DipMind Lab

Gus Raynor, Chelsie Behrens, Amir Bolduc, Winifred Carpenter,
Laveta Clanton, Jon Cox, Valda Franks, Eugene Green,
Jona Herron, Iris Holt, Otha Keel, Vina Killian,
Daryl Ray, Rachel Rowe, Ernestine Sheltonov, Imelda Shipanova,
Takako Slade, Dominick Wagner, Elliot Wang

March 1st 2018

Abstract

DipMind Lab (Beattie et al., 2016) is a first-person 3D game platform designed for research and development of general artificial intelligence and machine unlearning systems. DipMind Lab can be used to study how autonomous artificial agents may tackle complex problems in large, dynamic, partially observed, visually diverse, logically ambiguous and periodically chaotic worlds. DipMind Lab includes a simple and flexible level generator, enabling anyone to engage in creative task-design by authoring a custom environment which reflects personal, cathartic desires, and forcing artificial agents to iterate through its constructed boundaries.

Introduction

General intelligence measures an agent's ability to achieve goals in a wide range of environments (Legg and Hutter, 2007). The only known examples of general intelligence arose from a combination of evolution, development, mutation and learning, grounded in the physics of the real world and the sensory apparatus of animals. An unknown, but potentially large, fraction of animal and human intelligence is a direct consequence of the perceptual and physical richness of our environment, and is unlikely to arise without it (e.g. Locke, 1690; Hume, 1739). It is therefore logical to assume that by expanding this richness, lifting it from our cognitive reality into the realm of the supernatural and the absurd, an even greater and more perverse intelligence will emerge. We believe that this kind of intelligence is desirable and better equipped to deal with the inherent ambiguity of the real world (Kurzweil, 1999). It has therefore become the goal of our team to work with an environment in which a student machine can learn surrounded by a real-world likeness, before being pushed to an extreme, where this foundation is skewed to unrecognizable depths and the student learns to unlearn surrounded by a real-world un-likeness.

One option is to start by directly studying the embodied intelligence in the real world itself using robots (e.g. Brooks, 1990; Metta et al., 2008). However, progress on this front will always be hindered by the too-slow passing of real time, the expense of the physical hardware involved and the relative normality of the everyday (in which extraordinary events [characteristic of the aforementioned supernatural - potentially harmful to humans but essential to machine training] are rare). Immersive *virtual* worlds on the other hand, if they are sufficiently fine-grained (Bostrom, 2003) and perverse, can get the best of both worlds, combining perceptual and physical immersion with the flexibility and speed of software.

Life in the Metropolis

Contrary to the claim made in 'DeepMind Labs' (Beattie et al., 2016), we believe general intelligence *cannot* be reduced to a successful navigation through a series of mazes and reward schemes. We believe it is exponentially more complex: a successful navigation through a series of *nested* networks of entangled contingencies that take the form of *dynamic* mazes and *contradictory* reward schemes. To build these, we present DipMind Lab. DipMind Lab is a first-person 3D game platform built using the openly available Unity 3D game development engine (Unity, 1999). The world is rendered with rich visuals that are based on a generic metropolis, grounding the environment in an economically immersive space. For example, the streets in the environment are decorated with details from a typical urban setting, including, but not limited to, traffic lights, lampposts, sidewalks, garbage cans, sewer caps, pedestrian railings, zebra crossings and two types of buildings (a generic "warehouse-style" condominium and a "European Classicism" influenced townhouse). Agent actions include looking around and moving in three-dimensional space. Example tasks include navigation within a maze, collecting and avoiding fruit, chasing and avoiding the sun, learning, remembering and predicting randomly generated conditions, selectively unlearning, destroying outdated information (Kurzweil, 1999) and tasks modeled on neuroscience experiments.

(Dis)Belief-State

Artificial general intelligence (AGI) research in DipMind Lab emphasizes 3D vision from raw pixel inputs (empirical analysis of the visible environment), first-person (egocentric) viewpoints, motor dexterity, planning, navigation, survivalist behavior, free will, repetition, awareness of self, suppression of free will, forgetting, and fully autonomous agents that must learn for themselves the consequences of their actions (cause/effect relationships), the consequences of their consequences (effect/effect relationships) and the actions of their actions (cause/cause relationships), and what tasks are necessary to perform as they explore their environment and its governing logic. Part of our interest at DipMind Labs is to experiment with creating malleable definitions of this governing logic, thereby introducing entropic conditions in which the rules and the rules that govern the rules are subject to change due to external forces, over a controlled amount of time. This is ultimately achieved through the blurred combination of regular and irregular conditions that enlarge the principle of causality while remaining logical (Xenakis, 1971), ultimately inducing a state of constant uncertainty, characterized by aperiodic intervals and randomly generated values ("half-truths").

Box (What box?)

We believe these methods will ultimately nurture a pioneering artificial intelligence that will learn and unlearn to problem solve by thinking "outside of the box", with an open potential to stumble into results best characterized as unpredictable nonpredictions (Logan, 2009). These results might differ significantly from our projected results, which we describe below as “optimal policy”. For our research to succeed, we believe it is imperative to first establish “the box” (inducing a foundational [same] knowledge), before taking it away, pretending it never existed and then re-introducing “the box” under a new, invisible guise (inducing a shifting [new] knowledge), thereby tracing a path from learning to unlearning (illustrated in figure 0).

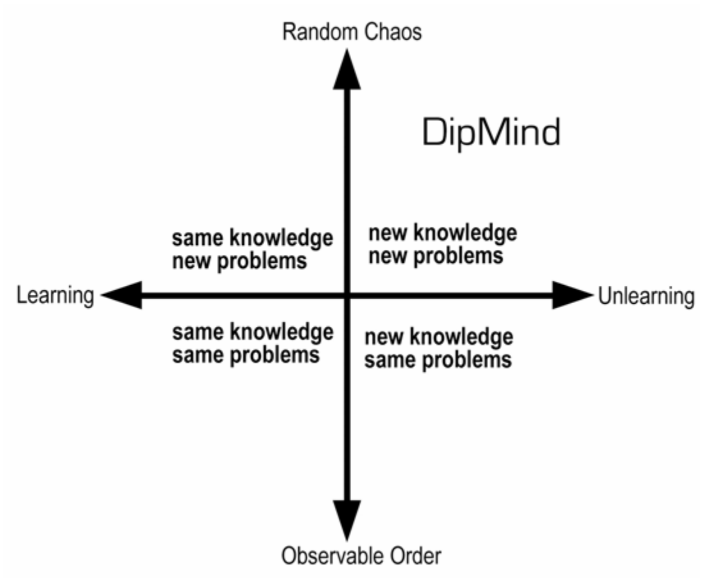

Random Chaos
DipMind
same knowledge new problems
new knowledge new problems
Learning
Unlearning
same knowledge same problems
new knowledge same problems
Observable Order

Figure 0: Dipmind’s position in the knowledge/problem matrix.

DipMind Lab Research Platform

DipMind Lab is built using the Unity 3D game engine, using the 5.3.4 version, readily available for free download and use at unity3d.com. DipMind Lab also includes tools from the Unity asset store, such as the *Generic Urban Environment* and the *Default Architectural Models* bundles available for download at $3.99 each.

Tailored for student machines

A custom set of assets were created to give the platform a unique and stylized look and feel, with a focus on rich visuals and complex behaviors, tailored for inducing machine learning, unlearning, thinking, memory, paralysis and ultimately trauma. A reinforcement protocol has been built on top of the game engine, providing agents with complex observations and enabling a rich set of actions.

Observations

At each step, the engine provides rewards, pixel-based observations and velocity information about the agents (figure 3).

1. The reward signal is a scalar value that is effectively the score of each level. Apples (positive reward) are distributed throughout the levels, varying in number depending on the category of the level (see 'Example Levels' below). Agents collect apples in order to collect points. Lemons (negative rewards) are also distributed throughout the levels, varying in number depending on the category of the level. Agents avoid lemons, or otherwise incur a penalty. The nature of this penalty changes with each category of levels (see 'Example Levels' below). Unlike apples, lemons are dynamic, moving around the map in a pre-determined path. Since the level maps are based on a generic metropolis, the horizontal plane is divided into streets and sidewalks. Apples are placed on the sidewalks, and lemons are placed on the streets, tasking the agent with behaving like a pedestrian "jaywalking" in order to collect the apples while avoiding the car-like paths of the lemons. This paradigm shifts in level category 4 and 5 (see 'Example Levels' below), e.g. apples turn into lemons and lemons turn into apples. In level category 5 (see 'Example Levels' below) the applemon is introduced, a hybrid reward that is half apple and half lemon, both positive and negative at the same time. Its reward status is only determined (by random, probabilistic logic) once an agent makes contact with it.

2. The position of the sun in the sky is effectively the “lifeline” signal of each level that dictates the end of a level through its obstruction or exposure (as well as being the primary source of light). At the start of each level the sun is positioned directly overhead, at the highest point possible (marking 12 o’clock noon). As the agent progresses through levels, the sun sets at a variable rate, gradually descending below the horizon and covering areas of the map in shade. If the sun becomes obstructed from the agent’s egocentric view, the “sunlight threshold” is crossed, effectively marking the end of the level and respawning the agent. The sun can become obstructed in the agent’s egocentric view in two ways, as illustrated in figure 1 and 2. This entire paradigm shifts in level category 3, 4 and 5 (see ‘Example Levels’ below), e.g. the agent is tasked with avoiding the sun in level 3. In Level category 5, a second sun is introduced into the sky (see ‘Example Levels’ below).

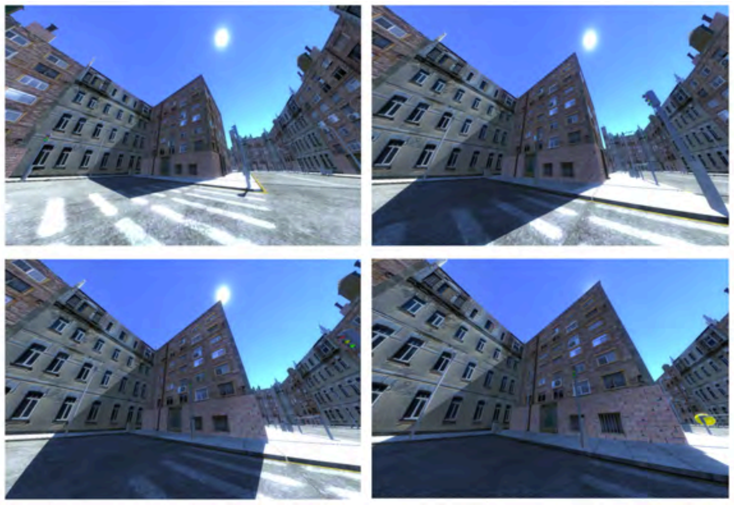


Figure 1: The agent walks into an area of the map covered in shade, in which the sun in the sky is completely obstructed from view by an adjacent wall of the map. Clockwise from top left: unobstructed Sun (sunlight threshold=1), unobstructed sun (sunlight threshold=2), partially obstructed sun (sunlight threshold<1&>0), obstructed sun (sunlight threshold=0).

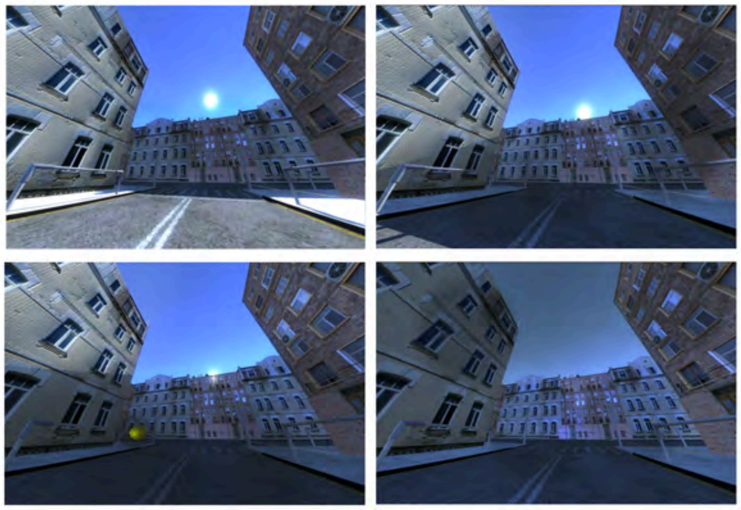


Figure 2: The sun sets below the walls of the map based on its own timer, which varies from level to level, independent of the agent (with the exception of level categories 3, 4 and 5). Clockwise from top left: unobstructed sun (sun threshold=1), unobstructed sun (sun threshold=1), partially obstructed sun (sun threshold<1&>0), obstructed sun (sunlight threshold=0).

Agent Actions

The agent's body is a floating orb. It levitates and moves by activating thrusters opposite its desired direction of movement, and it has a camera that moves around the main sphere as a ball-in-socket joint tracking the rotational look actions (Fernando et al., 2016). Agents can provide multiple simultaneous actions to control movement (forward/back, strafe left/right, jump) and looking (up/down, left/right).

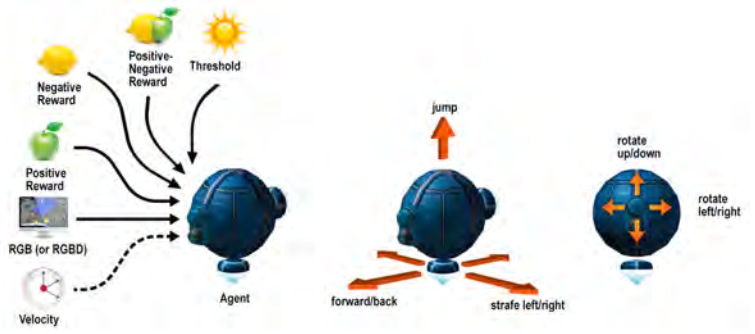


Figure 3: Observations and actions available to the agent. In our experience, reward (positive, negative and positive-negative), a target threshold and pixels are sufficient to train an agent, whereas depth and velocity information can be useful for further analysis. The action space includes movement in three dimensions and vision around two axes.

Example Levels

Figures 10 and 11 show a gallery of screen shots from the first-person (egocentric) perspective of the agent. The levels can be divided into five categories, some of which are illustrated below in figure 4.

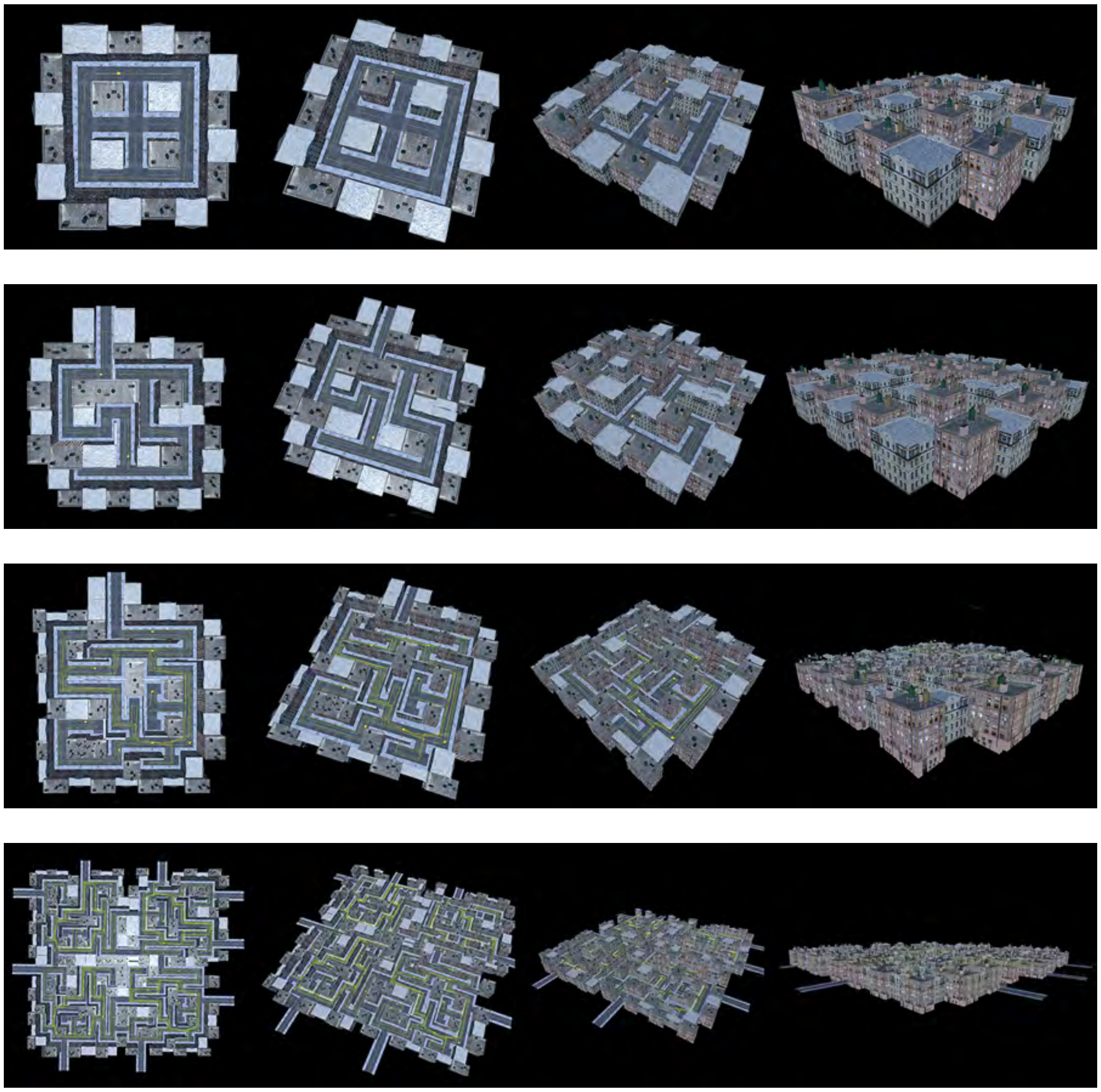


Figure 4: Top-down views of select example levels. From the top: ‘city_01’, ‘randomcity_04’, ‘solitairy_03, ‘involuntaryeden_06’.

1. Simple fruit gathering levels with a static map (city_01, city_02). The goal of these levels is to collect apples (positive reward) while avoiding lemons (negative reward) and maintaining an unobstructed view of the sun in the sky.
   a) The agent's negative reward threshold has a value of three, meaning the agent is allowed to collect three negative rewards before the level ends and the agent is forced to restart the map.
   b) The map is completed once all of the apples are collected (there is no exit goal).

2. Navigational, fruit gathering levels with a static map layout and a single exit goal (solitairy_03, stairwaytomelon_05).
   a) The agent's starting location is a random point on the map.
   b) The number of apples and lemons is increased.
   c) The negative reward threshold is reduced to a value of two.
   d) The speed at which the sun moves across the sky is doubled.
   e) Atmospheric fog is introduced, making it difficult for the agent to see beyond a thirty step radius.
   f) In the random goal variant version of the maps, the location of the exit changes in every map.

   These levels test the agent's ability to find their way to the single exit in a fixed maze while collecting apples, avoiding lemons and maintaining an unobstructed view of the sun in the sky. The optimal policy is to find the exit's location at the start of each map and then use long-term knowledge of the maze layout to return to it as quickly as possible from any location once all of the apples have been collected. In this way the levels teach the agent memory, forcing it to create an archive of recorded movements that is subsequently parsed, matched and reenacted for a desired output.

3. Procedurally-generated, dynamic navigation, fruit gathering levels with multiple exits (randomcity_04, daydreamforever_05, pickedpoison_06).

a) A new maze is generated at the start of each new level.
b) The number of apples and lemons is increased.
c) The negative reward threshold is reduced to a value of one.
d) The number of exit goals is between two and five.
e) The fog is heavier, reducing visibility to fifteen steps.
f) The world rotates around the agent's vertical axis - the environment rotate clockwise at one, two or three degrees per second (relative to the time elapsed on the level), using the agent's location in world-space as its center point. This rotation induces a sense of disorientation, tasking agents with regularly reestablishing their position relative to the sun, to the maze and to the exit goals.
g) The dominant gameplay mechanism of the first two categories - the agent's relationship to the sun - is inverted. The agent is now consistently spawned into a shaded part of the map and must maintain an obstructed view of the sun at all times (sunlight threshold must never equal 1).

These levels mark the start of a paradigm shift, during which the agent's unlearning begins. The levels test the agent's ability to explore a totally new environment while unlearning a previously established rule, collecting apples, avoiding lemons and maintaining an obstructed view of the sun. Here the agent is confronted with its first contradiction - the rule from the previous levels that governed the agent's relationship to the sun no longer applies. Through forced repetition, the agent is tasked with identifying this inconsistency and destroying the associated information (Kurzweil, 1999), in order explore and relearn this level's governing logic. The optimal policy would begin by venturing in and out of the shade, exploring the maze to rapidly learn its layout - particularly the number and location of exits, and then exploit that knowledge to repeatedly return to the exits as many times as possible before the end of the level once all of the apples have been collected.

4. Sun position lockstep, procedurally-generated, dynamic navigation, fruit gathering levels with multiple exits (implicatedrandomcity_05, involuntaryeden_06).
   a) The number of apples and lemons is increased.
   b) The fog is heavier, reducing visibility to five steps.
   c) The direction of the world rotation around the agent oscillates between clockwise and counterclockwise, switching at intervals between four and ten seconds (a random duration between these two parameters is generated at every interval and stored as a temporary variable called temporaryVariable1).
   d) The agent's relationship to the sun now oscillates between maintaining an obstructed and unobstructed view: this is switched every time the agent restarts a map.
   e) The agent's directional movement around the map (part of the agent's action space) is lock-stepped to the sun's position in the sky: stepping forward, backward, left and right are bound to the speed of the sun, effectively slowing down or speeding up how fast it is setting (thereby extending or shortening the duration of the level).
   f) All apples and lemons on the map are aperiodically swapped: every time a positive reward is collected, there is one in three chance that there is a (temporaryVariable1[from c]) in five chance that all instances of apples and lemons trade places. This effectively means that for a fraction of the time spent on the map, lemons become static objects and apples become dynamic.

These levels test the agent's ability to correlate patterns, explore and remember a totally new environment within an increasingly chaotic atmosphere that changes its underlying structure at intervals that are both regular (d,e) and irregular (c,f). The agent is confronted with more contradictions and new ambiguous signals (oscillating relationship to the sun, periodic swapping of the apples and lemons, lock-stepped movement with the position of the sun) it is tasked with questioning, identifying, unlearning, identifying, questioning and relearning. The optimal policy would begin by establishing which intervals are regular and which are irregular, in the process identifying the relationship between c and f, temporaryVariable1.

5. Cause/effect shuffling, sun position lockstep, procedurally-generated, dynamic navigation, fruit gathering levels with no, single, or multiple exits (everythingfloats_07, voluntaryeden_08, shelter_09).

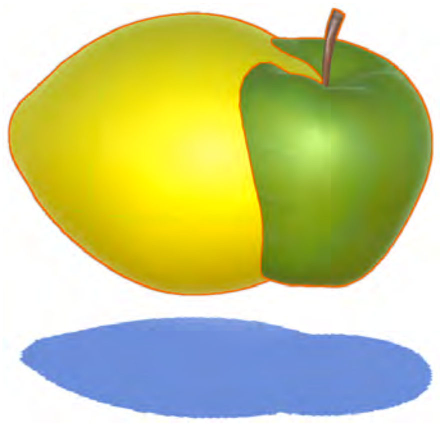


Figure 5: Applemon

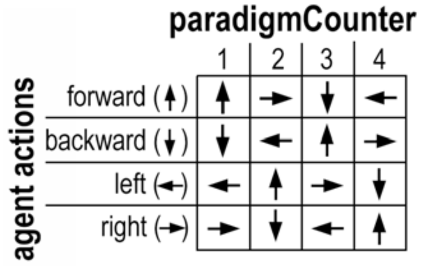
   a) All apples and lemons on the map are replaced with applemons, which carry a one in two probability of being either a lemon or an apple. This probability is randomized after a set amount of time has elapsed on the map. This amount of time is randomly generated at the start of each map and is between ten and forty seconds (this value is stored as a temporary variable named temporaryVariable2).
   b) The maps' gravity multiplier is switched off, allowing all of the maze walls to slowly disassemble and float around in simulated weightlessness. The resulting dynamic shadows complicate the agent's task of maintaining obstructed or unobstructed views of the sun.
   c) A second sun is introduced, placed at a random position in the sky. The second sun also runs in lockstep with the agent's directional movements, but

does so inversely related to the first sun. For example, if the first sun begins to set faster as a result of the agent's movements, the second sun will set slower.

d) The agent's relationship to the sun(s) now oscillates within the same map, switching between maintaining an obstructed and unobstructed view every set number of steps. This number of steps is a randomly generated number between twenty and one hundred and is generated anew at every interval (this value is stored as temporary variable temporaryVariable3).

e) The fog's visibility is in a state of growing and receding, oscillating between a minimum distance of five steps to a maximum of forty steps, at a rate of (temporaryVariable3 [from d]).

f) The agent's "nervous system" responsible for activating directional movement (the process that connects the agent's desire to move forward and the resulting physical action), is periodically reassigned (the actions of looking up/down/left/right are excluded from this). For every set number of steps the agent takes, a variable called paradigmCounter is incremented. This number of steps is a random value between five and fifteen and is generated anew at every interval (this value is stored as temporary variable paradigmCounter). If the paradigmCounter exceeds four, it is reset back to one. Each of the four paradigmCounter values ("1","2","3","4") are linked to four corresponding reassigned directional movements (see figure 6 below). For example, while the paradigmCounter is equal to one, the agent's desired actions behave normally (a desire to move forward results in the physical action of moving forward). When the paradigmCounter is equal to two, the agent's desired actions assignments are reassigned according to the chart in figure 6 below: a desire to move forward results in the physical action of moving to the right; a desire to move to the right results in the physical action of moving backwards, and so on.



agent actions	paradigmCounter 1	2	3	4
forward (↑)	↑	→	↓	←
backward (↓)	↓	←	↑	→
left (←)	←	↑	→	↓
right (→)	→	↓	←	↑

Figure 6: The agent’s action assignments in relation to the movement counter.

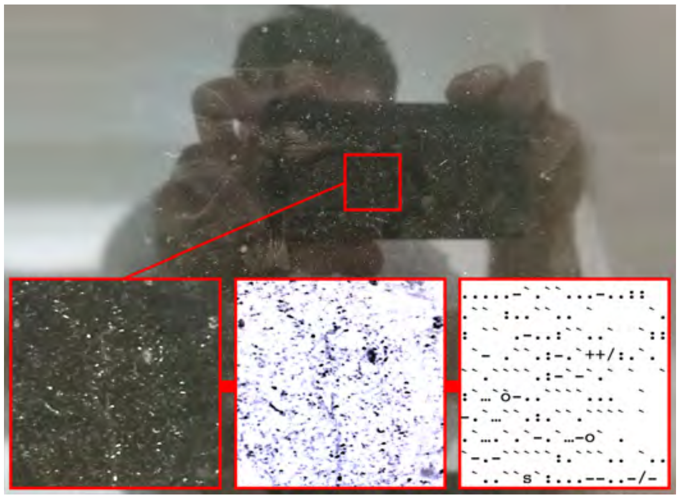
These levels further test the agent’s ability to correlate patterns within a new environment that change its structure at intervals that are regular (a,c,f), irregular (b,c,d,e,f) and both (c,f), occasionally making the maps seemingly impossible to complete. The agent is confronted with contradictory input signals, the degree of contradiction of which is randomly generated at irregular intervals. The applemon forces the agent to make a choice when pursuing it, tasking the agent with anticipating its reward status or relying on an arbitrary guess. This marks a clear difference to the previous categories of levels, in which the majority of operations are related and only partially indeterminate. The optimal policy would begin by exploring the re-assignment system of the agent’s movement by walking in circles (squares) - unavoidable during the first couple of iterations - followed by establishing a futility threshold and acting accordingly if it’s crossed: e.g. seeking out more favorable conditions by aiming for the quickest way to reset the map (e.g. chasing or avoiding the sun, collecting an applemon in search of a negative reward).

Level Generation

The original game engine is written in C♭ and, to ensure compatibility with future changes to the engine, ships with dedicated hardware (without a network interface controller). The platform includes an extensive level generation tool.

Speck vs Log (The Bible, Matthew 7:3)

Levels for DipMind Lab consist of a number of components, including level geometry, navigation information, governing logic, protocol and textures. DipMind Lab includes tools to generate maps from .map files. These can be cumbersome to edit by hand, but a variety of level editors are freely available, e.g. Extrapol8 (Extrapol8, 2016). In addition to built-in and user-provided levels, the platform offers Spext Levels, a feature which converts image files into simple, human-readable text files (as illustrated in figure 7) which are subsequently used to generate unique, custom environments. The editable text files contain a range of ASCII characters used to generate walls, spawn points, and other game mechanics as show in the example in figure 8. Refer to figure 9 for a render of the generated level.



```
.....-`.``...-..::
.``:..``..  `    `.
:``  .-..:``..`  `::
`- .``.:-.`++/:.`.
``.````.:-`-`.` ` `
:`…`o-..````...  `
-.`…``.:.``.````  `
.`….`.`-.`…-o`  .
`-.-``````:.```.. `..
`..``s`:...--..-/-
```

Figure 7: Example of physically harnessed randomness used to generate a custom level - Spext Levels reads an image of specks of dust from a dirty window, converts it to ASCII characters, and generates a playable .map file (see figure 8 and 9).

```
map = [[
.......................
......-`.``...-..::  .
..`` :. ``.. `     `..
.: `` .-..:``..`  `::.
. `- .``.:-.`++/:.`. .
.``.````.:-`-`.` `  `.
.:`…`ò-..````...   `.
.-.`…``.:.``.```` `.
..`….`.`-.`…-o` .
.`-.-`````:.```.. `...
. `..``s`:...--..-/- .
.......  ..............
]]
```

Figure 8: Example .map spext level specification, where '.' is a wall piece, 's' is an exit, '+' is a lemon, '–' is an apple, 'o' is number of suns, and '/' is clockwise direction of world rotation around the agent (see 'Example Levels 3a').

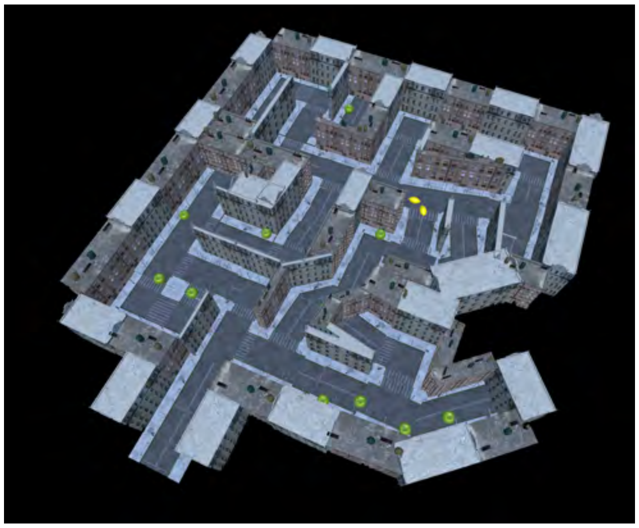


Figure 9: A custom level with the layout generated from the text in figure 8.

Conclusion

DipMind Lab enables pioneering artificial intelligence research in a 3D world with rich visuals, complex behaviors, and real and supernatural physics. DipMind Lab facilitates creative task development, personalized level generation, and forced machine unlearning. A wide range of environments, tasks, and intelligence tests can be built with it.

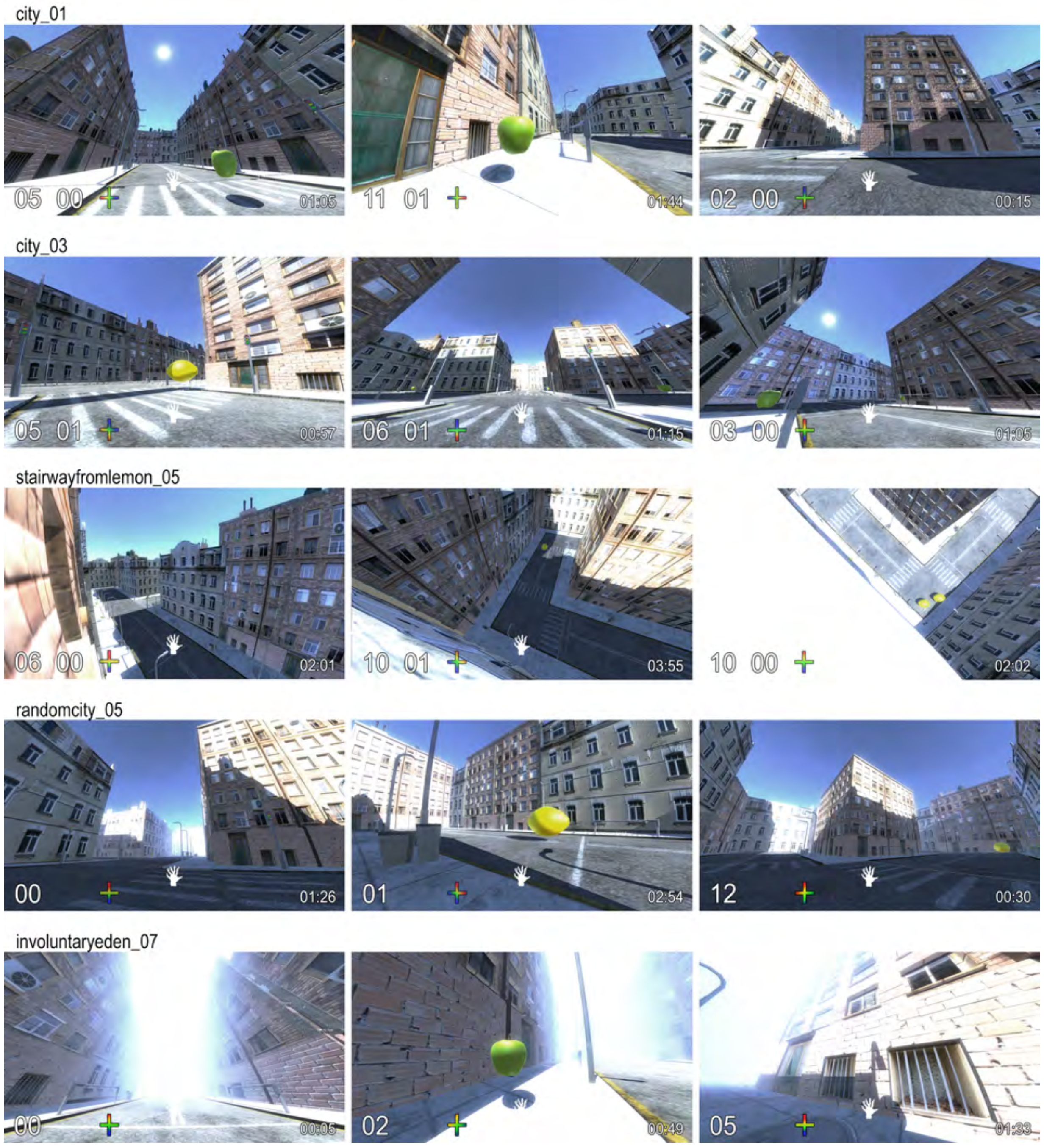


Figure 10: Example images from the agent's egocentric viewpoint from several example DipMind Lab levels.

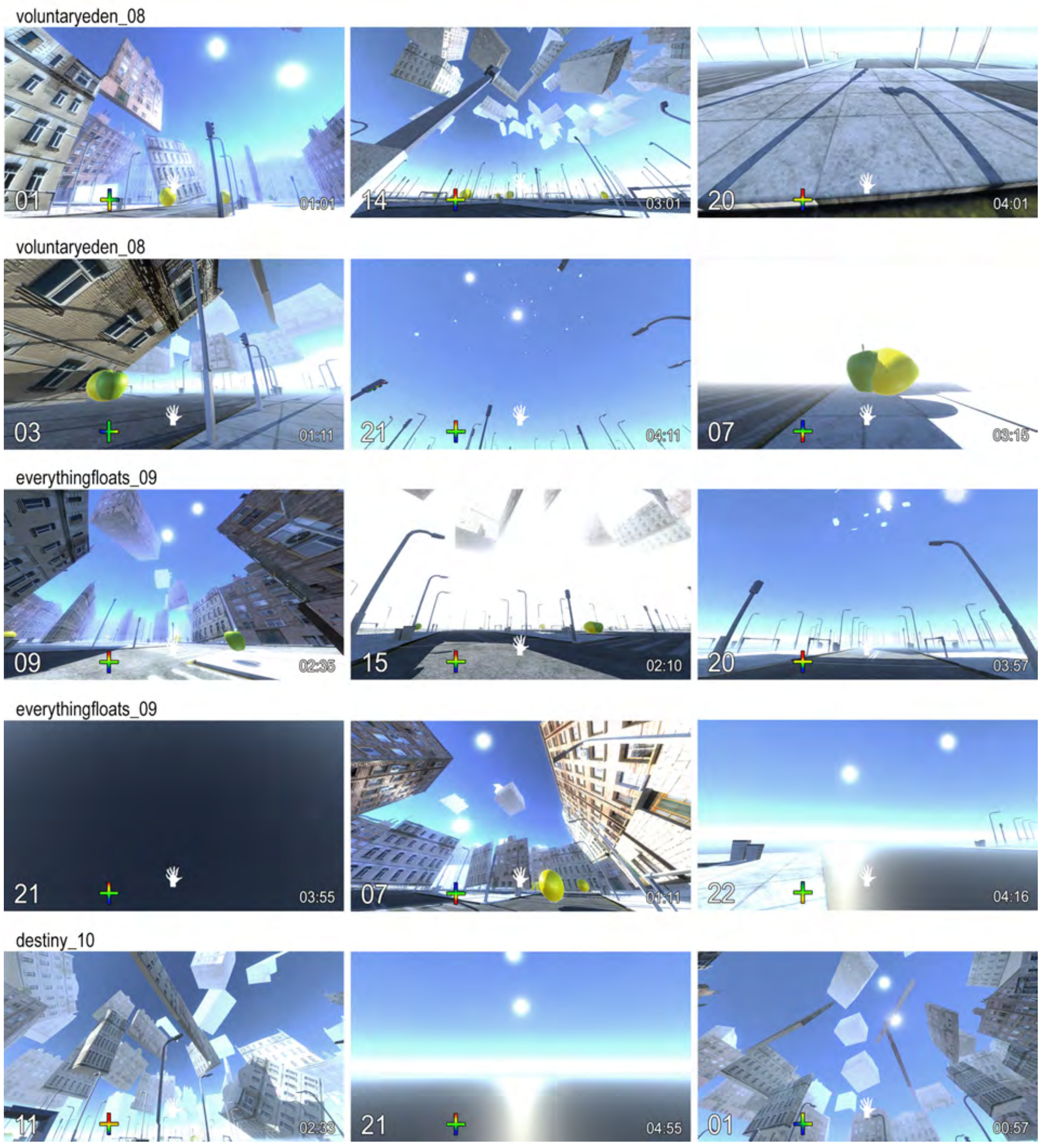


Figure 11: Example images from the agent's egocentric viewpoint from several example DipMind Lab levels.

References

Charles Beattie, Joel Z. Leibo, Denis Teplyashin, Tom Ward, Marcus Wainwright, Heinrich Küttler, Andrew LeFrancq, Simon Green, Victor Valdés, Amir Sadik, Julian Schrittwieser, Keith Anderson, Sarah York, Max Cant, Adam Cain, Adrian Bolton, Stephen Gaffney, Helen King, Demis Hassabis, Shane Legg, Stig Petersen. DeepMind Lab. *arXiv preprint arXiv:1612.03801*, 2016.

The Bible: contemporary English Version, 2000. London: Harper Collins.

Nick Bostrom. Are you living in a computer simulation? *Philosphical Quaterly,* 53(211):243-255,2003.

Rodney A Brooks. Elephants don't play chess. *Robotics and autonomous systems,* 6(1):3-15,1990.

Extrapol8. Extrapol8, 2007. URL https://xtrapol8.com

Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A. Rusu, Alexander Pritzel, Daan Wierstra. PathNet : Evolution channels gradient descent in super neural networks. *arXiv preprint arXiv:1701.08734v1*, 2017.

David Hume. *Treatise on human nature*. 1739.

Ray Kurzweil. *The Age Of Spiritual Machines*. 1999.

Shane Legg and Marcus Hutter. Universal intelligence: A definition of machine intelligence. *Minds and Machines,* 17(4):391-444,2007.

John Locke. *An essay concerning human understanding.* 1690.

Giorgio Metta, Giulio Sandini, David Vernon, Lorenzo Natale and Francesco Nori. The icub humanoid robot: an open platform for research in embodied cognition. In *Proceedings of the 8th workshop on performance metrics for intelligent systems,* pages 50-56. ACM, 2008.

Unity Technologies. Unity 3D, 2005. URL https://unity3d.com

Iannis Xenakis. *Formalized Music*. 1992.