DIRT: The Distributed Intelligent Replicator Toolkit

Chengrui Wang[1,2], Chase Van Amburg[1], Chloe Huangyuan Su[1], Sham Kakade[1], Kiante Brantley[1], Aaron Walsman[1]

[1]Kempner Institute, Harvard University, USA
[2]Fudan University, China
aaronwalsman@fas.harvard.edu

Abstract

We introduce **DIRT**, the **D**istributed **I**ntelligent **R**eplicator **T**oolkit, a new evolutionary simulator designed to support GPU-accelerated massive populations and a large physical scale for artificial life research. DIRT is designed to simulate mobile agents in a natural environment that must consume resources in order to survive and reproduce. It uses highly configurable grid world dynamics that include multiple sensor types, complex terrain, water features, and a climate system. Agents in DIRT have traits that control their individual abilities. DIRT is interoperable with a wide variety of population algorithms that are able to produce policies and traits for new agents. DIRT is built on JAX and as a result, it can support populations of over 10,000 agents on a single GPU. We also provide a rich set of measurement tools and a 3D viewer which allows fine-grained inspection and tracking of individual agents.

Submission type: **Full Paper**

Introduction

The natural world is enormous. Researchers in biology and ecology have made important connections between the physical size of ecosystems and their resulting diversity and ecological dynamics (MacArthur and Wilson, 2001; Rosenzweig, 1995). Unfortunately for researchers studying these phenomena in simulated settings, computational requirements can force difficult decisions about the level of abstraction and fidelity with which a system is modeled. While many high-quality, large-scale simulation platforms exist (Luke et al., 2005; Railsback et al., 2006; Macal, 2016; Abar et al., 2017), recent software and hardware systems developed for AI training offer new tools for acceleration and scaling beyond what has previously been possible (Paszke, 2019; Jouppi et al., 2017; Bradbury et al., 2018; Rasley et al., 2020; Shoeybi et al., 2019; Brown et al., 2020; Chowdhery et al., 2023).

With this in mind, we have built a new distributed system to study populations of mobile intelligent agents in a simulated natural setting. Following recent trends in reinforcement learning and artificial life, we compute environment dynamics on the GPU, which allows for massive parallelization of computation and reduces communication overhead with neural network policies, dramatically speeding up the training/evolution loop. Our goal in designing DIRT is to construct an environment that is cognitively challenging yet computationally efficient enough to support fast iterations with very large population sizes. Given this objective, we embrace a relatively high degree of environmental abstraction and use discrete grid world dynamics in a 2D landscape. This allows us to take advantage of tensor-based computational tools developed for modern neural-network training workflows. The grid world abstraction has a long and rich history in Artificial Life and Reinforcement Learning, and a comprehensive analysis of our relationship to these methods is elaborated in the Related Work section.

Our environment is also designed to be highly configurable for a variety of Artificial Life and AI applications. DIRT supports terrain elevation, a water cycle, weather system, a day/night cycle and a seasonal cycle, as well as resource dynamics, each of which impacts the agents' actions and observations in different ways. DIRT also supports visual, olfactory and auditory sensors, which can be added or removed and independently configured. Agents in DIRT collect resources in order to survive and reproduce, and have individual traits that can affect their abilities. However, we do not enforce any particular model of heredity, and instead allow these traits to be modified arbitrarily in an effort to make the system interoperable with a range of evolutionary and learning algorithms. The Model section provides a mathematical description of the high-level interface between agents and the environment, while the Simulator section provides the details of the environmental dynamics, how they interact with each other and their configuration parameters.

Given the difficulty of performing experiments on large population sizes, we have also developed a suite of tools designed to measure and inspect simulation runs. This includes a 3D visualizer, shown in Figure 1, that allows users to inspect and track individual agents and environment properties. We have also developed a framework for running behavioral assays, as well as tools for tracking agent lineages

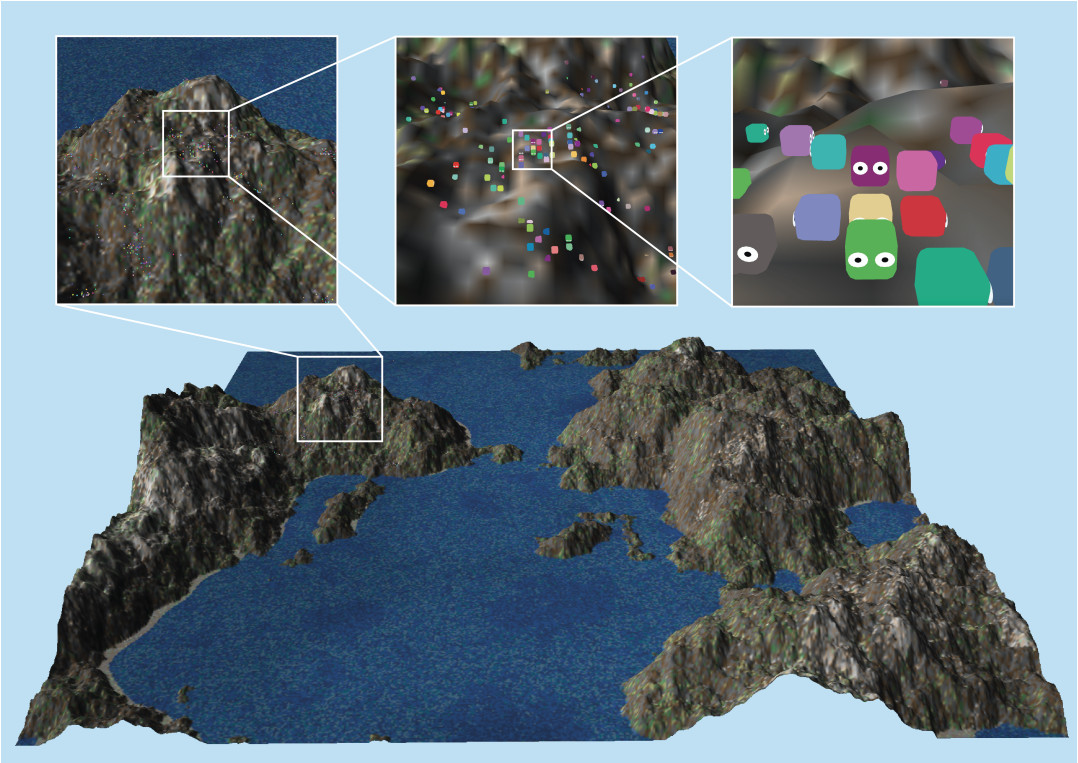

Figure 1: Simulator Overview: This map is contains 1024×1024 grid cells and 10000 individual agents, which are too small to see when fully zoomed out. The inset areas progressively zoom in on one patch of terrain to reveal detail.

and recording other customized data about the population over time. The Measurement section provides details on these tools.

Our goal in building DIRT is to provide a scalable and configurable environment for researchers studying massive populations of agents in evolutionary settings. We hope that this system provides new tools for studying the emergent behavior of intelligent agents, especially population dynamics, on a large scale.

Related Work

While an important goal of Artificial Life research is the open-ended evolution of emergent behavior, the abstractions and computational paradigms used to achieve this goal have developed significantly over several decades. In many instances these changes have been driven by the availability of new computational resources. Below we give a brief overview of this important history and discuss its relationship to DIRT, our new simulator.

Cellular Automata and Agent-Based Models

Some of the earliest artificial life systems were cellular automata developed using only pen and paper (Von Neumann et al., 1966; Gardner, 1970). These systems apply simple rules to state values stored in a discrete grid. Since this early foundational work, these systems have grown in size and complexity (Adamatzky and Martínez, 2016), yielding insight into the nature of computation (Wolfram and Gad-el Hak, 2003). In recent years, these systems have also been extended to continuous domains (Rafler, 2011; Chan, 2018; Plantec et al., 2023), and learning-based models (Mordvintsev et al., 2020; Kumar et al., 2024).

In parallel to these efforts, researchers also began developing agent-based models that simulate the interactions between discrete individuals, often embedded within a larger spatial structure. Early examples studied flocking and schooling behaviors in animals (Reynolds, 1987), and agents that adapt to their environment due to competitive pressures (Holland, 1992). Some of these simulated interactions between computer programs (Ray, 1992; Ray and Hart, 1999). As computational resources grew, some environments (Yaeger et al., 1994) began making use of basic rendering in order to incorporate visual behaviors, while others used increasingly sophisticated physical simulation to explore adaptive morphology (Sims, 1994). Using early Neural Networks to control agents rather than fixed rules also became an important approach in this era (Channon and Damper, 1998; Channon, 2001). The development of open-ended agent-based toolkits such as Avida (Ofria and Wilke, 2004), MASON (Luke et al., 2005), Netlogo

Name	GPU	Distributed	# Agents	Dynamics	Births	Deaths	Objectives	Config.
MASON	×	✓	$> 10^5$	Generic	✓	✓	Generic	++++
Griddly	✓	×	$> 10^2$	Gridworld	✓	✓	Generic	+++
Gigastep	✓	×	$> 10^4$	Particle	×	✓	Reward	++
MAgent	×	×	$> 10^{6*}$	Gridworld	×	✓	Reward	+
XLand-Minigrid	✓	×	$= 10^0$	Gridworld	×	×	Reward	+++
Jax-MARL	✓	×	$> 10^1$	Various	×	×	Reward	+
Neural MMO 2	×	×	$> 10^4$	Gridworld	×	✓	Reward	++
JaxLife	✓	×	$> 10^2$	Gridworld	✓	✓	Open	+
Amorphous Fortress	×	×	$\sim 10^1$	Gridworld	✓	✓	Open	++
DIRT (Ours)	✓	(soon!)	$> 10^5$	Gridworld	✓	✓	Open	++

Table 1: Comparison of agent-based artificial life systems and multi-agent reinforcement learning environments. **GPU** indicates that the environment code runs on accelerator hardware. **Distributed** means that a single large environment can be distributed across multiple machines or accelerators. **Agents** is an approximate order of magnitude estimate of simultaneous agents that can coexist in a single environment. The * by MAgent indicates that this is an advertised capacity which is rarely used. **Dynamics** indicates they kind of environmental dynamics supported. **Births** and **Deaths** indicates the kinds of population dynamics supported. **Objectives** indicates if the environment is a generic open-ended platform for building a variety of scenarios, reward-based or Open-ended (objective-free). Finally **Config.** indicates the level of configurability, with ++++ indicating a tool for building completely new environments with many types of dynamics, +++ indicating a tool for building completely new environments with many different objectives, but a single type of dynamics, ++ indicating a tool for building heavily configured environments or scenarios within a more specific context, and + indicating either a number of fixed scenarios with limited configuration options. Citations for these methods are MASON (Luke et al., 2005), Griddly (Bamford et al., 2020), Gigastep (Lechner et al., 2023), MAgent (Zheng et al., 2017), XLand-Minigrid (Nikulin et al., 2023), Jax-MARL (Rutherford et al., 2023), Neural MMO 2 (Suarez et al., 2023), JaxLife (Lu et al., 2024), and Amorphous Fortress (Charity et al., 2023).

(Tisue and Wilensky, 2004) and Mesa (Kazil et al., 2020) allowed greater customization for a wider range of research objectives. Along with these technical advances, researchers continued building new systems to study specific components of open-ended evolution (Spector et al., 2007; Soros and Stanley, 2014).

Our DIRT simulator sits between general-purpose agent-based modeling tools, which offer more generality but require greater authoring effort, and special-purpose environments designed to explore specific phenomena. Some recent work that is similar to this effort is Amorphous Fortress (Charity et al., 2023) which provides flexible open-ended game design and JaxLife (Lu et al., 2024) which provides a GPU-accelerated grid world environment with resources and open-ended reproduction rules.

Our goal is to create a system that can be used for a variety of large-scale artificial life applications with easy-to-use configuration parameters.

Biomechanical and Neurosensory Simulations

While many of the systems discussed above use abstract settings to study interactions between many agents, there has also been a growing push to utilize new computational tools to produce more realistic and accurate simulations of model organisms such as nematodes (Szigeti et al., 2014), fruit flies (Lobato-Rios et al., 2022; Wang-Chen et al., 2023; Vaxenburg et al., 2025), rats (Aldarondo et al., 2024) and ants (Sun et al., 2025).

Large-Scale Learning Environments

Simulating agent interactions with synthetic environments has also been an important component of reinforcement learning (RL) and multi-agent reinforcement learning (MARL) research. These settings have benefited from standardized APIs such as Gym/Gymnasium (Brockman et al., 2016; Towers et al., 2024), Gymnax (Lange, 2022) and Petting Zoo (Terry et al., 2021), which allow researchers to develop policy architectures and learning algorithms for a variety of environments. While developing a new API for population-based systems is not the primary focus of this work, we take inspiration from these frameworks and build our system in a way that is similarly agnostic to agent policies and inheritance mechanisms for greater flexibility.

In recent years, as RL/MARL experiments (Vinyals et al., 2019; Berner et al., 2019; Team et al., 2021; Zheng et al., 2017) and simulators (Guss et al., 2019; Bamford et al., 2020; Yu et al., 2024; Suarez et al., 2023) have dramatically increased in scale and complexity, researchers have begun to develop GPU-accelerated environments (Koyamada et al., 2023; Rutherford et al., 2023; Matthews et al., 2024; Lechner et al., 2023; Nikulin et al., 2023) in order to reduce the computational bottleneck introduced by interaction with the environment. Much of this work has relied on the JAX ecosystem (Bradbury et al., 2018), which provides tools for

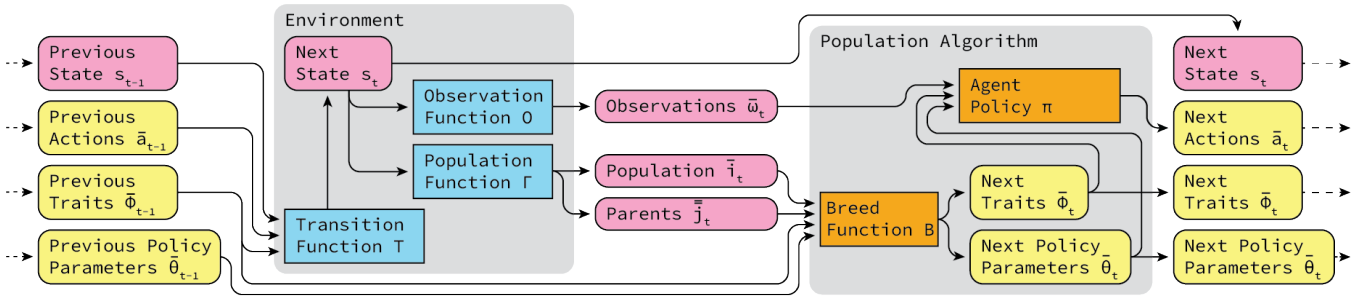

Figure 2: **Illustration of** information flow between the environment and the learning algorithm. Pink boxes represent environment data and blue boxes represent environment functions. Yellow boxes represent population algorithm data and orange boxes represent population algorithm functions.

parallelized tensor processing.

Model

Inspired by mathematical models such as Markov decision processes (MDPs), stochastic games (SGs) and their partially observable counterparts (POMDPs and POSGs), we build a model that supports open-ended populations of agents with underlying Markovian state dynamics. This allows us to cleanly separate the agent's decision-making from the environment and present a simple interface that is compatible with a variety of agent policies and population algorithms. This interface is depicted in Figure 2. This model is a general framework meant to describe any open-ended interactive environment with a dynamic population. In this section we describe the components of that model and in the Simulator Section below we provide details on the specific instantiation used in DIRT.

Our mathematical model combines the multi-agent Markovian dynamics of a POSG (Hansen et al., 2004) but adds heritable agent traits and replaces the reward function and discount factor with a model of the population dynamics. This yields a structure that we refer to as a partially observable ecological game (POEG):

$$(I, \Gamma, S, \rho_0, A, \Phi, T, \Omega, O) \tag{1}$$

Similar to a POSG, I is a set of agent identifiers, S is a set of states and ρ_0 is an initial distribution over states. Unlike a POSG, which typically contains a fixed number of agents over time, a POEG includes the function Γ, which describes the dynamic number of agents in the underlying state.

$$\Gamma(s_t, a_t) \to \bar{i}_t, \bar{\bar{j}}_t$$

Here we use the notation $\bar{i}_t = \{i_1 \dots i_n\} \subseteq I$ to represent the population of n agents that are alive at time t, and $\bar{\bar{j}}_t = \{\bar{j}_0 \dots \bar{j}_n\}$ to represent a set of parent vectors $\bar{j}_{t,k} \in \mathbb{R}^{|I|}_{>=0}$ for each agent $i_{t,k}$. Representing the parents as a vector of nonnegative real numbers allows each individual to have an arbitrary number of parents, and for those parents to have unequal contributions to their offspring's genes. The only constraint we impose is that for an individual $i_{t,k}$ born at time t (implying $i_{t,k} \in \bar{i}_t$, but $i_{t,k} \notin \bar{i}_{t-1}$), its parents must have been alive at some point in the previous time steps $j_{t,k,l} > 0 \implies l \in \cup_{t'=0}^{t-1} \bar{i}_{t'}$.

Note that the values $\bar{i}_t$ and $\bar{\bar{j}}_t$ are not state variables, but should instead be thought of as information communicated to an external user about the changing population so that policies for newly created agents can be instantiated, and old ones for the dead can be discarded.

Agents influence the dynamics of the environment by taking actions $\bar{a}_t = \{a_{t,0} \dots a_{t,n}\} \in A^n$, where A is the set of available actions for a single agent. The effect of these actions and the environmental dynamics are also influenced by traits $\bar{\phi}_t = \{\phi_0 \dots \phi_n\} \in \Phi^n$ which are meant to represent the potential differences between individual agents that are not captured by their actions. The model described here does not require these traits to be constant for each agent over time, which allows for both developmental and heritable traits. The state dynamics are described by a transition function $T(s_t, \bar{a}_t, \bar{\phi}_t) \to p(s_{t+1}|s_t, \bar{a}_t, \bar{\phi}_t)$ that maps states, actions and traits to a distribution over future states.

Finally, Ω is the set of all possible observations an agent could receive, and a vector of observations for a population would be $\bar{\omega} = \{\omega_0 \dots \omega_n\} \in \Omega^n$. O maps a state to a distribution over observations for each agent $O(s_t) \to p(\bar{\omega}_t|s_t)$.

In order to interface with this model, a population algorithm (the equivalent of a learning algorithm in an objective-based reinforcement learning setting) only needs to provide a number of actions and traits that align with the population size at each time step. Realistically, a population algorithm should maintain a set of policy parameters $\bar{\theta} = \{\theta_0 \dots \theta_n\} \in \Theta^n$ for each agent that is updated as new agents are born or old agents die. In Figure 2 we refer to a breed function B that takes the previous traits $\bar{\phi}_{t-1}$ and policy parameters $\bar{\theta}_{t-1}$, as well as the population information $\bar{i}_t$ and $\bar{\bar{j}}_t$, and produces a new set of traits $\bar{\phi}_t$ and policy parameters $\bar{\theta}_t$. These traits and parameters can then be used

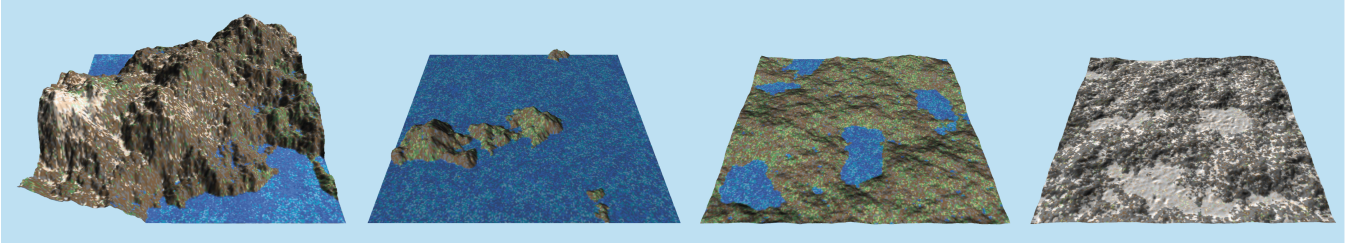

Figure 3: A mountain range, island chain, wetlands and a frozen tundra, created using different configuration parameters.

in a policy π along with the observations $\bar{\omega}_t$ to compute the next actions $\bar{a}_t$.

Importantly, the policy π, its parameterization Θ and the breed function B are all part of the population algorithm, and thus not specified by the environment. This separation allows researchers to implement and compare different policy classes and inheritance models of the environment's traits, while keeping the environmental dynamics fixed.

Simulator

In the DIRT simulator, the environment state space S from Equation 1 contains data with three high-level shapes. The first is two-dimensional map information $s^m_{t,x,y}$ containing spatial information about the distribution of resources and terrain features in the world, indexed using time t and spatial dimensions x and y. The second is agent-specific information $s^a_{t,i}$ about the position and resources allocated to each agent, indexed by time t and agent index i. Finally, global information s^g_t, such as the current time of day, is not specific to any grid cell or agent and is only indexed by time t.

$$s_t = \{s^m_t, s^a_t, s^g_t\}$$

The subsections below describe the components of the environment and their dynamics.

Terrain

Although our system is a two-dimensional grid world, we add terrain height to each cell in order to provide greater environmental complexity. This terrain affects the weather conditions and the amount of incoming light, described in Subsection Days, Seasons and Climate below, as well as the effort agents must take to traverse the landscape, described in Subsection Agent Dynamics.

The terrain consists of a height map representing the underlying rock $r_{t,x,y}$. This is initialized using Fractal noise, a sum of multiple layers (octaves) of Perlin Noise (Perlin, 1985) with increasing frequency and decreasing magnitude:

$$r_{0,x,y} = h \sum_{i=0}^{c-1} w^i \eta\left(u\lambda^i\left(x, y\right)\right)$$

where h is an overall height scale, c is the number of octaves, and η is 2D Perlin noise. $w \in (0, 1)$ describes the persistence, or how quickly the noise magnitude is reduced between each octave, $\lambda \in (1, \infty)$ is the lacunarity, which is the increase in frequency for each octave, and u is the unit scale controlling the frequency of the first octave.

Water

Water $w_{t,x,y}$ in DIRT can evaporate, precipitate, and flow downhill on the map based on the terrain. Water is initialized using a desired sea level z_0 relative to the terrain.

$$w_{0,x,y} = \max(z_0 - r_{0,x,y}, 0)$$

We refer to the water plus rock as the "altitude" $a_{t,x,y} = r_{t,x,y} + w_{t,x,y}$ which is used for several weather-related calculations discussed later.

We implement the hydrodynamics using a discrete water flow simulation. Water flows between adjacent cells in direction $d \in \{\leftarrow, \uparrow, \rightarrow, \downarrow\}$ based on local differences in altitude scaled by a flow rate $\alpha_w \in [0, 1]$.

$$\Delta w^d_{t,x,y} = \text{clip}\left(-\alpha_w \Delta a^d_{t,x,y}, 0, \frac{w_{t,x,y}}{4}\right)$$

The term above describes the water flowing out from one location to its neighbor, where $\Delta a^{(d)}_{t,x,y}$ represents the difference of the altitude between the two locations. The combined effect is

$$w_{t+1,x,y} = \Delta m_{t,x,y} + \sum_d \Delta w^{-d}_{t,(x,y)+d} - \Delta w^d_{t,x,y} \quad (2)$$

where $\Delta m_{x,y,t}$ is an evaporation term described in the Days, Seasons and Climate subsection below. In areas where the temperature drops below 0, water freezes and α_w is reduced to almost 0.

These dynamic water effects have several configuration parameters to control the evaporation, precipitation and flow rates, and can also be disabled for performance considerations.

Days, Seasons and Climate

The climate simulator in DIRT manages light, temperature, and air moisture values at each grid cell, as well as a global wind direction. The climate system also propagates scents and audio for the agents sensors.

Light is controlled by a global day/night and seasonal cycle. The light entering each cell is computed by comparing the global light direction, which is dependent on the current time of day and year, to a cell-specific altitude normal. This value is multiplied by local air moisture, representing shade caused by clouds.

The wind direction ξ_t is a global variable governed by an OU process with configuration parameters θ_ξ and σ_ξ:

$$\xi_t = \theta_\xi \xi_{t-1} + \Delta\xi_t,\ \Delta\xi_t \sim \mathcal{N}_{\sigma_\xi}$$

Many quantities in the climate system are modeled using simplified gas propagation. This model stores gas values $g_{t,x,y}$ at each grid cell, offsets them according to the wind such that $x', y' = (x, y) - \xi$, and diffuses them using a convolution kernel K_σ. It also includes an optional reversion term g_0 and reversion rate α_g to return the quantity to some mean over time, and a term $\Delta g_{t,x,y}$ which describes an external quantity added to the system.

$$g_{t,x,y} = K_\sigma \odot (\Delta g_{t,x,y} + \alpha_g g_0 + (1 - \alpha_g)\, g_{t-1,x',y'}) \quad (3)$$

The model also supports storing quantities at lower resolution so that they can be propagated faster and at lower cost, at the expense of coarser granularity.

The temperature $\tau_{t,x,y}$ is also simulated using this model, where Δg is heat gain from the amount of light entering a grid cell and g_0 is an altitude-specific baseline night-time temperature. The temperature reversion rate α_g depends on the standing water in each grid cell, such that areas with water change temperature more slowly.

Moisture $m_{t,x,y}$ is accumulated in the air due to evaporation $\Delta m_{t,x,y}$, which is a function of the local temperature and water.

$$\Delta m^e_{t,x,y} = \min\left(\tau_{t,x,y}\alpha_m, w_{t,x,y}\right)$$

The evaporation is subtracted from the water $w_{x,y,t}$ (see Equation 2). Once the air has accumulated enough moisture, it begins to rain, and continues to do so until the moisture level has been depleted.

$$\Delta m^r_{x,y,t} = \min(\delta^r, m_{x,y,t})$$

Odors $o_{t,x,y}$ are represented as a multi-channel signal at each grid location. The number of channels is configurable. New odors $\Delta g_{t,x,y}$ are created by both agents and resources, and then propagated using the gas model with a slow reversion to zero ($g_0 = 0$).

Audio $a_{t,x,y}$ is similarly represented as a multi-channel signal with a configurable number of frequencies generated by agents. In order to capture the instantaneous quality of audio, it is simulated on a much coarser grid defaulting to $1/64$ of the full gridworld resolution. Audio also does not persist, meaning $g_0 = 0$ and $\alpha_g = 1$.

Resource Dynamics

There are four resources in DIRT: light, energy, biomass, and water. The propagation of light and water have already been discussed in the Terrain and Days, Seasons and Climate subsections. Biomass is the physical quantity necessary to build agents, while energy and water are necessary for metabolism and movement. Like water, the total quantity of biomass is conserved in the environment. When a new agent is born, its parent must give it a certain fixed quantity of biomass. An agent can increase its biomass by finding and eating more of it in the environment.

Energy, on the other hand, is not conserved. It can be produced internally by agents with a photosynthetic trait in the presence of light. It can also optionally be produced by standing biomass in the environment based on a biomass photosynthesis configuration parameter. The more activity agents perform, the more energy and water they use. Used energy is simply destroyed, while used water is returned to the air as evaporated moisture.

Agent Dynamics

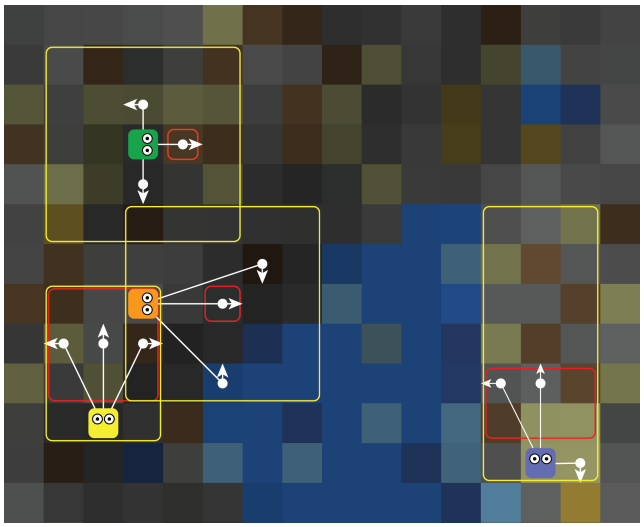

Figure 4: An example of several agent observation and action spaces. In this example, each agent has three motion primitives, shown here as white arrows. Note that these primitives perform different motions for each agent, as they each have different traits. Each agent has a single attack primitive in this example represented by the red box. The agent's visual range is represented by the yellow box. The pixelated background represents the color map that agents see, where blue pixels are water, grey is rock and shades of yellow and brown represent different distributions of the biomass and energy resources.

DIRT is an open-ended, objective-free environment. Instead of an external reward signal, the environment's dynamics provide an implicit fitness function by determining which agents survive and reproduce. Agents' born with a certain amount of starting resources and full health. Agents'

health will be reduced if they are attacked by other agents or if they do not collect enough resources to accommodate their metabolic functions. When an agent dies, the resources it was carrying inside its body are returned to the grid cell in which they died.

Agent actions in DIRT are discrete and belong to one of six high-level action categories: **Move**, **Attack**, **Eat**, **Call**, **Mark Scent** and **Reproduce**. However, within each of these categories, an agent may have multiple distinct choices, such as which direction to move or attack.

Move is controlled with a fixed number of motion primitives. Each of these specifies a relative offset to the agent's current position and orientation. The number of these movement primitives is configurable, and the exact offset specified by each one depends on the agent's traits, which allows for agents with different speeds. The white arrows in Figure 4 give examples of this for agents with three movement primitives each. These offsets are real-valued, but discretized using stochastic rounding. The energy required to perform a movement action depends on its distance. Only one agent can be in a grid cell at once. If an agent attempts to move into a currently occupied cell, or if two agents attempt to move to the same cell, the movement fails and they are returned to their original position.

Attack is similarly structured with a set of primitives. Each of these specifies an offset, shape and a strength which are again controlled by agent-specific traits. Attacks damage the health of other agents that lie within the attack radius based on the strength of the attack and the toughness (another trait) of the opposing agent. The red boxes in Figure 4 give examples of the attack areas of several agents.

Call and **Mark Scent** actions also consist of a set of primitives, each of which adds a different value to the audio and odor features in the agent's current grid cell. As before, the values corresponding to these actions are controlled by agent-specific traits.

Eat is a single action that consumes whatever resources are available in the current grid cell. Each agent has a stomach size that can accommodate a fixed amount of each resource.

Reproduce is also a single action that creates new offspring using single-parent reproduction. This action only works if the agent has accumulated enough resources to donate to its new offspring.

Agents in DIRT have access to six sensors: **visual**, **audio**, **odor**, **thermal**, **wind**, and **internal** that they can use to make decisions.

Visual sensing is approximated by generating a color map of the environment and slicing out a visible window in front of each agent. This is a common model used in other grid-world settings (Chevalier-Boisvert et al., 2023; Matthews et al., 2024; Lu et al., 2024). The color map is generated by overlaying the colors of the rock, water, and resources and then multiplying by the light intensity l_t. In addition to this color map, we also give the agents access to the local elevation change in this window, which is normalized relative to the agent's current elevation. The visual sensing range can be configured on a per-agent basis using a set of traits that describe the position, length and width of the visual window. A light and altitude sensitivity trait adds uniform noise to these values to simulate agents with noisy sensors. The yellow boxes in Figure 4 give examples of agents' viewing windows.

Odor, **Audio**, and **Thermal** sensing provides the agent with the odor, audio, and temperature values at their present location. The **Wind** sensor tells the agent which direction the wind is blowing, and the **Internal** sensor tells the agent the contents of its stomach, and its health level. As with the visual sensor, each agent has a sensitivity trait that controls the amount of noise added to these values.

Traits

As mentioned previously, agents have a number of traits that can control their individual behavior. Some of these traits can be more beneficial than others. For example, an agent with stronger attacks, or a larger attack box might be strictly more powerful than weaker agents. In order to balance these, we attempt to provide a comprehensive set of configuration parameters allowing these more beneficial traits to be costly in some way. For example, the amount of energy required to move an agent should depend on the distance of the chosen movement primitive. In this way faster agents must eat more in order to support their more energetic lifestyle. Similarly, traits can have biomass costs as well, for example greater speed may require longer legs which require more resources to build.

Measurement

Across many artificial life environments, it remains notoriously difficult to explore both the environmental state and individual agent parameters at specific time steps. Our environment is no different. Given our large map sizes and populations, logging the entire state information from each time step would generate dozens of gigabytes of data in minutes. Furthermore, storing all this data to disk can substantially slow the simulation. To prioritize depth of analysis within our system, we have carefully designed reporting and measurement tools as built-in features that occur between chunks of just-in-time compiled code. Our high level strategy is to store small quantities of customizable **report** data per time step, and then periodically store the entire state as a **checkpoint** which contains enough information to restart the simulation from a particular point in time. These tools enable users to design detailed and rigorous experiments to test general hypotheses about artificial life systems.

Reporting

Our system includes a robust and customizable reporting structure to analyze environmental dynamics across many levels of detail. Frequency of reporting relative to the number of time steps can be adjusted depending on memory limitations and simulation length. Reporting can include three primary types of information:

1. Global information, such as the presence of food at a grid cell or the total number of players at a given time point.
2. Agent-specific information, such as traits.
3. Derived values, such as a family tree of individuals across the simulation.

Global Environmental and population-level values, such as the locations of food on the grid, the locations of players, or the altitude at certain points on the grid, can all be directly included in reports. This allows for various statistical analyses that mirror those performed in experimental biological settings; for example, a simulation could be assessed for the correlation between total amount of food and total number of players per time step.

Agent-Specific Individual agents can be extracted from the environment at any step, and analyzed either in isolation or in the context of a new environment. Easily accessible state variables and reproducible dynamics are the two primary engineering features that make this possible.

Derived The reporting system is not constrained to existing environmental state parameters; sets of parameters can be interpreted as custom function arguments, and the function output can be directly included in a report file. For example, a tree-like data structure mapping the parent-child relationships of all agents that existed at some point during the simulation (i.e. family tree) can be constructed to analyze distinct lineages and evolutionary bottlenecks.

Visualization

Due to the potential complexity of the systems produced by DIRT, visual analysis can sometimes provide nuanced behavioral insight than could be missed by statistical analysis of report data. The interactive 3D visualizer built into DIRT allows for step-by-step replays of past simulations, starting at any specified point in a simulation. Visual information such as environmental altitude, water, rain, and the location, direction, and energy of agents are all included in the visualizer as shown in Figure 5. It should be noted that the visualizer requires substantial report data in order to function. Consequently, we also support the workflow of selective visualization, in which a large simulation is run with periodic checkpoints, but without the reporting necessary for visualization. Then for any desired range, the checkpoints can be used to rerun small sections of the simulation that need to be visualized with the reporting data necessary.

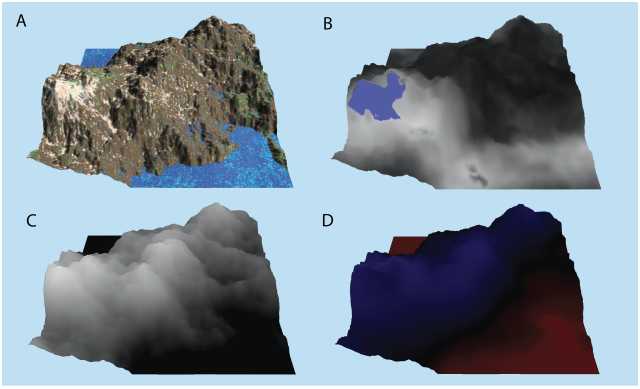

Figure 5: Visualization of environmental phenomena in the mountain example. A: the fully rendered environment. B: cloud cover with raining areas shown in blue. C: Altitude. D: Temperature, with blue as negative (freezing) and red positive. Note that the reason the mountain peaks in A are white is the presence of frozen water due to this temperature.

Behavioral Assays

We have also developed tools for performing behavioral assays, where individuals from a population can be instantiated into a smaller controlled environment to measure their performance. For example to test how quickly an agent will find and eat food. We again leverage the parallel processing power of the GPU to run these in batch across many agents simultaneously.

Conclusion

Our goal in releasing DIRT is to provide researchers with new tools to accelerate and scale artificial life experiments. DIRT is designed to allow for large populations of mobile agents controlled by deep neural networks. To this end, we prioritize environment dynamics that provide a rich cognitive challenge, yet are cheap and parallelizable on modern hardware accelerators. We have also built in a comprehensive set of configuration parameters to customize the environment to a variety of applications. We have also developed a rich set of inspection and measurement tools that enable fine-grained tracking and experimentation on individuals, lineages and subpopulations. We hope that DIRT will be a useful tool for ALIFE and AI researchers studying the population dynamics of intelligent agents.

References

Abar, S., Theodoropoulos, G. K., Lemarinier, P., and O'Hare, G. M. (2017). Agent based modelling and simulation tools: A review of the state-of-art software. *Computer Science Review*, 24:13–33.

Adamatzky, A. and Martínez, G. J. (2016). *Designing beauty: the art of cellular automata*, volume 20. Springer.

Aldarondo, D., Merel, J., Marshall, J. D., Hasenclever, L., Klibaite, U., Gellis, A., Tassa, Y., Wayne, G., Botvinick, M., and Ölveczky, B. P. (2024). A virtual rodent predicts the structure of neural activity across behaviours. *Nature*, 632(8025):594–602.

Bamford, C., Huang, S., and Lucas, S. (2020). Griddly: A platform for ai research in games. *arXiv preprint arXiv:2011.06363*.

Berner, C., Brockman, G., Chan, B., Cheung, V., Debiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., et al. (2019). Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. (2018). JAX: composable transformations of Python+NumPy programs.

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

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Chan, B. W.-C. (2018). Lenia-biology of artificial life. *arXiv preprint arXiv:1812.05433*.

Channon, A. (2001). Passing the alife test: Activity statistics classify evolution in geb as unbounded. In *Advances in Artificial Life: 6th European Conference, ECAL 2001 Prague, Czech Republic, September 10–14, 2001 Proceedings 6*, pages 417–426. Springer.

Channon, A. and Damper, R. (1998). Perpetuating evolutionary emergence.

Charity, M., Rajesh, D., Earle, S., and Togelius, J. (2023). Amorphous fortress: Observing emergent behavior in multi-agent fsms. *arXiv preprint arXiv:2306.13169*.

Chevalier-Boisvert, M., Dai, B., Towers, M., de Lazcano, R., Willems, L., Lahlou, S., Pal, S., Castro, P. S., and Terry, J. (2023). Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. *CoRR*, abs/2306.13831.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., et al. (2023). Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Gardner, M. (1970). Mathematical games. *Scientific american*, 222(6):132–140.

Guss, W. H., Houghton, B., Topin, N., Wang, P., Codel, C., Veloso, M., and Salakhutdinov, R. (2019). Minerl: A large-scale dataset of minecraft demonstrations. *arXiv preprint arXiv:1907.13440*.

Hansen, E. A., Bernstein, D. S., and Zilberstein, S. (2004). Dynamic programming for partially observable stochastic games. In *AAAI*, volume 4, pages 709–715.

Holland, J. H. (1992). *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press.

Jouppi, N. P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., Bates, S., Bhatia, S., Boden, N., Borchers, A., et al. (2017). In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th annual international symposium on computer architecture*, pages 1–12.

Kazil, J., Masad, D., and Crooks, A. (2020). Utilizing python for agent-based modeling: The mesa framework. In *Social, Cultural, and Behavioral Modeling: 13th International Conference, SBP-BRiMS 2020, Washington, DC, USA, October 18–21, 2020, Proceedings 13*, pages 308–317. Springer.

Koyamada, S., Okano, S., Nishimori, S., Murata, Y., Habara, K., Kita, H., and Ishii, S. (2023). Pgx: Hardware-accelerated parallel game simulators for reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pages 45716–45743.

Kumar, A., Lu, C., Kirsch, L., Tang, Y., Stanley, K. O., Isola, P., and Ha, D. (2024). Automating the search for artificial life with foundation models. *arXiv preprint arXiv:2412.17799*.

Lange, R. T. (2022). gymnax: A JAX-based reinforcement learning environment library.

Lechner, M., Yin, L., Seyde, T., Wang, T.-H., Xiao, W., Hasani, R., Rountree, J., and Rus, D. (2023). Gigastep - one billion steps per second multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*.

Lobato-Rios, V., Ramalingasetty, S. T., Özdil, P. G., Arreguit, J., Ijspeert, A. J., and Ramdya, P. (2022).

Neuromechfly, a neuromechanical model of adult drosophila melanogaster. *Nature Methods*, 19(5):620–627.

Lu, C., Beukman, M., Matthews, M., and Foerster, J. (2024). Jaxlife: An open-ended agentic simulator. In *ALIFE 2024: Proceedings of the 2024 Artificial Life Conference*. MIT Press.

Luke, S., Cioffi-Revilla, C., Panait, L., Sullivan, K., and Balan, G. (2005). Mason: A multiagent simulation environment. *Simulation*, 81(7):517–527.

Macal, C. M. (2016). Everything you need to know about agent-based modelling and simulation. *Journal of Simulation*, 10(2):144–156.

MacArthur, R. H. and Wilson, E. O. (2001). *The theory of island biogeography*, volume 1. Princeton university press.

Matthews, M., Beukman, M., Ellis, B., Samvelyan, M., Jackson, M., Coward, S., and Foerster, J. (2024). Craftax: A lightning-fast benchmark for open-ended reinforcement learning. In *International Conference on Machine Learning (ICML)*.

Mordvintsev, A., Randazzo, E., Niklasson, E., and Levin, M. (2020). Growing neural cellular automata. *Distill*, 5(2):e23.

Nikulin, A., Kurenkov, V., Zisman, I., Agarkov, A., Sinii, V., and Kolesnikov, S. (2023). Xland-minigrid: Scalable meta-reinforcement learning environments in jax. *arXiv preprint arXiv:2312.12044*.

Ofria, C. and Wilke, C. O. (2004). Avida: A software platform for research in computational evolutionary biology. *Artificial life*, 10(2):191–229.

Paszke, A. (2019). Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*.

Perlin, K. (1985). An image synthesizer. *ACM Siggraph Computer Graphics*, 19(3):287–296.

Plantec, E., Hamon, G., Etcheverry, M., Oudeyer, P.-Y., Moulin-Frier, C., and Chan, B. W.-C. (2023). Flowlenia: Towards open-ended evolution in cellular automata through mass conservation and parameter localization. In *Artificial Life Conference Proceedings 35*, volume 2023, page 131. MIT Press One Rogers Street, Cambridge, MA 02142-1209, USA journals-info ....

Rafler, S. (2011). Generalization of conway's" game of life" to a continuous domain-smoothlife. *arXiv preprint arXiv:1111.1567*.

Railsback, S. F., Lytinen, S. L., and Jackson, S. K. (2006). Agent-based simulation platforms: Review and development recommendations. *Simulation*, 82(9):609–623.

Rasley, J., Rajbhandari, S., Ruwase, O., and He, Y. (2020). Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 3505–3506.

Ray, T. S. (1992). Evolution, ecology and optimization of digital organisms. *Santa Fe*.

Ray, T. S. and Hart, J. (1999). Evolution of differentiated multi-threaded digital organisms. In *Proceedings 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human and Environment Friendly Robots with High Intelligence and Emotional Quotients (Cat. No. 99CH36289)*, volume 1, pages 1–10. IEEE.

Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 25–34.

Rosenzweig, M. L. (1995). Species diversity in space and time. *(No Title)*.

Rutherford, A., Ellis, B., Gallici, M., Cook, J., Lupu, A., Ingvarsson, G., Willi, T., Khan, A., de Witt, C. S., Souly, A., et al. (2023). Jaxmarl: Multi-agent rl environments in jax. *arXiv preprint arXiv:2311.10090*.

Shoeybi, M., Patwary, M., Puri, R., LeGresley, P., Casper, J., and Catanzaro, B. (2019). Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*.

Sims, K. (1994). Evolving 3d morphology and behavior by competition. *Artificial life*, 1(4):353–372.

Soros, L. and Stanley, K. (2014). Identifying necessary conditions for open-ended evolution through the artificial life world of chromaria. In *Artificial Life Conference Proceedings*, pages 793–800. MIT Press One Rogers Street, Cambridge, MA 02142-1209, USA journals-info ....

Spector, L., Klein, J., and Feinstein, M. (2007). Division blocks and the open-ended evolution of development, form, and behavior. In *Proceedings of the 9th annual conference on genetic and evolutionary computation*, pages 316–323.

Suarez, J., Bloomin, D., Choe, K. W., Li, H. X., Sullivan, R., Kanna, N., Scott, D., Shuman, R., Bradley, H., Castricato, L., et al. (2023). Neural mmo 2.0: A massively

multi-task addition to massively multi-agent learning. *Advances in Neural Information Processing Systems*, 36:50094–50104.

Sun, X., Mangan, M., Peng, J., and Yue, S. (2025). I2bot: an open-source tool for multi-modal and embodied simulation of insect navigation. *Journal of the Royal Society Interface*, 22(222):20240586.

Szigeti, B., Gleeson, P., Vella, M., Khayrulin, S., Palyanov, A., Hokanson, J., Currie, M., Cantarelli, M., Idili, G., and Larson, S. (2014). Openworm: an open-science approach to modeling caenorhabditis elegans. *Frontiers in computational neuroscience*, 8:137.

Team, O. E. L., Stooke, A., Mahajan, A., Barros, C., Deck, C., Bauer, J., Sygnowski, J., Trebacz, M., Jaderberg, M., Mathieu, M., et al. (2021). Open-ended learning leads to generally capable agents. *arXiv preprint arXiv:2107.12808*.

Terry, J., Black, B., Grammel, N., Jayakumar, M., Hari, A., Sullivan, R., Santos, L. S., Dieffendahl, C., Horsch, C., Perez-Vicente, R., et al. (2021). Pettingzoo: Gym for multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 34:15032–15043.

Tisue, S. and Wilensky, U. (2004). Netlogo: A simple environment for modeling complexity. In *International conference on complex systems*, volume 21, pages 16–21. Citeseer.

Towers, M., Kwiatkowski, A., Terry, J., Balis, J. U., De Cola, G., Deleu, T., Goulão, M., Kallinteris, A., Krimmel, M., KG, A., et al. (2024). Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*.

Vaxenburg, R., Siwanowicz, I., Merel, J., Robie, A. A., Morrow, C., Novati, G., Stefanidi, Z., Both, G.-J., Card, G. M., Reiser, M. B., et al. (2025). Whole-body physics simulation of fruit fly locomotion. *Nature*, pages 1–3.

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., et al. (2019). Grandmaster level in starcraft ii using multi-agent reinforcement learning. *nature*, 575(7782):350–354.

Von Neumann, J., Burks, A. W., et al. (1966). Theory of self-reproducing automata. *IEEE Transactions on Neural Networks*, 5(1):3–14.

Wang-Chen, S., Stimpfling, V. A., Lam, T. K. C., Özdil, P. G., Genoud, L., Hurtak, F., and Ramdya, P. (2023). Neuromechfly v2, simulating embodied sensorimotor control in adult drosophila. *bioRxiv*, pages 2023–09.

Wolfram, S. and Gad-el Hak, M. (2003). A new kind of science. *Appl. Mech. Rev.*, 56(2):B18–B19.

Yaeger, L. et al. (1994). Computational genetics, physiology, metabolism, neural systems, learning, vision, and behavior or poly world: Life in a new context. In *SANTA FE INSTITUTE STUDIES IN THE SCIENCES OF COMPLEXITY-PROCEEDINGS VOLUME-*, volume 17, pages 263–263. ADDISON-WESLEY PUBLISHING CO.

Yu, X., Fu, J., Deng, R., and Han, W. (2024). Mineland: Simulating large-scale multi-agent interactions with limited multimodal senses and physical needs. *arXiv preprint arXiv:2403.19267*.

Zheng, L., Yang, J., Cai, H., Zhang, W., Wang, J., and Yu, Y. (2017). Magent: A many-agent reinforcement learning platform for artificial collective intelligence. *arXiv preprint arXiv:1712.00600*.