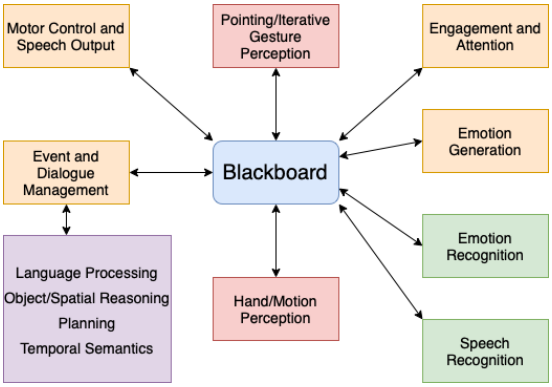

Figure 7: Blackboard integration with endpoints for the Diana agent.

The second is a variant on a pushdown automaton (PDA), used for higher-level dialogue management. The states are coarse-grained states in the dialogue (e.g., *is the agent engaged with a human?*, *is the agent answering a question?*, *is the agent learning something new?*, etc.), and the transition relation between states may be dictated by a custom-defined stack symbol class as suits the developer's needs, or may even exploit the entire blackboard for use as a stack symbol. Since the size of the stack symbol may be arbitrarily large, we write the transition relation in turns of satisfiable predicates. Therefore, an arbitrary condition may be defined in the transition relation, and a stack symbol that satisfies that condition may be used to trigger the associated state transition. In doing so, we exploit a continuation-passing style semantics (Van Eijck and Unger, 2010) to facilitate the asynchronous exchange of information across the blackboard while maintaining the separation of different high-level dialogue states (i.e., the agent must note and store various pieces of information before switching states).[1] When the PDA enters a new state, an equivalently-named function is executed if one exists. Any change that is to be effected on the agent's world can be written into these functions. More details on this portion of the architecture can be found in Krishnaswamy and Pustejovsky (2019b).

VoxWorld developers can instantiate new blackboard keys by simply writing a new arbitrary string key and

[1]This functional design pattern is not compatible with web builds (Sec. 5.1.2).

value type to the blackboard and then subscribing functions to changes on that key. They can also create custom PDAs by typing the contents of their stack symbol and then adding states and transition relations to their PDA class. Functions that are called upon changes to the blackboard or PDA can thereafter access the agent and the world to drive interaction.

5. Agent Implementations

In this section we detail 3 specific agents that use the VoxWorld platform, accompanying agent architectures, and underlying semantics. These agents all have different capabilities and use cases, and demonstrate how VoxWorld can use the same framework to create diverse agents. We enumerate key lessons learned in the development of each agent type that shaped the development of VoxWorld as a platform.

5.1. Diana

Diana (Krishnaswamy et al., 2017; Narayana et al., 2018; McNeely-White et al., 2019; Krishnaswamy et al., 2020a; Krishnaswamy et al., 2020b) is an interactive multimodal agent intended to develop and test peer-to-peer human-computer communication. With multiple modalities even a simple use case like Blocks World presents a challenge of extracting and integrating meaning from each modality. These are the challenges that VoxSim, and later VoxWorld, was explicitly designed to address. Advancing the state of the art in peer-to-peer human-computer communication necessarily entailed a deep study of how humans conduct the same task. Therefore, the set of gestures that Diana interprets were derived from EGGNOG, a dataset of human-human interactions in a shared Blocks World construction task (Wang et al., 2017).

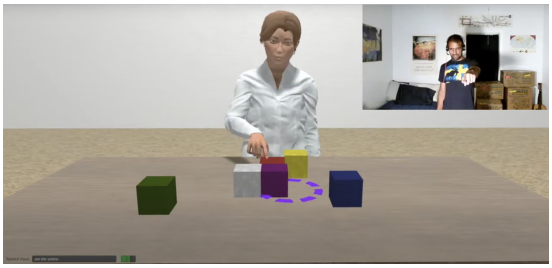

Figure 8: Diana reaches for a block to demonstrate an interpretation of the person's deixis.

Diana provided the first use case of an agent that could not be isolated to the Unity environment; an agent designed to interpret gesture and speech must have access to those inputs. The gestures of Diana's human interlocutor are recognized via custom deep CNNs over depth video. Speech recognition is currently handled using Google Cloud ASR. Outputs from these endpoints are posted to the blackboard for processing, facilitating integration of diverse multimodal inputs such as new gesture or speech engines.

One key lesson learned during the development of Diana was that it is not simply enough to give an agent access to multimodal sensor data; it must know how to react to those inputs even with incomplete information. For example, if the human indicates an object but not what to do with it, if Diana receives that information, she must react in a way that demonstrates that receipt. If she does not react, the system has no explicit way of moving the interaction forward. Fig. 8 shows this, where the human points to a the purple block, and Diana demonstrates her understanding by reaching for the purple block in turn.[2]

Diana has been used for studying referring expressions (Krishnaswamy and Pustejovsky, 2019a; Krishnaswamy and Pustejovsky, 2020), human-computer interaction (Pustejovsky and Krishnaswamy, 2021b; Krishnaswamy and Pustejovsky, 2021), and object affordances (Pustejovsky and Krishnaswamy, 2021a).

5.1.1. Diana Evaluation

Diana was evaluated in the summer of 2021 with the aim of assessing whether naive users could interact with her to build specific block structures without prior instruction. We assessed our evaluation based on task completion rate and user satisfaction according to the System Usability Scale (SUS), a common HCI metric (Brooke, 1996; Bangor et al., 2008).

Thirty subjects, evenly divided between men and women, aged 18-57 ($\mu = 27, \sigma = 11.8$) participated in the evaluation. Users were asked to collaborate with Diana to build a variety of block structures with a 10-minute time limit.

Diana achieved a high task completion rate of 90%. Out of 240 total trials, only 24 could not be completed within the time limit. According to the SUS, 68 is considered "average" and 80 or above is considered "excellent." Diana achieved a mean SUS of 74.3 ($\sigma = 8.2$), with scores ranging from 67.5-90. Only four scores with a 67.5 missed the "average" mark of 68 and eight SUS scores (27% of the total) received scores above 80. Qualitative feedback from participants was also positive and highlighted the multimodal aspect of the interaction, e.g., "the combination of speech and gesture at the same time was useful and unique."

5.1.2. Diana in a Web Browser

The fully-featured Diana system relies on some specialized hardware, such as Kinect cameras, but the underlying interactive mechanisms can be used independently of these. We subsequently took the Diana system and deployed a version that can run natively in a web browser, using the mouse for deixis and the WebSpeech API (Adorf, 2013) for speech recognition. By simply switching out the endpoints, we keep the core interaction between Diana and the human intact. The same control processing shown in Fig. 4 and the same underlying VoxML semantics drive the dialogue.

The web-deployable version of Diana uses Unity's WebGL build functionality. As such, the VoxWorld plat-

[2] A video of Diana can be viewed here.

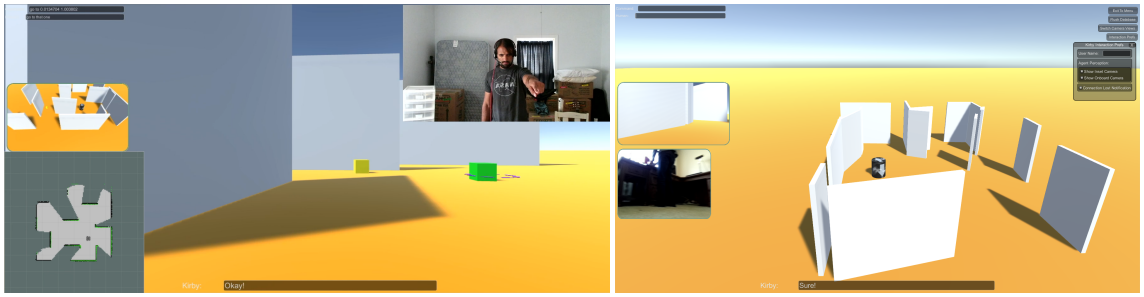

Figure 9: [L] Main: Kirby's view of virtually-rendered environment; left inset: omniscient view; bottom left: Gazebo simulator; top right: human instructing Kirby. [R] Main: Omniscient view of Kirby's virtual environment with Kirby avatar centered; left inset (top): Kirby's rendered perspective; left inset (bottom): Kirby's camera view.

form needed to be made maximally compatible with the restrictions of WebGL. This involved 3 key modifications:

1. Web builds restrict access to external files and directory trees. The solution is to compile all required VoxML encoding files and directories directly into the final binary as Unity resources.
2. No just-in-time (JIT) code is allowed by WebGL, so the functional PDA architecture, which encodes stack symbol states in the transition relation as satisfiable predicates, cannot be accommodated, and so must be deactivated for web builds. The blackboard architecture, which is compiled ahead-of-time (AoT), can still be used, so all behaviors need to be written using the blackboard.
3. Certain 3rd-party dependencies which VoxWorld initially incorporated are not compatible with WebGL (e.g., Google ASR), therefore, removing 3rd-party dependencies from the core VoxWorld platform and leaving their inclusion to the discretion of individual agent developers is key (see Sec. 7)

5.2. Kirby

An embodied agent in a purely virtual world can directly access parameters of the world, such as exact locations of objects. Therefore the real challenge for embodied agent behaviors comes when they must be enacted in the real world where inference is noisier.

Krajovic et al. (2020) presented a prototype of a mobile robotic agent built on the VoxWorld platform. This agent, Kirby, uses the same gesture and speech recognition components as Diana, but exists not only in a virtual world, but as a real mobile robot (specifically a modified GoPiGo3 outfitted with a LIDAR) using the common Robot Operating System (ROS).

Kirby functions as an agent that can navigate through locations where the human is not physically present or cannot go. As Kirby navigates its environment, it builds up a coherent model of obstacles in the environment using the LIDAR data, and of items in the environment using object or fiducial detection from its onboard camera. The human can point and gesture to guide Kirby (e.g., beckoning for Kirby to move toward the human), along with speech (e.g., "go there", "go to the green one", etc.), and can use gestures to change camera views (through swiping) or rotate the camera in three dimensions (through iterative directional gestures). Fig. 9 depicts Kirby's interface. We use a Redis store connected via VoxSim's socket connections to exchange messages with the ROS client running on the robot, and the blackboard manages speech and gesture inputs while the PDA manages dialogue state.

One key lesson learned in building Kirby was the ease with which Diana's blackboard architecture can be used to manage inputs, as demonstrated by the reuse of Diana's gesture set in the Kirby use case. This prompted the integration of the blackboard architecture, initially created for Diana, into VoxWorld directly. An evaluation of Kirby was planned but had to be canceled due to the COVID-19 global pandemic and an inability to hold trials with in-person subjects.

5.3. BabyBAW

Diana and Kirby are deterministic agents. Their behaviors, while customizable, are programmed for known use cases. A known set of inputs will lead to a known set of outputs or actions. However, 3D environments and embodied simulation are also very useful for exploration and learning through interaction with the world. This is a common area of research in reinforcement learning (Aluru et al., 2015; Kolve et al., 2017; Savva et al., 2017; Juliani et al., 2019) and developmental psychology (Battaglia et al., 2013; Ullman et al., 2017).

BabyBAW is an agent that learns through interaction with the environment. It can be initialized with different levels of underlying knowledge (e.g., knowledge of gravity, different object properties, or different actions), and given tasks to test what it can accomplish. It uses neural networks, symbolic reasoning, and embodied simulation for their respective strengths ("*B*est of *A*ll *W*orlds") to approximate certain aspects of infant and child learning (Hartshorne and Pustejovsky, 2021).

Learning from exploration and interaction is an obvious problem for reinforcement learning (RL). To develop a VoxWorld agent for RL, we focused on integrating two common, well-developed RL platforms: Unity ML-Agents (Juliani et al., 2018) and OpenAI Gym (Brockman et al., 2016). The goal here was to make building environments and tasks for BabyBAW and testing multiple environmental and architectural configurations as

simple and rapid as possible. BabyBAW agent behaviors interface directly with the Unity ML-Agents API and the VoxWorld event management, relational reasoning, and interactive architecture components.

Our current experiments in BabyBAW are based on explorations of infant intuition about objects and support relations from developmental psychology. At slightly more than 6 months old, most infants appear able to intuit than an object will not fall if supported from the bottom on over 50% of its lower surface (Baillargeon et al., 1992; Dan et al., 2000; Huettel and Needham, 2000; Spelke and Kinzler, 2007). Therefore, an RL algorithm should be able to solve for a policy that resembles this intuition in a stacking task.

Due to the popularity of OpenAI Gym and increasing adoption of Unity ML-Agents, successful VoxWorld integration with these tools allows the use in turn of other packages that are compatible with them. In the current work, we use the Stable-Baselines3 package (Raffin et al., 2019), a set of reliable implementations of RL algorithms written in PyTorch. For a continuous action space, we use a DDPG or TD3 algorithm (Lillicrap et al., 2016; Fujimoto et al., 2018) and explore training an agent to stack objects in a 3D world (Fig. 10). We evaluate BabyBAW in this task in Sec. 5.3.1

Figure 10: Unsuccessful stacking (left and middle) and successful stacking (right) during training.

Learning to stack is, of course, not a novel task in the RL community (cf. Lerer et al. (2016), Li et al. (2017), Li et al. (2020), Hundt et al. (2020), just to name a few), and our intention is not to propose new RL algorithms but rather to present a novel platform that can exploit what is already available to rapidly model and test cognitive theories, as discussed next.

5.3.1. BabyBAW Evaluation

We evaluated BabyBAW in VoxWorld to test its ability to learn a stacking task, given the ability to extract measurements of certain concepts from the environment. We evaluate two agents: one trained with the mechanism to measure height and center of gravity, and a baseline without those capabilities.

Our initial evaluation uses a 2D action that corresponds to a location in 3D space calculated relative to the surface of the destination block. The optimal solution places the theme object exactly centered atop the destination block and the agent must solve for an action that corresponds to that event in VoxWorld. To increase the problem complexity, we can make the action space (when rescaled in the 3D environment) arbitrarily large so that the optimal solution lies in a very small section of the rescaled action space, and can perturb the action space through VoxWorld so that the optimal solution may not lie in the exact center of the action space. Each action (attempt to stack) is one timestep and a max of 10 timesteps are allowed per episode. The agent receives a negative reward for missing the destination block entirely, a small positive reward for touching the destination block with the theme block even if it falls off, and a large positive reward for stacking successfully, with a 10% decay on each additional attempt.

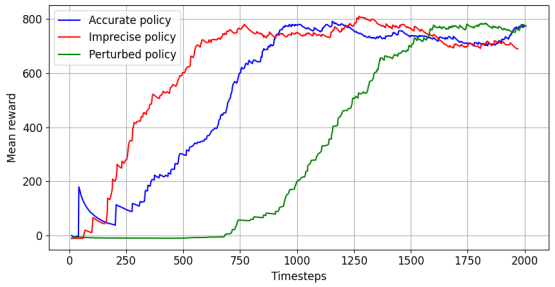

Figure 11: Episode mean reward vs. training time.

Fig. 11 shows the training reward plots for three BabyBAW stacking policies. The observation space is defined by the height of the stack and center of gravity of the stack relative to that of the bottom object. The blue plot (the *accurate policy* is well-optimized), while the red plot (the *imprecise policy*) is less so. In the green plot, where the reward starts climbing around timestep 350, the action space was perturbed so the optimal policy is far from the center of the action space, to test the algorithm's ability to generalize. Max reward is 1000, and policies were trained for 2000 timesteps.

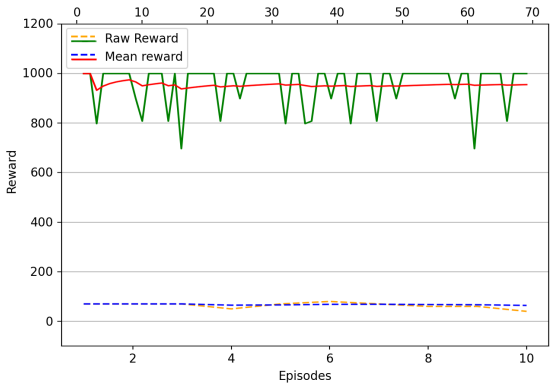

Figure 12: Reward vs. evaluation episodes for BabyBAW and baseline. Episodes terminate upon successful stacking, so more episodes elapse in the 100 testing timesteps using the trained model than the baseline.

When BabyBAW is given the mechanism to extract veridical knowledge of height and center of gravity from the environment, it can rapidly solve a stacking task in real time (∼15 mins.), without even speeding up rendering. Fig. 12 shows the reward and cumulative mean reward per episode for a trained model (solid red and green lines) vs. the baseline trained without the height and center of gravity information (dashed blue

and orange lines). A mean reward close to 1000 means BabyBAW frequently stacked the blocks on the first try. Ongoing BabyBAW work involves learning about object properties from differences in their behavior (e.g., what happens when BabyBAW tries to stack a sphere on top of a cube?), and correlating those differences in behavior to novel object classes and linguistic labels.

6. Timeline of Selected Publications

This section provides a brief list of selected prior VoxWorld-related publications.

- LREC 2016: (Pustejovsky and Krishnaswamy, 2016). First publication of the VoxML modeling language.
- COLING 2016: (Krishnaswamy and Pustejovsky, 2016). First demonstration of the VoxSim software.
- IWCS 2017: (Krishnaswamy et al., 2017). First publication of the Diana agent.
- IntelliSys 2018: (Narayana et al., 2018). Diana agent with integrated gesture and speech.
- IEEE HCC 2019: (McNeely-White et al., 2019). "Modern" Diana agent with blackboard architecture.
- AAAI 2020: (Krishnaswamy et al., 2020b). Public demo of the Diana agent.
- RoboDial 2020: (Krajovic et al., 2020). First publication of the Kirby agent.
- CogSci 2022: (Krishnaswamy and Ghaffari, 2022). BabyBAW agent used for novel concept detection.

7. Conclusion

Interaction takes many forms. VoxWorld is intended to accommodate as many as possible with an extensible, event-centric language semantics and straightforward pipeline. In this paper we detailed how VoxWorld evolved from the theoretical VoxML framework into a distinct platform targeted to developers of embodied reasoning agents, and discussed key lessons learned through the development of different agent types. One final takeaway is the importance of keeping VoxWorld independent of 3rd-party packages, thus allowing developers to use their preferred methods for things like animation, speech recognition or text-to-speech. We have refactored the VoxWorld API to allow developers to incorporate their own endpoints as universally as possible, with a combination of C# interface classes and event handlers, allowing us to make stable builds of VoxWorld available as a single Unity package.

The bleeding edge version of the source code is at `https://github.com/VoxML/VoxSim`, and we have created a sample project with a simple interaction to let interested researchers get started quickly at `https://github.com/VoxML/VoxWorld-QS`. Online documentation is under construction at `https://www.voxicon.net/api/`.

8. Acknowledgements

Special thanks to Dr. Jaime Ruiz (University of Florida) for conducting the Diana evaluation. Thanks to Sadaf Ghaffari for providing Figure 11. We would also like to thank the reviewers for their helpful comments. This work was supported in part by the U.S. Defense Advanced Research Projects Agency (DARPA) and the Army Research Office (ARO) under contract W911NF-15-C-0238 at Brandeis University and contract W911NF-15-1-0459 at Colorado State University, and by the National Science Foundation (NSF) under grant numbers CNS 2033932 and DRL 2019805 to Brandeis University and DRL 1559731 to Colorado State University. The views expressed are those of the authors and do not reflect the official policy or position of the U.S. Government. All errors and mistakes are, of course, the responsibilities of the authors.

9. Bibliographical References

Abrami, G., Henlein, A., Kett, A., and Mehler, A. (2020). Text2scenevr: Generating hypertexts with vannotator as a pre-processing step for text2scene systems. In *Proceedings of the 31st ACM Conference on Hypertext and Social Media*, pages 177–186.

Adorf, J. (2013). Web speech api. *KTH Royal Institute of Technology*.

Allen, J. F. (1983). Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843.

Allouch, M., Azaria, A., and Azoulay, R. (2021). Conversational agents: Goals, technologies, vision and challenges. *Sensors*, 21(24):8448.

Aluru, K. C., Tellex, S., Oberlin, J., and MacGlashan, J. (2015). Minecraft as an experimental world for ai in robotics. In *2015 aaai fall symposium series*.

Anderson, J. R., Matessa, M., and Lebiere, C. (1997). Act-r: A theory of higher level cognition and its relation to visual attention. *Human–Computer Interaction*, 12(4):439–462.

Anscombe, G. E. M. (1957). *Intention*. Oxford: Basil Blackwell.

Baillargeon, R., Needham, A., and DeVos, J. (1992). The development of young infants' intuitions about support. *Early development and parenting*, 1(2):69–78.

Balbiani, P., Condotta, J.-F., and Del Cerro, L. F. (1998). A model for reasoning about bidimensional temporal relations. In *PRINCIPLES OF KNOWLEDGE REPRESENTATION AND REASONING-INTERNATIONAL CONFERENCE-*, pages 124–130. Citeseer.

Bangor, A., Kortum, P. T., and Miller, J. T. (2008). An empirical evaluation of the system usability scale. *Intl. Journal of Human–Computer Interaction*, 24(6):574–594.

Battaglia, P. W., Hamrick, J. B., and Tenenbaum, J. B. (2013). Simulation as an engine of physical scene understanding. *Proceedings of the National Academy of Sciences*, 110(45):18327–18332.

Beßler, D., Porzel, R., Pomarlan, M., Beetz, M., Malaka, R., and Bateman, J. (2020). A formal model of affordances for flexible robotic task execution. In *ECAI 2020*, pages 2425–2432. IOS Press.

Bonn, J., Palmer, M., Cai, J., and Wright-Bettner, K. (2020). Spatial amr: Expanded spatial annotation in the context of a grounded minecraft corpus. In *Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020),*.

Breazeal, C., Brooks, A., Gray, J., Hoffman, G., Kidd, C., Lee, H., Lieberman, J., Lockerd, A., and Mulanda, D. (2004). Humanoid robots as cooperative partners for people. *Int. Journal of Humanoid Robots*, 1(2):1–34.

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

Brooke, J. (1996). Sus-a quick and dirty usability scale. *Usability evaluation in industry*, 189(194):4–7.

Brooks, R. A. (1991). Intelligence without representation. *Artificial intelligence*, 47(1-3):139–159.

Cohen, P. (2017). Context in communication. In *2017 AAAI Spring Symposium Series*.

Dan, N., Omori, T., and Tomiyasu, Y. (2000). Development of infants' intuitions about support relations: Sensitivity to stability. *Developmental Science*, 3(2):171–180.

Davidson, D. (1963). Actions, reasons, and causes. *The journal of philosophy*, 60(23):685–700.

Dennett, D. C. (1987). *The intentional stance*. MIT press.

Dzifcak, J., Scheutz, M., Baral, C., and Schermerhorn, P. (2009). What to do and how to do it: Translating natural language directives into temporal and dynamic logic representation for goal management and action execution. In *2009 IEEE International Conference on Robotics and Automation*, pages 4163–4168. IEEE.

Erman, L. D., Hayes-Roth, F., Lesser, V. R., and Reddy, D. R. (1980). The hearsay-ii speech-understanding system: Integrating knowledge to resolve uncertainty. *ACM Computing Surveys (CSUR)*, 12(2):213–253.

Ferguson, G., Allen, J. F., et al. (1998). Trips: An integrated intelligent problem-solving assistant. In *Aaai/Iaai*, pages 567–572.

Fischer, L., Hasler, S., Deigmöller, J., Schnürer, T., Redert, M., Pluntke, U., Nagel, K., Senzel, C., Ploennigs, J., Richter, A., et al. (2018). Which tool to use? grounded reasoning in everyday environments with assistant robots. In *CogRob@ KR*, pages 3–10.

Fujimoto, S., Hoof, H., and Meger, D. (2018). Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pages 1587–1596. PMLR.

Gatsoulis, Y., Alomari, M., Burbridge, C., Dondrup, C., Duckworth, P., Lightbody, P., Hanheide, M., Hawes, N., Hogg, D., Cohn, A., et al. (2016). Qsrlib: a software library for online acquisition of qualitative spatial relations from video.

Gibson, J. J. (1977). The theory of affordances. *Hilldale, USA*, 1(2):67–82.

Hartshorne, J. and Pustejovsky, J. (2021). A playground and proposal for growing an artificial general intelligence. Technical report, Waltham, MA.

Henlein, A., Abrami, G., Kett, A., and Mehler, A. (2020). Transfer of isospace into a 3d environment for annotations and applications. In *16th Joint ACL-ISO Workshop on Interoperable Semantic Annotation PROCEEDINGS*, pages 32–35.

Huettel, S. A. and Needham, A. (2000). Effects of balance relations between objects on infant's object segregation. *Developmental Science*, 3(4):415–427.

Hundt, A., Killeen, B., Greene, N., Wu, H., Kwon, H., Paxton, C., and Hager, G. D. (2020). "good robot!": Efficient reinforcement learning for multi-step visual tasks with sim to real transfer. *IEEE Robotics and Automation Letters*, 5(4):6724–6731.

Juliani, A., Berges, V.-P., Teng, E., Cohen, A., Harper, J., Elion, C., Goy, C., Gao, Y., Henry, H., Mattar, M., et al. (2018). Unity: A general platform for intelligent agents. *arXiv preprint arXiv:1809.02627*.

Juliani, A., Khalifa, A., Berges, V.-P., Harper, J., Teng, E., Henry, H., Crespi, A., Togelius, J., and Lange, D. (2019). Obstacle tower: A generalization challenge in vision, control, and planning. *arXiv preprint arXiv:1902.01378*.

Kazhoyan, G. and Beetz, M. (2019). Executing underspecified actions in real world based on online projection. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5156–5163. IEEE.

Kempka, M., Wydmuch, M., Runc, G., Toczek, J., and Jaśkowski, W. (2016). Vizdoom: A doom-based ai research platform for visual reinforcement learning. In *2016 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 1–8. IEEE.

Kolve, E., Mottaghi, R., Han, W., VanderBilt, E., Weihs, L., Herrasti, A., Gordon, D., Zhu, Y., Gupta, A., and Farhadi, A. (2017). Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*.

Kozierok, R., Aberdeen, J., Clark, C., Garay, C., Goodman, B., Korves, T., Hirschman, L., McDermott, P. L., and Peterson, M. W. (2021). Hallmarks of human-machine collaboration: A framework for as-

sessment in the darpa communicating with computers program. *arXiv preprint arXiv:2102.04958*.

Krajovic, K., Krishnaswamy, N., Dimick, N. J., Salas, R. P., and Pustejovsky, J. (2020). Situated multimodal control of a mobile robot: Navigation through a virtual environment. *arXiv preprint arXiv:2007.09053*.

Krishnamurthy, J. and Kollar, T. (2013). Jointly learning to parse and perceive: Connecting natural language to the physical world. *Transactions of the Association for Computational Linguistics*, 1:193–206.

Krishnaswamy, N. and Ghaffari, S. (2022). Exploiting embodied simulation to detect novel object classes through interaction. *arXiv preprint arXiv:2204.08107*.

Krishnaswamy, N. and Pustejovsky, J. (2016). Voxsim: A visual platform for modeling motion language. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations*, pages 54–58.

Krishnaswamy, N. and Pustejovsky, J. (2019a). Generating a novel dataset of multimodal referring expressions. In *Proceedings of the 13th International Conference on Computational Semantics-Short Papers*, pages 44–51.

Krishnaswamy, N. and Pustejovsky, J. (2019b). Multimodal continuation-style architectures for human-robot interaction. *arXiv preprint arXiv:1909.08161*.

Krishnaswamy, N. and Pustejovsky, J. (2020). A formal analysis of multimodal referring strategies under common ground. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 5919–5927.

Krishnaswamy, N. and Pustejovsky, J. (2021). The role of embodiment and simulation in evaluating hci: Experiments and evaluation. In *International Conference on Human-Computer Interaction*. Springer.

Krishnaswamy, N., Narayana, P., Wang, I., Rim, K., Bangar, R., Patil, D., Mulay, G., Beveridge, R., Ruiz, J., Draper, B., et al. (2017). Communicating and acting: Understanding gesture in simulation semantics. In *IWCS 2017—12th International Conference on Computational Semantics—Short papers*.

Krishnaswamy, N., Beveridge, R., Pustejovsky, J., Patil, D., McNeely-White, D. G., Wang, H., and Ortega, F. R. (2020a). Situational awareness in human computer interaction: Diana's world.

Krishnaswamy, N., Narayana, P., Bangar, R., Rim, K., Patil, D., McNeely-White, D., Ruiz, J., Draper, B., Beveridge, R., and Pustejovsky, J. (2020b). Diana's world: A situated multimodal interactive agent. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 13618–13619.

Krishnaswamy, N. (2017). *Monte Carlo Simulation Generation Through Operationalization of Spatial Primitives*. Ph.D. thesis, Brandeis University.

Kuhl, P. K. (1998). Language, culture and intersubjectivity: The creation of shared perception. *Intersubjective communication and emotion in early ontogeny*, (3):297.

Laird, J. E. (2019). *The Soar cognitive architecture*.

Lerer, A., Gross, S., and Fergus, R. (2016). Learning physical intuition of block towers by example. In *International conference on machine learning*, pages 430–438. PMLR.

Li, W., Bohg, J., and Fritz, M. (2017). Acquiring target stacking skills by goal-parameterized deep reinforcement learning. *arXiv preprint arXiv:1711.00267*.

Li, R., Jabri, A., Darrell, T., and Agrawal, P. (2020). Towards practical multi-object manipulation using relational reinforcement learning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4051–4058. IEEE.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2016). Continuous control with deep reinforcement learning. In *ICLR (Poster)*.

McNeely-White, D. G., Ortega, F. R., Beveridge, J. R., Draper, B. A., Bangar, R., Patil, D., Pustejovsky, J., Krishnaswamy, N., Rim, K., Ruiz, J., et al. (2019). User-aware shared perception for embodied agents. In *2019 IEEE International Conference on Humanized Computing and Communication (HCC)*, pages 46–51. IEEE.

Miller, G. A. and Johnson-Laird, P. N. (2013). *Language and perception*. Harvard University Press.

Mooney, R. J. (2008). Learning to connect language and perception. In *AAAI*, pages 1598–1601.

Moratz, R., Nebel, B., and Freksa, C. (2002). Qualitative spatial reasoning about relative position. In *International Conference on Spatial Cognition*, pages 385–400. Springer.

Narayana, P., Krishnaswamy, N., Wang, I., Bangar, R., Patil, D., Mulay, G., Rim, K., Beveridge, R., Ruiz, J., Pustejovsky, J., et al. (2018). Cooperating with avatars through gesture, language and action. In *Proceedings of SAI Intelligent Systems Conference*, pages 272–293. Springer.

Pecher, D. and Zwaan, R. A. (2005). *Grounding cognition: The role of perception and action in memory, language, and thinking*. Cambridge University Press.

Pomarlan, M. and Bateman, J. A. (2020). Embodied functional relations: A formal account combining abstract logical theory with grounding in simulation. In *FOIS*, pages 155–168.

Pustejovsky, J. and Krishnaswamy, N. (2016). Voxml: A visualization modeling language. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 4606–4613.

Pustejovsky, J. and Krishnaswamy, N. (2019). Situational grounding within multimodal simulations. *arXiv preprint arXiv:1902.01886*.

Pustejovsky, J. and Krishnaswamy, N. (2020). Em-

bodied human-computer interactions through situated grounding. In *Proceedings of the 20th ACM International Conference on Intelligent Virtual Agents*, pages 1–3.

Pustejovsky, J. and Krishnaswamy, N. (2021a). Embodied human computer interaction. *KI-Künstliche Intelligenz*, pages 1–21.

Pustejovsky, J. and Krishnaswamy, N. (2021b). The role of embodiment and simulation in evaluating hci: Theory and framework. In *International Conference on Human-Computer Interaction*. Springer.

Pustejovsky, J. and Moszkowicz, J. L. (2011). The qualitative spatial dynamics of motion in language. *Spatial Cognition & Computation*, 11(1):15–44.

Pustejovsky, J. (2013). Dynamic event structure and habitat theory. In *Proceedings of the 6th International Conference on Generative Approaches to the Lexicon (GL2013)*, pages 1–10.

Pylyshyn, Z. W. (1978). What has language to do with perception? some speculations on the lingua mentis. *American Journal of Computational Linguistics*, pages 79–86.

Quick, D. and Morrison, C. T. (2017). Composition by conversation. In *43rd International Computer Music Conference, ICMC 2017 and the 6th International Electronic Music Week, EMW 2017*, pages 52–57. Shanghai Conservatory of Music.

Raffin, A., Hill, A., Ernestus, M., Gleave, A., Kanervisto, A., and Dormann, N. (2019). Stable baselines3. *GitHub repository*.

Ramirez-Amaro, K., Yang, Y., and Cheng, G. (2019). A survey on semantic-based methods for the understanding of human movements. *Robotics and Autonomous Systems*, 119:31–50.

Randell, D. A., Cui, Z., and Cohn, A. G. (1992). A spatial logic based on regions and connection. *KR*, 92:165–176.

Rao, A. S., Georgeff, M. P., et al. (1995). Bdi agents: From theory to practice. In *Icmas*, volume 95, pages 312–319.

Rehm, M. and Goecke, K. U. (2000). Perception, concepts and language: Road and ipage. In *COLING 2000 Volume 2: The 18th International Conference on Computational Linguistics*.

Richard-Bollans, A. L. (2021). *Modelling the semantic variability of spatial prepositions in referring expressions*. Ph.D. thesis, University of Leeds.

Rodrigues, E. J., Santos, P. E., Lopes, M., Bennett, B., and Oppenheimer, P. E. (2020). Standpoint semantics for polysemy in spatial prepositions. *Journal of Logic and Computation*, 30(2):635–661.

Rohlfing, K. J., Rehm, M., and Goecke, K. U. (2003). Situatedness: The interplay between context (s) and situation. *Journal of Cognition and Culture*, 3(2):132–156.

Rosskamp, J., Weller, R., Kluss, T., Maldonado C, J. L., and Zachmann, G. (2020). Improved cnn-based marker labeling for optical hand tracking. In *International Conference on Virtual Reality and Augmented Reality*, pages 165–177. Springer.

Russell, S. and Norvig, P. (2002). Artificial intelligence: a modern approach.

Rusu, R. B., Marton, Z. C., Blodow, N., Dolha, M., and Beetz, M. (2008). Towards 3d point cloud based object maps for household environments. *Robotics and Autonomous Systems*, 56(11):927–941.

Savva, M., Chang, A. X., Dosovitskiy, A., Funkhouser, T., and Koltun, V. (2017). Minos: Multimodal indoor simulator for navigation in complex environments. *arXiv preprint arXiv:1712.03931*.

Savva, M., Kadian, A., Maksymets, O., Zhao, Y., Wijmans, E., Jain, B., Straub, J., Liu, J., Koltun, V., Malik, J., et al. (2019). Habitat: A platform for embodied ai research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9339–9347.

Scherer, S., Marsella, S., Stratou, G., Xu, Y., Morbini, F., Egan, A., Rizzo, A. S., and Morency, L.-P. (2012). Perception markup language: Towards a standardized representation of perceived nonverbal behaviors. In *International Conference on Intelligent Virtual Agents*, pages 455–463. Springer.

Schermerhorn, P. W., Kramer, J. F., Middendorff, C., and Scheutz, M. (2006). Diarc: A testbed for natural human-robot interaction. In *AAAI*, volume 6, pages 1972–1973.

Spelke, E. S. and Kinzler, K. D. (2007). Core knowledge. *Developmental science*, 10(1):89–96.

Strout, J. J. (2020). Multimodal agents for cooperative interaction. Master's thesis, Colorado State University.

Tamari, R., Shani, C., Hope, T., Petruck, M. R., Abend, O., and Shahaf, D. (2020). Language (re) modelling: Towards embodied language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6268–6281.

Tellex, S., Gopalan, N., Kress-Gazit, H., and Matuszek, C. (2020). Robots that use language. *Annual Review of Control, Robotics, and Autonomous Systems*, 3:25–55.

Thrun, S., Beetz, M., Bennewitz, M., Burgard, W., Cremers, A. B., Dellaert, F., Fox, D., Haehnel, D., Rosenberg, C., Roy, N., et al. (2000). Probabilistic algorithms and the interactive museum tour-guide robot minerva. *The International Journal of Robotics Research*, 19(11):972–999.

Ullman, T. D., Spelke, E., Battaglia, P., and Tenenbaum, J. B. (2017). Mind games: Game engines as an architecture for intuitive physics. *Trends in cognitive sciences*, 21(9):649–665.

Van Eijck, J. and Unger, C. (2010). *Computational semantics with functional programming*. Cambridge University Press.

Waltz, D. L. (1978). On the interdependence of lan-

guage and perception. *American Journal of Computational Linguistics*, pages 56–63.

Wang, I., Fraj, M. B., Narayana, P., Patil, D., Mulay, G., Bangar, R., Beveridge, J. R., Draper, B. A., and Ruiz, J. (2017). Eggnog: A continuous, multimodal data set of naturally occurring gestures with ground truth labels. In *2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017)*, pages 414–421. IEEE.

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