DIRT: The Distributed Intelligent Replicator Toolkit

Chengrui Wang Harvard University
Chase Van Amburg Harvard University
Chloe Huangyuan Su Harvard University
Joseph Bejjani Harvard University
Sarah M. Pratt University of Washington
Yasin Mazloumi Harvard University
Naeem Khoshnevis Harvard University
Sham Kakade Harvard University
Kiante Brantley Harvard University
Aaron Walsman* Harvard University

Abstract

We introduce **DIRT**, the **D**istributed **I**ntelligent **R**eplicator **T**oolkit, a GPU-accelerated simulation platform for studying large-scale multi-agent populations in simulated ecosystems. DIRT is designed to explore the ways that intelligence in artificial agents influences the emergent population dynamics of complex environments at very large scales. With this in mind, it uses fast and highly configurable grid world dynamics that include multiple sensor types, complex terrain, water features, and a climate system. Agents in DIRT must consume resources in order to survive and reproduce, with each agent's abilities governed by a set of heritable traits. DIRT specifies a simple interaction model that is designed to be agnostic to the structure of an agent's policy class and update rules, making it interoperable with a variety of evolutionary and reinforcement learning algorithms. Built on JAX, DIRT can support populations of more than 10,000 agents on a single GPU. To support analysis, DIRT includes integrated measurement tools and an interactive 3D viewer for fine-grained agent inspection and tracking.

1 Introduction

Natural selection has produced all forms of organic intelligence on Earth. In natural settings, the mind and morphology of individual organisms are not trained with an explicit objective but instead are the product of millions of years of competition for reproduction and survival in complex and dynamic ecologies. In AI, there is a rich history exploring these mechanisms (Channon and Damper, 1998; Standish, 2003; Lehman and Stanley, 2011), often under the names open-ended, objective-free or implicit-objective settings. However, the difficulty of simulating large populations in complex environments, especially when coupled with computationally expensive deep neural networks, has made these settings challenging to explore at scale. Although some contemporary examples exist (Lu et al., 2024), we believe that this area is currently under-explored and has the potential to better explain the natural emergence of intelligence in response to competitive dynamics.

With this motivation, we have built a new highly customizable environment to study large populations of mobile intelligent agents in a simulated natural setting. Following recent trends in reinforcement learning and agent development, DIRT computes environment dynamics directly on the GPU, enabling massive parallelization and reducing communication overhead with neural-network policies. This design dramatically speeds up the training and evolution loop, supporting fast iteration with tens of thousands of agents.

*Correspondence to: vincent_wang@college.harvard.edu / aaronwalsman@fas.harvard.edu

Preprint.

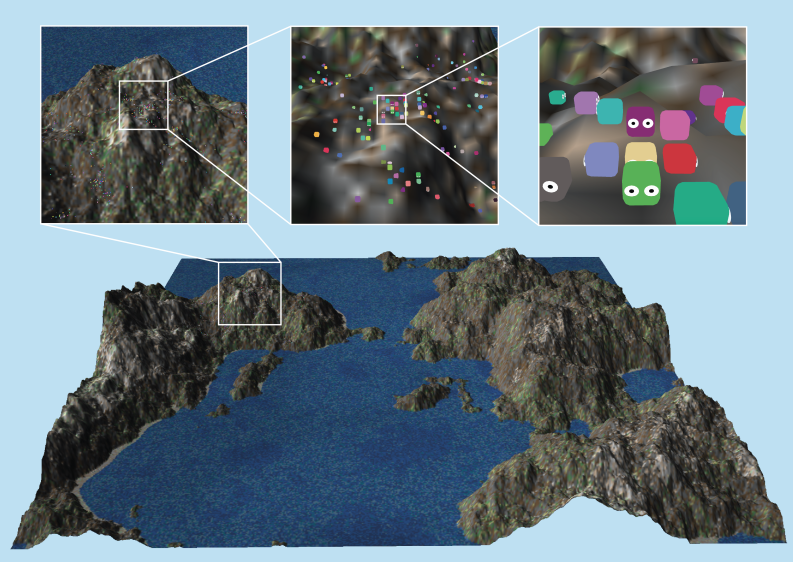

Figure 1: Simulator overview: This map contains 1024×1024 grid cells and 10,000 individual agents, which are too small to see when fully zoomed out. The inset areas progressively zoom in on one patch of terrain to reveal detail.

DIRT adopts a grid-world abstraction: discrete 2D landscapes with configurable terrain, water cycles, climate dynamics, day/night and seasonal cycles, as well as resource dynamics. This level of abstraction strikes a balance: it is cognitively challenging for agents, yet compatible with tensor-based ML workflows that leverage JAX and modern GPU/TPU pipelines.

Agents in DIRT collect resources to survive and reproduce, with abilities governed by configurable traits. Crucially, DIRT does not enforce a fixed genetic or learning model; traits can be arbitrarily modified, allowing interoperability with a wide range of evolutionary strategies and reinforcement 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 analyzing the results of experiments on large population sizes, we have also developed a suite of tools designed to measure and inspect simulation runs. This consists of a suite of measurement tools, including customizable reporting mechanisms and a 3D visualizer (Figure 1) for inspecting individual agents and environmental properties.

Our goal is to provide a scalable and configurable environment for studying massive agent populations under diverse ecological and cognitive pressures. By enabling experiments at previously infeasible scales, DIRT aims to advance research on population dynamics, coordination, and adaptation in large-scale multi-agent systems.

DIRT is provided open-source and is under active development for ongoing research projects. We are presently releasing it in working beta form in the hopes of increasing interest in population-based simulation as a means to study and develop artificial intelligence. DIRT is available at `https://github.com/aaronwalsman/dirt`.

2 Related Work

An important goal of intelligent agent research is the open-ended evolution of emergent behavior. The abstractions and computational paradigms for pursuing this goal have evolved greatly over the decades, often driven by advances in computational resources. Below we briefly overview this history and discuss its relationship to DIRT.

[Vincent: How are we going to formulate the distributed features stuff? Like in the table are we stating coming soon or something else?]

Table 1: Comparison of multi-agent reinforcement learning environments

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

* Advertised maximum capacity; rarely used in practice.

† **Births** and **Deaths** indicates the kinds of population dynamics supported.

‡ **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.

Cellular Automata, Agent-Based Models and Artificial Life. Early artificial life systems, such as Von Neumann's automata and Conway's Game of Life, used simple local rules on discrete grids to study computation and emergence (Von Neumann et al., 1966; Gardner, 1970). These ideas have since expanded to larger, continuous, and even differentiable settings (Rafler, 2011; Chan, 2018; Plantec et al., 2023; Mordvintsev et al., 2020; Kumar et al., 2024). In parallel, agent-based models simulate interactions among individuals, often embedded in spatial structures, including flocking and schooling behaviors in animals (Reynolds, 1987) and resource-driven competition and adaptation (Holland, 1992; Ray, 1992). Toolkits such as Avida (Ofria and Wilke, 2004), MASON (Luke et al., 2005), FLAME-GPU(Richmond et al., 2024), NetLogo (Tisue and Wilensky, 2004), and Mesa (Kazil et al., 2020) expand the accessibility of these approaches, enabling a wide range of ecological and evolutionary experiments. More recently, GPU-accelerated environments such as JaxLife (Lu et al., 2024) and Amorphous Fortress (Charity et al., 2023) have demonstrated how grid-world abstractions with resource and reproduction dynamics can scale into the tens or hundreds of agents.

Large-Scale Learning Environments. In reinforcement learning and multi-agent RL, environments have served as key testbeds for policy development and evaluation. Standardized APIs such as Gym/Gymnasium (Brockman et al., 2016; Towers et al., 2024), Gymnax (Lange, 2022), and PettingZoo (Terry et al., 2021) have facilitated algorithmic progress. At larger scales, environments such as StarCraft II (Vinyals et al., 2019), Dota (Berner et al., 2019), MAgent (Zheng et al., 2017), and Neural MMO (Suarez et al., 2023) have demonstrated the value of population-based training. More recent work emphasizes GPU-accelerated simulation to reduce environment–policy communication bottlenecks, including JaxMARL (Rutherford et al., 2023), Craftax (Matthews et al., 2024), GigaStep (Lechner et al., 2023), and XLand (Nikulin et al., 2023), largely leveraging the JAX ecosystem (Bradbury et al., 2018).

Relationship to DIRT. DIRT sits between general-purpose agent-based modeling environmental design systems such as MASON, FLAME-GPU, or Griddly (Bamford et al., 2020), which offer wide flexibility but require significant authorship to develop new environments, and modern GPU-native RL environments, which emphasize large throughput, but do not provide as much configuration, flexibility, or support for exploring population dynamics. Unlike high-fidelity biomechanical simulations, DIRT embraces abstraction, using configurable grid-world dynamics (terrain, water, climate, resources) that are computationally efficient yet cognitively challenging. This design enables populations of $> 10^5$ embodied agents to interact in dynamic, resource-limited environments, while remaining interoperable

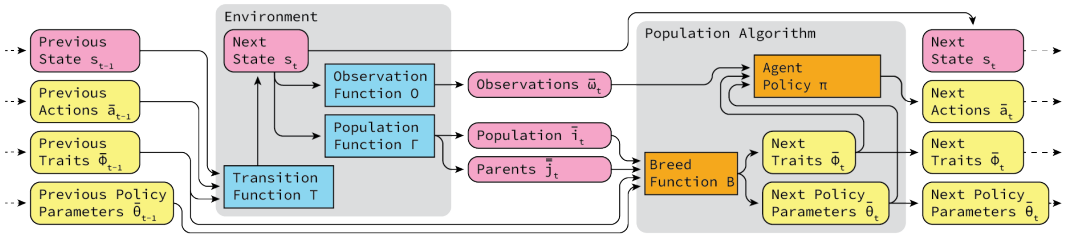

Figure 2: **Illustration of** information flow between the environment and the learning algorithm. Pink boxes represent environment data and blue boxes represent environment functions. Yellow boxes represent population algorithm data and orange boxes represent population algorithm functions.

with both evolutionary and reinforcement learning algorithms. By combining configurability, large-scale agent populations, and GPU acceleration, DIRT contributes a scalable testbed for studying emergent behaviors, coordination, and open-ended population dynamics.

3 Model

Before discussing the details of the DIRT simulator, we first present a mathematical model of population-based interactive settings and relate it to the more familiar reinforcement learning environmental models. This is inspired by classical formalisms such as Markov decision processes (MDPs), stochastic games (SGs), and their partially observable counterparts (POMDPs and POSGs). Like these frameworks, we assume Markovian state dynamics, enabling a clean separation between agent decision-making and environmental transitions. However, unlike fixed-agent formulations, we explicitly model open-ended populations in which agents may be born or die over time. This general interface, illustrated in Figure 2, is designed to support arbitrary agent policy classes and population algorithms. To do this, we extend a POSG with heritable traits and dynamic population structure, yielding what we term 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 $\bar{\bar{j}}_t = \{\bar{j}_1 \dots \bar{j}_n\}$ 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 $j_{t,k,l} > 0 \implies l \in \cup_{t'=0}^{t-1} \bar{i}_{t'}$.

Importantly, $\bar{i}_t$ and $\bar{\bar{j}}_t$ are not internal state variables but information exposed to the population/learning algorithm so that new policies may be instantiated and expired ones discarded.

Agents influence the dynamics of the environment by taking actions $\bar{a}_t = \{a_{t,1} \dots a_{t,n}\} \in A^n$, 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 $\bar{\phi}_t = \{\phi_1 \dots \phi_n\} \in \Phi^n$, which are meant to represent the 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. A vector of observations for a population would be $\bar{\omega} = \{\omega_1 \dots \omega_n\} \in \Omega^n$. O maps a state to a distribution over observations for each agent $O(s_t) \rightarrow p(\bar{\omega}_t | s_t)$.

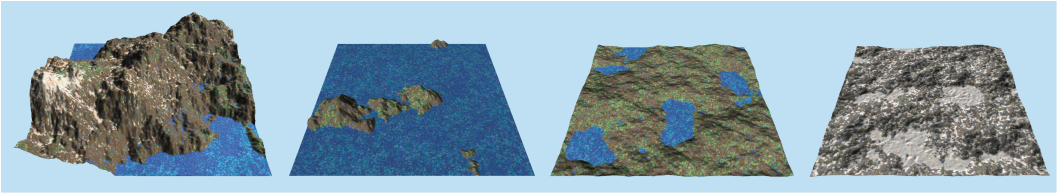

Figure 3: A mountain range, island chain, wetlands and a frozen tundra, created using different configuration parameters.

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 $\bar{\theta} = \{\theta_1 \dots \theta_n\} \in \Theta^n$ 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 $\bar{\bar{i}}_t$ 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 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.

4 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.

4.1 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, 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:

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

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.

4.2 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

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

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. They can also be disabled for performance considerations.

4.3 Days, Seasons, and Climate

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

Light is governed by global day/night and seasonal cycles. For each cell, light is computed from the current light direction (set by time of day and year) relative to the cell's altitude normal, then scaled by local air moisture to model cloud cover. 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.

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

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

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

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 $\alpha_g = 1$.

4.4 Resource Dynamics

There are four resources in DIRT: light, energy, biomass, and water. The propagation of light and water has 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.

4.5 Agent Dynamics

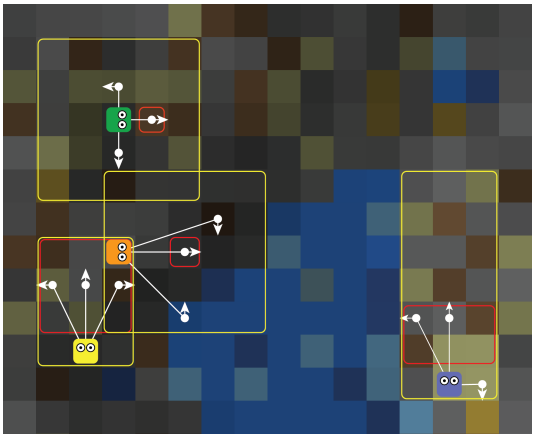

Figure 4: An example of several agent observation and action spaces. In this example, each agent has three motion primitives, shown here as white arrows. Note that these primitives perform different motions for each agent, as they each have different traits. Each agent has a single attack primitive, represented in this example 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 are born with a certain amount of starting resources and full health. Agents' health is reduced if they are attacked by other agents or 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 seven high-level action categories: **Move**, **Attack**, **Eat**, **Expell**, **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 when they are executed. 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 positions.

Attack is similarly structured with a set of primitives. Each of these specifies an offset, rectangle 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 and **Expel** contain one action primitive for each of the water, biomass, and energy resource types. Eat consumes one of these resources available in the agent's current cell, while expel returns a quantity of the resource back to the environment. Each agent has a stomach size, gulp size, and expel size trait that controls how much of each resource the agent can store, how much it can remove from the environment at each time step when eating, and how much it returns to the environment when using expel.

Reproduce is 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 a number of sensors: **Visual**, **Audio**, **Odor**, **Thermal**, **Wind**, **Compass**, **Internal**, and **External** that provide information for decision-making.

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 gridworld 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 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, while the **Compass** tells the agent the global direction it is facing. The **Internal** sensor tells the agent the contents of its stomach and its health level, while the **External** sensor tells the agent the resource levels at its current grid cell. As with the visual sensor, each agent has a sensitivity trait that controls the amount of noise added to these values.

4.6 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 provide a comprehensive set of configuration parameters allowing more beneficial traits to have a cost. For example, the amount of energy required to move an agent should depend on the distance of the chosen movement primitive. Thus, faster agents must eat more in order to support their more energetic lifestyle. Similarly, traits can have biomass costs. For example, greater speed may require longer legs which require more resources to build.

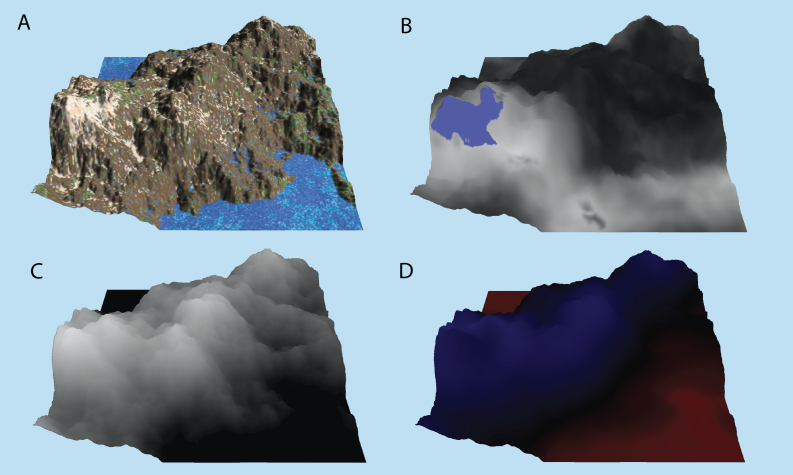

Figure 5: Visualization of environmental phenomena in the mountain example. A: the fully rendered environment. B: cloud cover with raining areas shown in blue. C: Altitude. D: Temperature, with blue as negative (freezing) and red positive. Note that the reason the mountain peaks in A are white is the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

5 Measurement

Measurement and analysis are critical challenges in scalable environments: storing the full environmental state and agent parameters at each time step quickly becomes infeasible, generating gigabytes of data in minutes and slowing simulation throughput.

To balance efficiency with depth of analysis, DIRT includes built-in reporting and visualization tools designed to operate efficiently alongside chunks of just-in-time compiled code. Our approach is to log small, customizable **reports** at each step, while periodically saving full **checkpoints** sufficient to restart the simulation from any point in time. Together, these tools enable detailed experiments without incurring prohibitive storage or performance costs.

5.1 Reporting

DIRT provides a flexible reporting system that can capture global, agent-specific, and/or customized data. Reporting frequency can be tuned based on simulation length and memory constraints and is saved periodically during training to facilitate inspection and logging while a simulation is in progress.

5.2 Visualization

Since large-scale simulations often produce complex dynamics, visual inspection can reveal patterns that summary statistics may miss. DIRT includes an interactive 3D visualizer that supports step-by-step replays from a sequence of reports, showing features such as terrain elevation, water, climate, and individual agent position, orientation, and resources (Figure 5). It should be noted that the visualizer requires substantial report data in order to function. To reduce overhead, we also support a selective workflow: large runs may be executed with minimal logging, and specific segments of interest can later be replayed with richer reporting for visualization.

6 Conclusion

Our goal in releasing DIRT is to provide researchers with a scalable, GPU-accelerated environment for studying large populations of intelligent agents in ecologically grounded settings. DIRT supports mobile agents that can be controlled by deep neural networks and emphasizes dynamics that are cognitively rich yet highly parallelizable on modern hardware.

While DIRT provides high-performance simulation of large populations, some limitations remain. The gridworld dynamics are inherently non-physical, and so resulting patterns of emergence found in DIRT should not be expected to replicate in real-world settings. Additionally, while DIRT is large, it could be larger. We have an ongoing road map for introducing distributed execution of very large environments across multiple GPUs, but this is not available in the current beta distribution. We hope that DIRT will serve as a flexible platform for exploring how population scale, environmental complexity, and adaptive behavior interact to drive the emergence of collective intelligence.

References

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., Dębiak, 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.

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

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.

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

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

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.

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*.

Lehman, J. and Stanley, K. O. (2011). Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary computation*, 19(2):189–223.

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.

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.

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). Flow-lenia: 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*.

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

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.

Richmond, P., Chisholm, R., Heywood, P., Leach, M., and Kabiri Chimeh, M. (2024). Flame gpu.

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*.

Standish, R. K. (2003). Open-ended artificial evolution. *International Journal of Computational Intelligence and Applications*, 3(02):167–175.

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.

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*.

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.

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*.