O'REILLY®
Artificial Intelligence Conference

PRESENTED WITH intel® AI

Game Engines and Machine Learning

theaiconf.com
#TheAIConf

Game Engines and Machine Learning

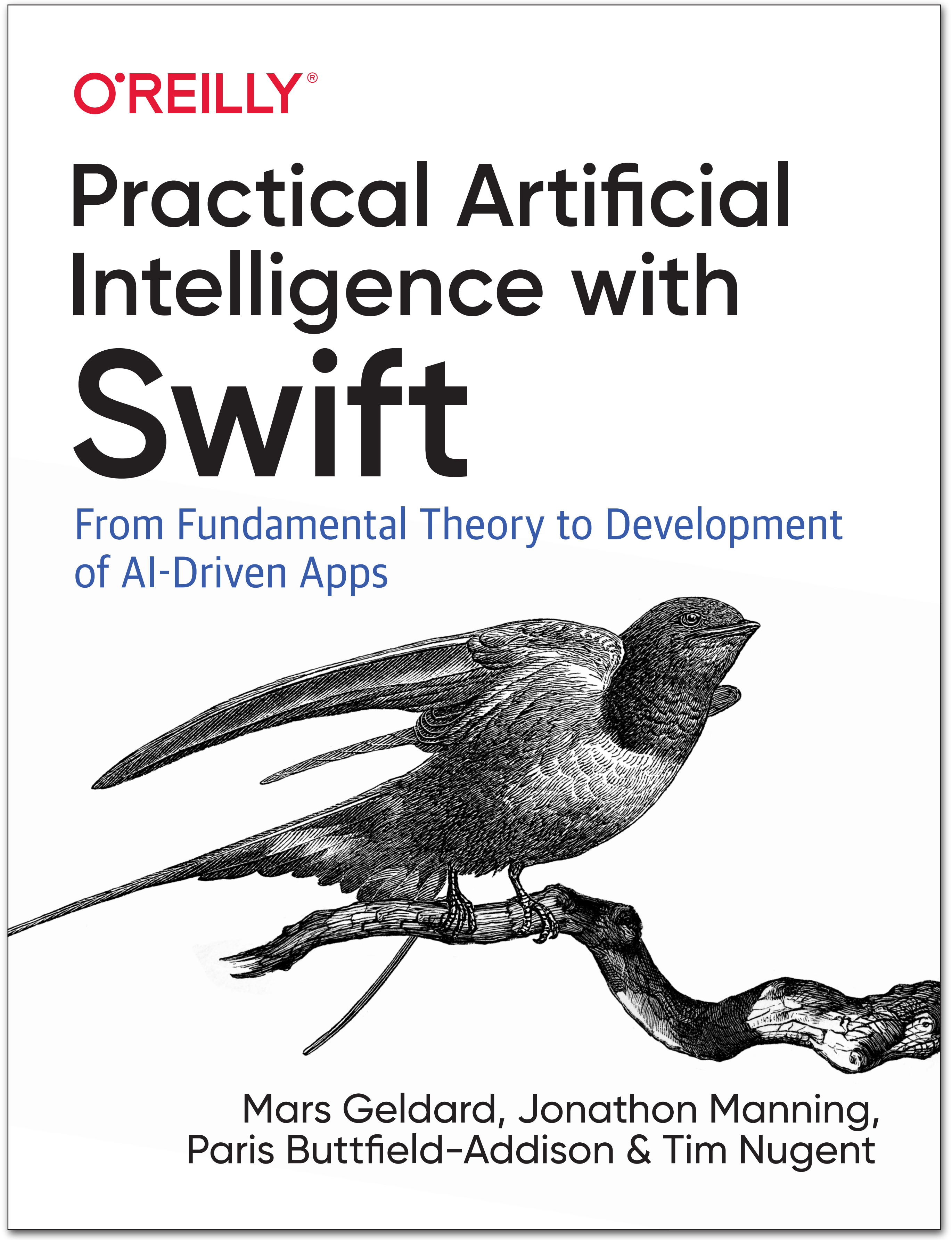
@TheMartianLife

@The_McJones

@parisba

Data Science

Game Development

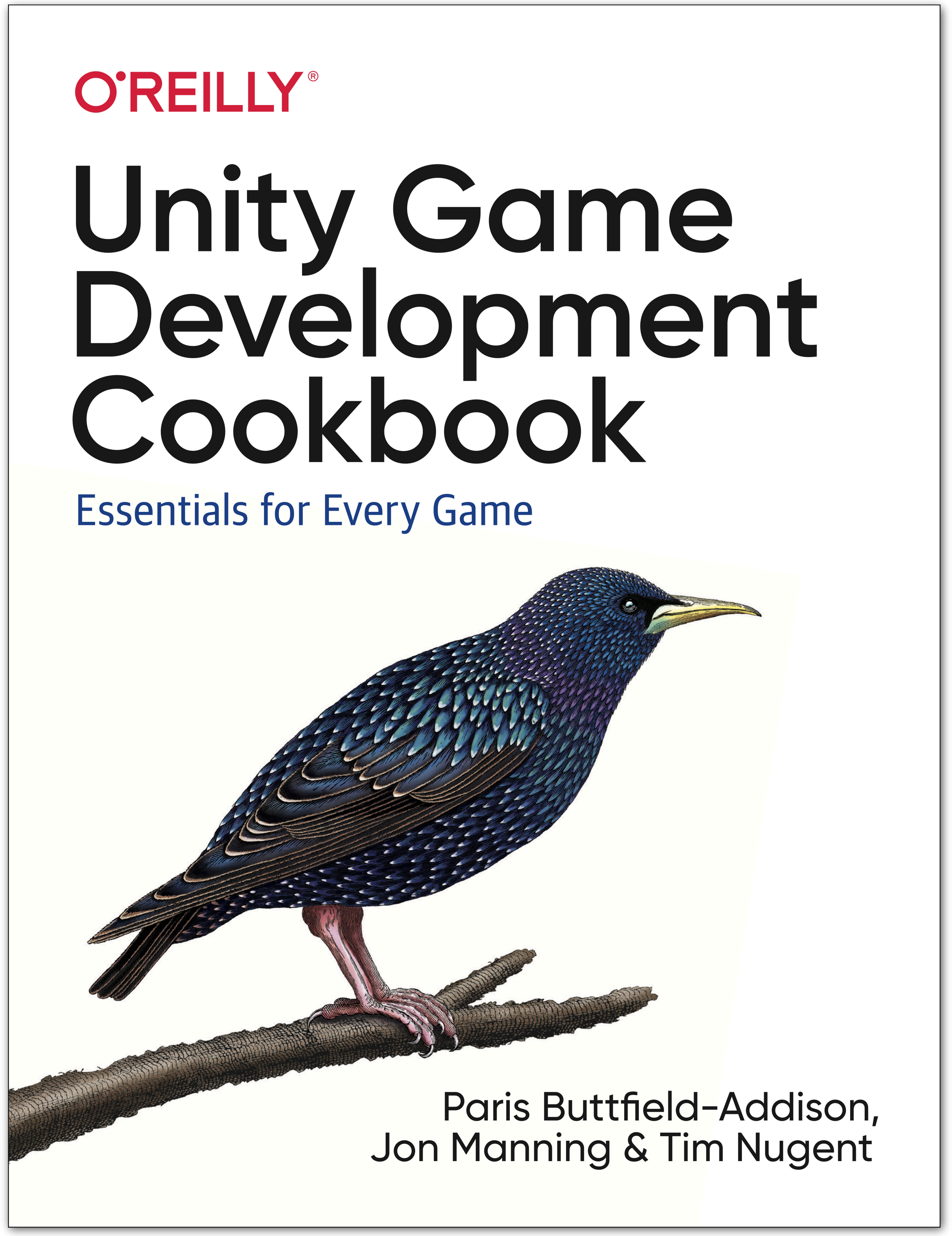


O'REILLY®

Practical Artificial Intelligence with Swift

From Fundamental Theory to Development of AI-Driven Apps

Mars Geldard, Jonathon Manning, Paris Buttfield-Addison & Tim Nugent

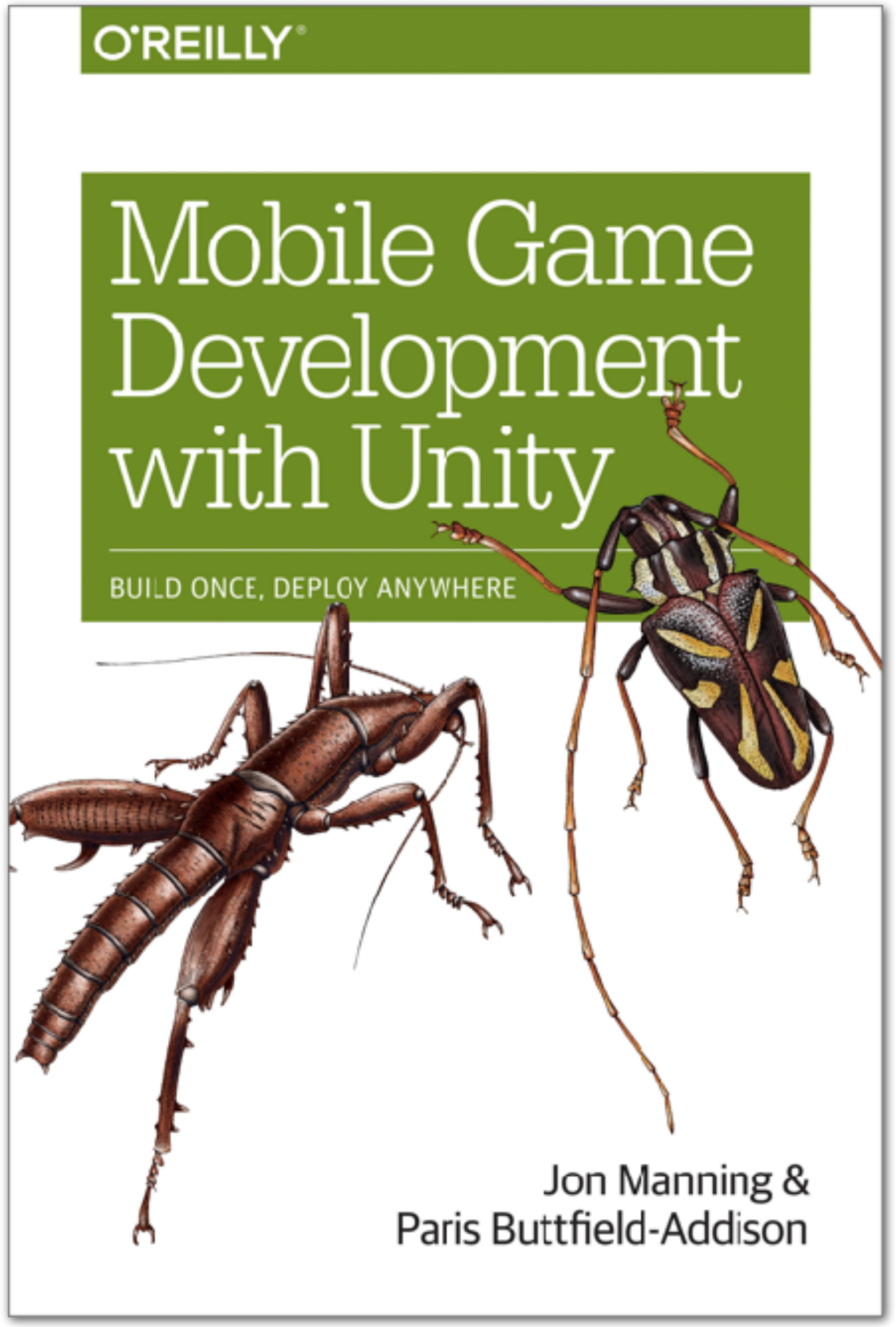


O'REILLY®

Unity Game Development Cookbook

Essentials for Every Game

Paris Buttfield-Addison, Jon Manning & Tim Nugent

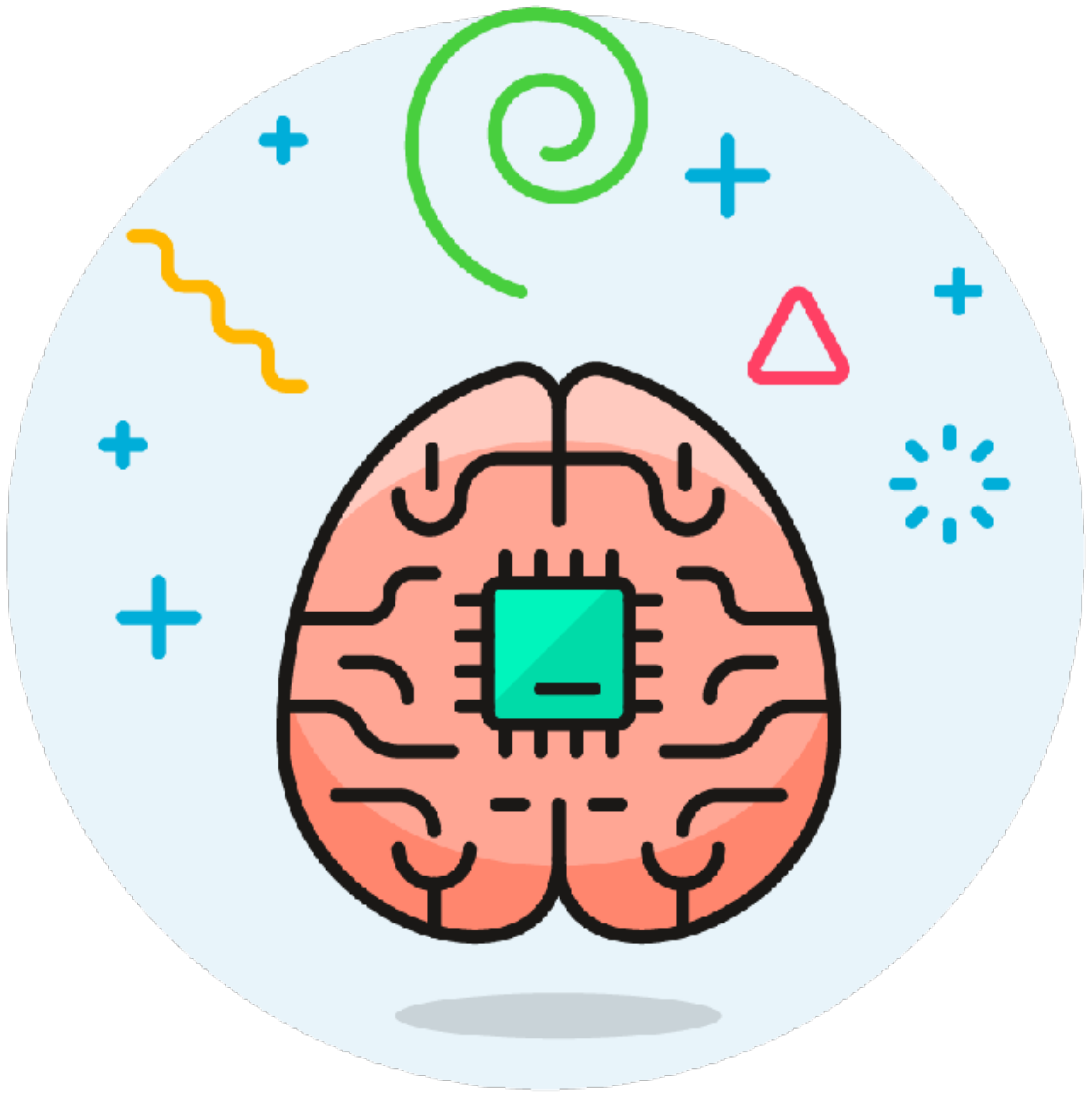


O'REILLY®

Mobile Game Development with Unity

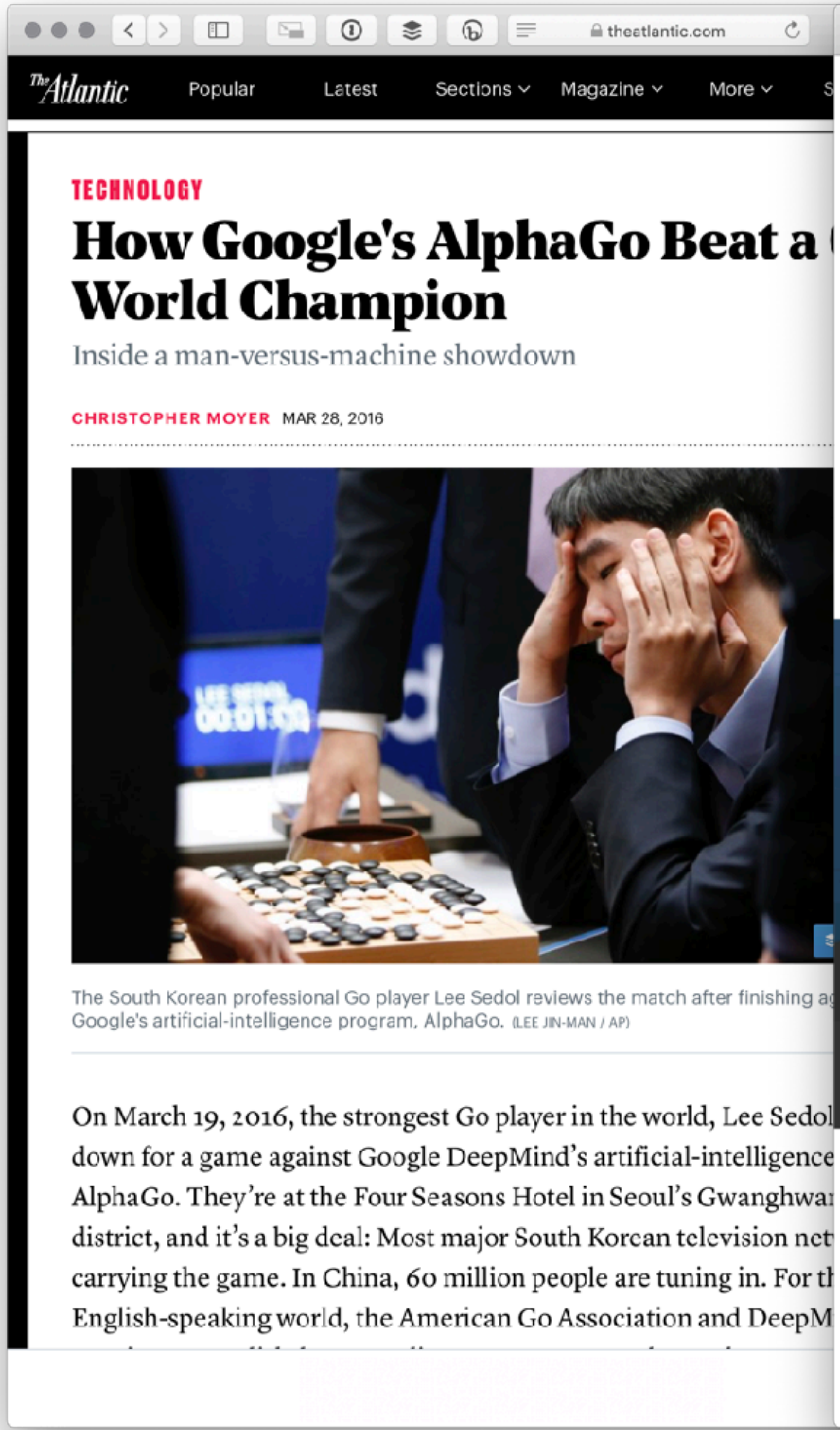
BUILD ONCE, DEPLOY ANYWHERE

Jon Manning & Paris Buttfield-Addison



Why a game engine?

A game engine is a controlled, self-contained spatial, physical environment that can (closely) replicate (enough of) the real world (to be useful).



TECHNOLOGY

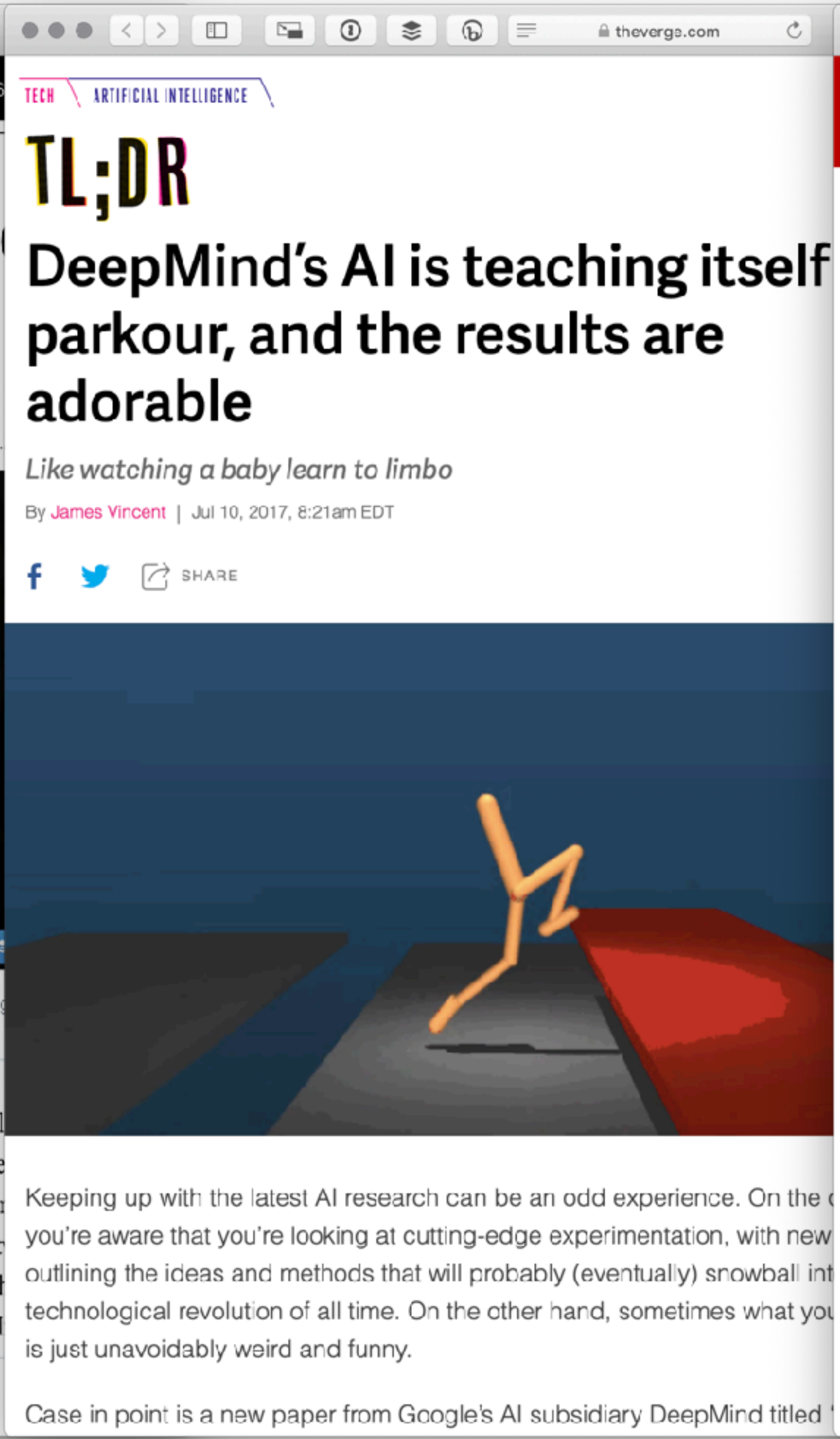
How Google's AlphaGo Beat a
World Champion

Inside a man-versus-machine showdown

CHRISTOPHER MOYER MAR 28, 2016

The South Korean professional Go player Lee Sedol reviews the match after finishing a
Google's artificial-intelligence program, AlphaGo. (LEE JIN-MAN / AP)

On March 19, 2016, the strongest Go player in the world, Lee Sedol
down for a game against Google DeepMind's artificial-intelligence
AlphaGo. They're at the Four Seasons Hotel in Seoul's Gwanghwa
district, and it's a big deal: Most major South Korean television net
carrying the game. In China, 60 million people are tuning in. For th
English-speaking world, the American Go Association and DeepM



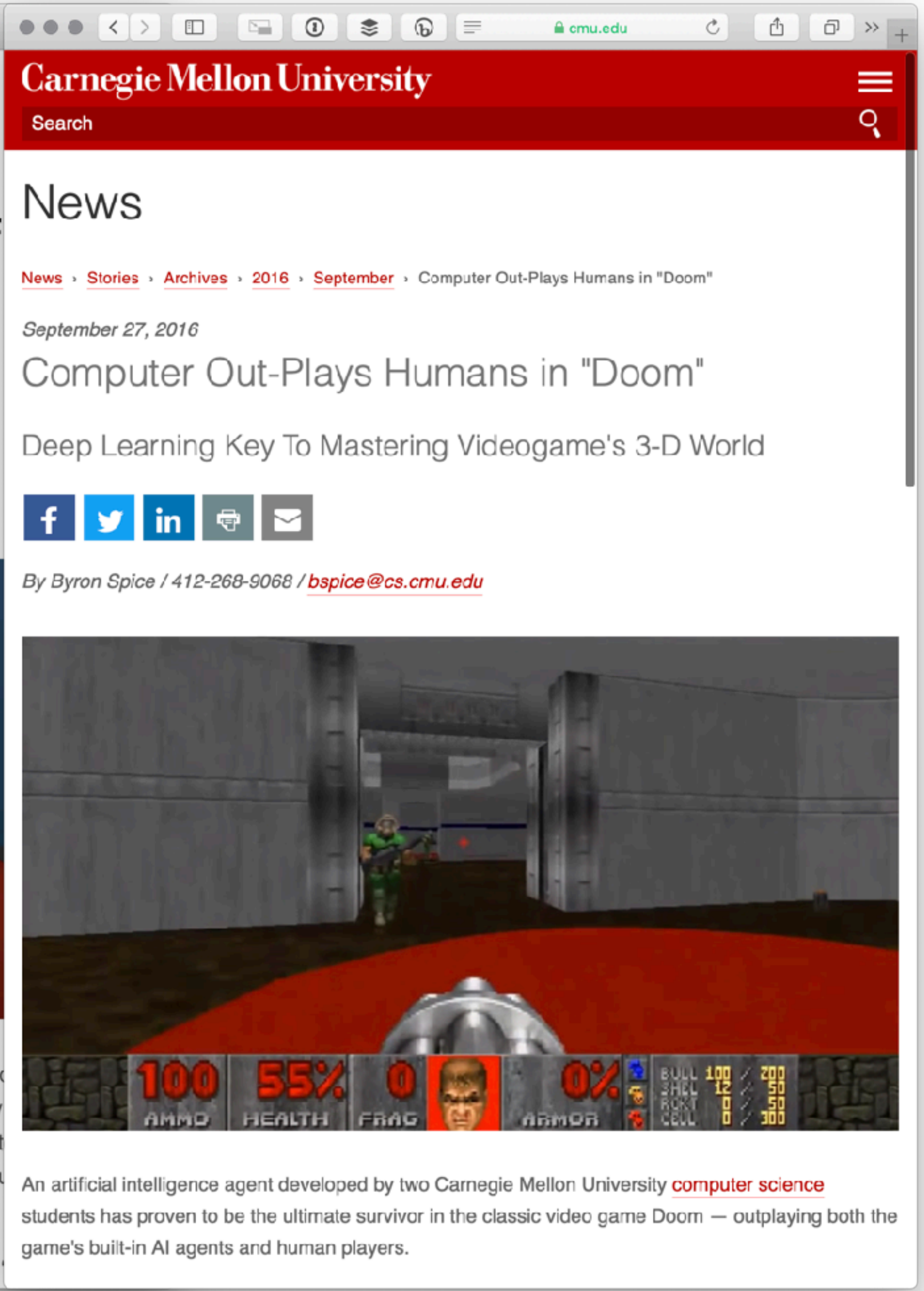
DeepMind's AI is teaching itself parkour, and the results are adorable

Like watching a baby learn to limbo

By James Vincent | Jul 10, 2017, 8:21am EDT

Keeping up with the latest AI research can be an odd experience. On the
you're aware that you're looking at cutting-edge experimentation, with new
outlining the ideas and methods that will probably (eventually) snowball int
technological revolution of all time. On the other hand, sometimes what you
is just unavoidably weird and funny.

Case in point is a new paper from Google's AI subsidiary DeepMind titled



News

News › Stories › Archives › 2016 › September › Computer Out-Plays Humans in "Doom"

September 27, 2016

Computer Out-Plays Humans in "Doom"

Deep Learning Key To Mastering Videogame's 3-D World

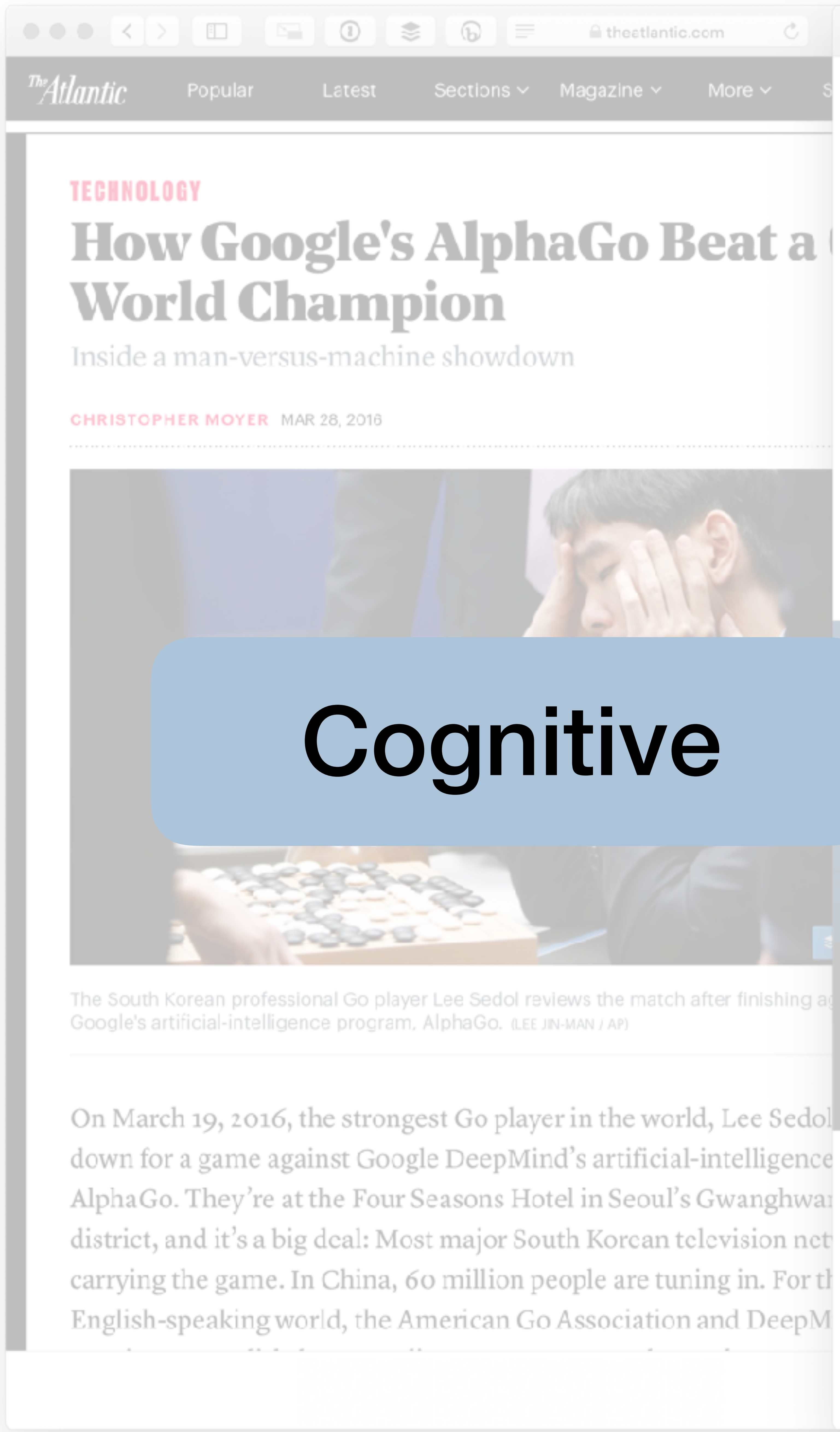
By Byron Spice / 412-268-9068 / bspice@cs.cmu.edu

An artificial intelligence agent developed by two Carnegie Mellon University computer science students has proven to be the ultimate survivor in the classic video game Doom — outplaying both the game's built-in AI agents and human players.

Cognitive

Physical

Visual


theatlantic.com

The Atlantic Popular Latest Sections Magazine More

TECHNOLOGY

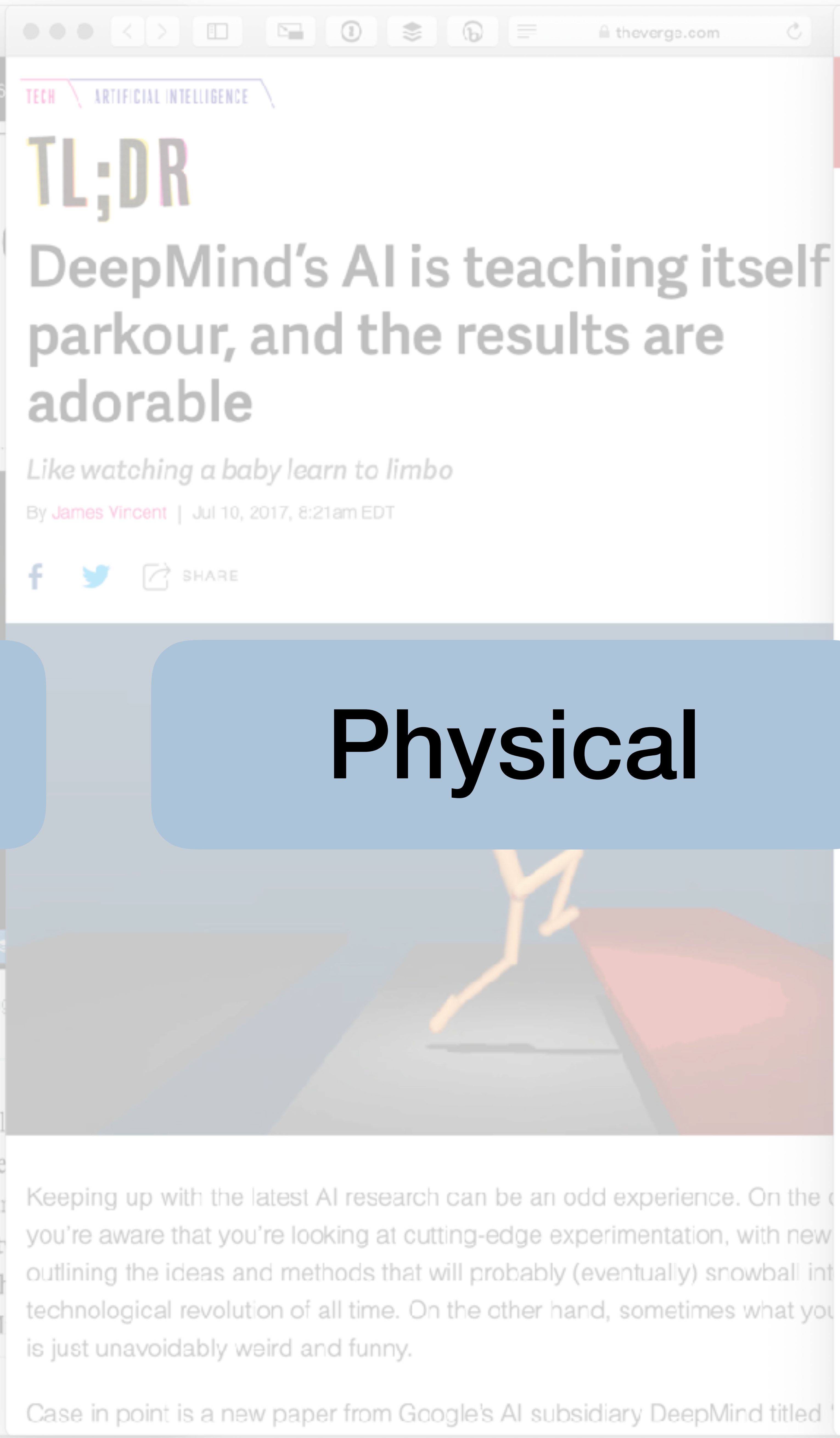
How Google's AlphaGo Beat a
World Champion

Inside a man-versus-machine showdown

CHRISTOPHER MOYER MAR 28, 2016

The South Korean professional Go player Lee Sedol reviews the match after finishing a
Google's artificial-intelligence program, AlphaGo. (LEE JIN-MAN / AP)

On March 19, 2016, the strongest Go player in the world, Lee Sedol
down for a game against Google DeepMind's artificial-intelligence
AlphaGo. They're at the Four Seasons Hotel in Seoul's Gwanghwa
district, and it's a big deal: Most major South Korean television net
carrying the game. In China, 60 million people are tuning in. For th
English-speaking world, the American Go Association and DeepM


theverge.com

TECH \ ARTIFICIAL INTELLIGENCE

TL;DR

DeepMind's AI is teaching itself
parkour, and the results are
adorable

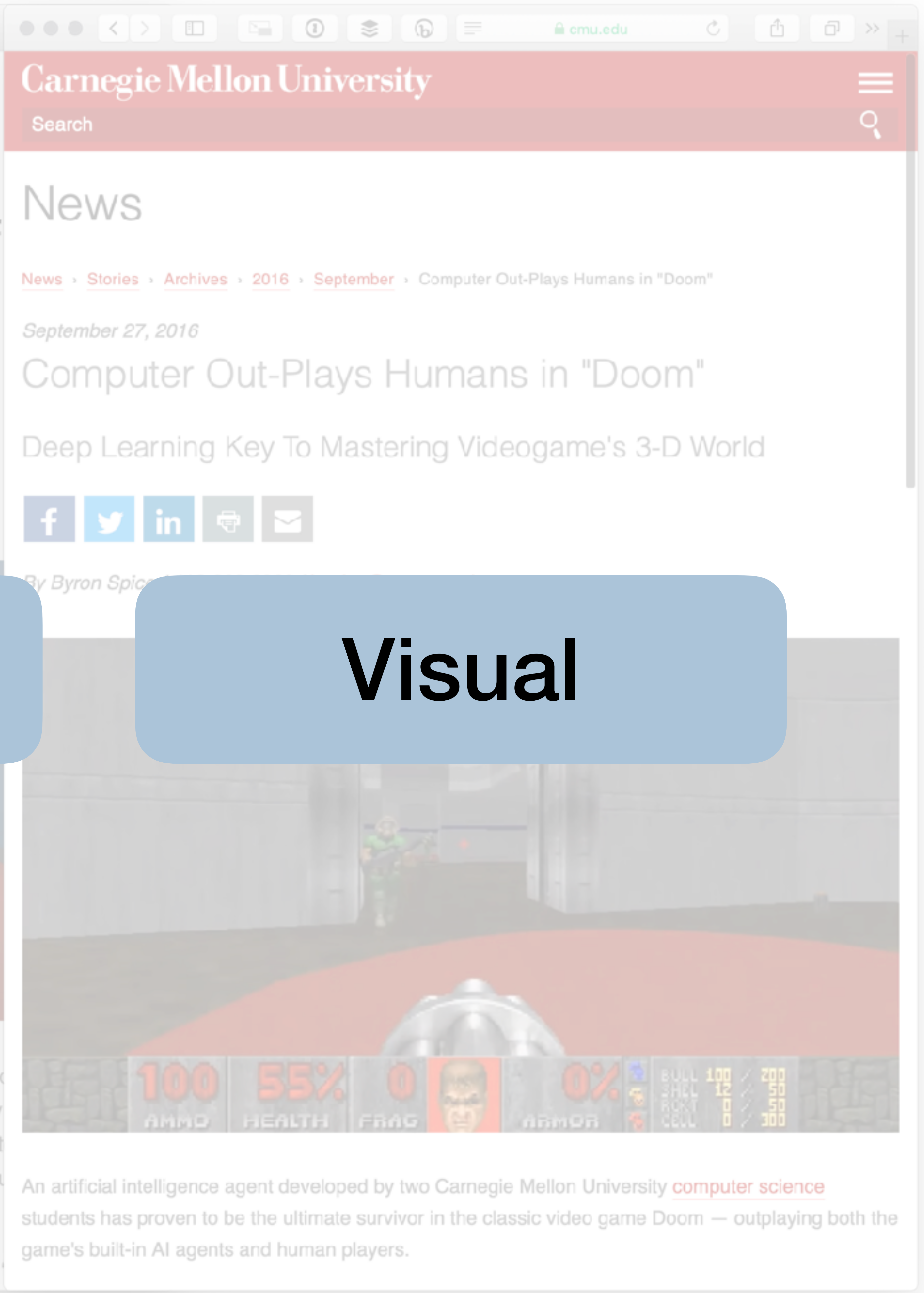
Like watching a baby learn to limbo

By James Vincent | Jul 10, 2017, 8:21am EDT

SHARE

Keeping up with the latest AI research can be an odd experience. On the c
you're aware that you're looking at cutting-edge experimentation, with new
outlining the ideas and methods that will probably (eventually) snowball int
technological revolution of all time. On the other hand, sometimes what yo
is just unavoidably weird and funny.

Case in point is a new paper from Google's AI subsidiary DeepMind titled


cmu.edu

Carnegie Mellon University

Search

News

News › Stories › Archives › 2016 › September › Computer Out-Plays Humans in "Doom"

September 27, 2016

Computer Out-Plays Humans in "Doom"

Deep Learning Key To Mastering Videogame's 3-D World

By Byron Spic

An artificial intelligence agent developed by two Carnegie Mellon University computer science
students has proven to be the ultimate survivor in the classic video game Doom — outplaying both the
game's built-in AI agents and human players.

Unity: A General Platform for Intelligent Agents

Arthur Juliani
Unity Technologies
arthurj@unity3d.com

Vincent-Pierre Berges
Unity Technologies
vincentpierre@unity3d.com

Esh Vckay
Unity Technologies
esh@unity3d.com

Yuan Gao
Unity Technologies
vincentg@unity3d.com

Hunter Henry
Unity Technologies
brandonh@unity3d.com

Marwan Mattar
Unity Technologies
marwan@unity3d.com

Danny Lange
Unity Technologies
dlange@unity3d.com

Abstract

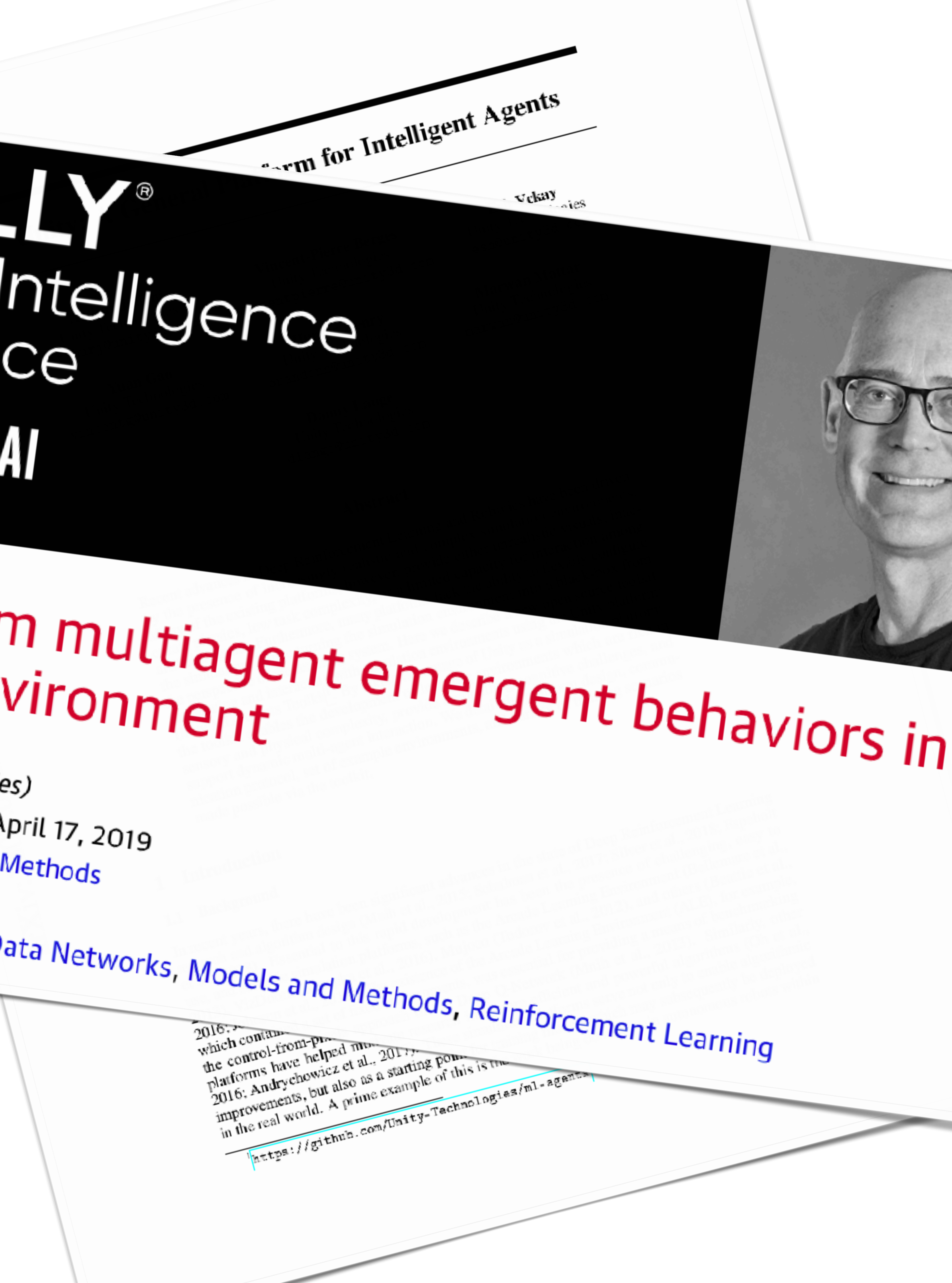
Recent advances in Deep Reinforcement Learning and Robotics have been driven by the presence of increasingly realistic and complex simulation environments. Many of the existing platforms, however, provide either unrealistic visuals, inaccurate physics, low task complexity, or a limited capacity for interaction among artificial agents. Furthermore, many platforms lack the ability to flexibly configure the simulation, hence turning the simulation environment into a black-box from the perspective of the learning system. Here we describe a new open source toolkit for creating and interacting with simulation environments using the Unity platform: Unity ML-Agents Toolkit[1]. By taking advantage of Unity as a simulation platform, the toolkit enables the development of learning environments which are rich in sensory and physical complexity, provide compelling cognitive challenges, and support dynamic multi-agent interaction. We detail the platform design, communication protocol, set of example environments, and variety of training scenarios made possible via the toolkit.

1 Introduction

1.1 Background

In recent years, there have been significant advances in the state of Deep Reinforcement Learning research and algorithm design (Mnih et al., 2015; Schulman et al., 2017; Silver et al., 2018; Espeholt et al., 2018). Essential to this rapid development has been the presence of challenging, easy to use, and scalable simulation platforms, such as the Arcade Learning Environment (Bellemare et al., 2013), VizDoom (Kempka et al., 2016), Mujoco (Todorov et al., 2012), and others (Beattie et al., 2016; Johnson et al., 2016). The existence of the Arcade Learning Environment (ALE), for example, which contained a set of fixed environments, was essential for providing a means of benchmarking the control-from-pixels approach of the Deep Q-Network (Mnih et al., 2013). Similarly, other platforms have helped motivate research into more efficient and powerful algorithms (Oh et al., 2016; Andrychowicz et al., 2017). These simulation platforms serve not only to enable algorithmic improvements, but also as a starting point for training models which may subsequently be deployed in the real world. A prime example of this is the work being done to train autonomous robots within

[1]https://github.com/Unity-Technologies/ml-agents



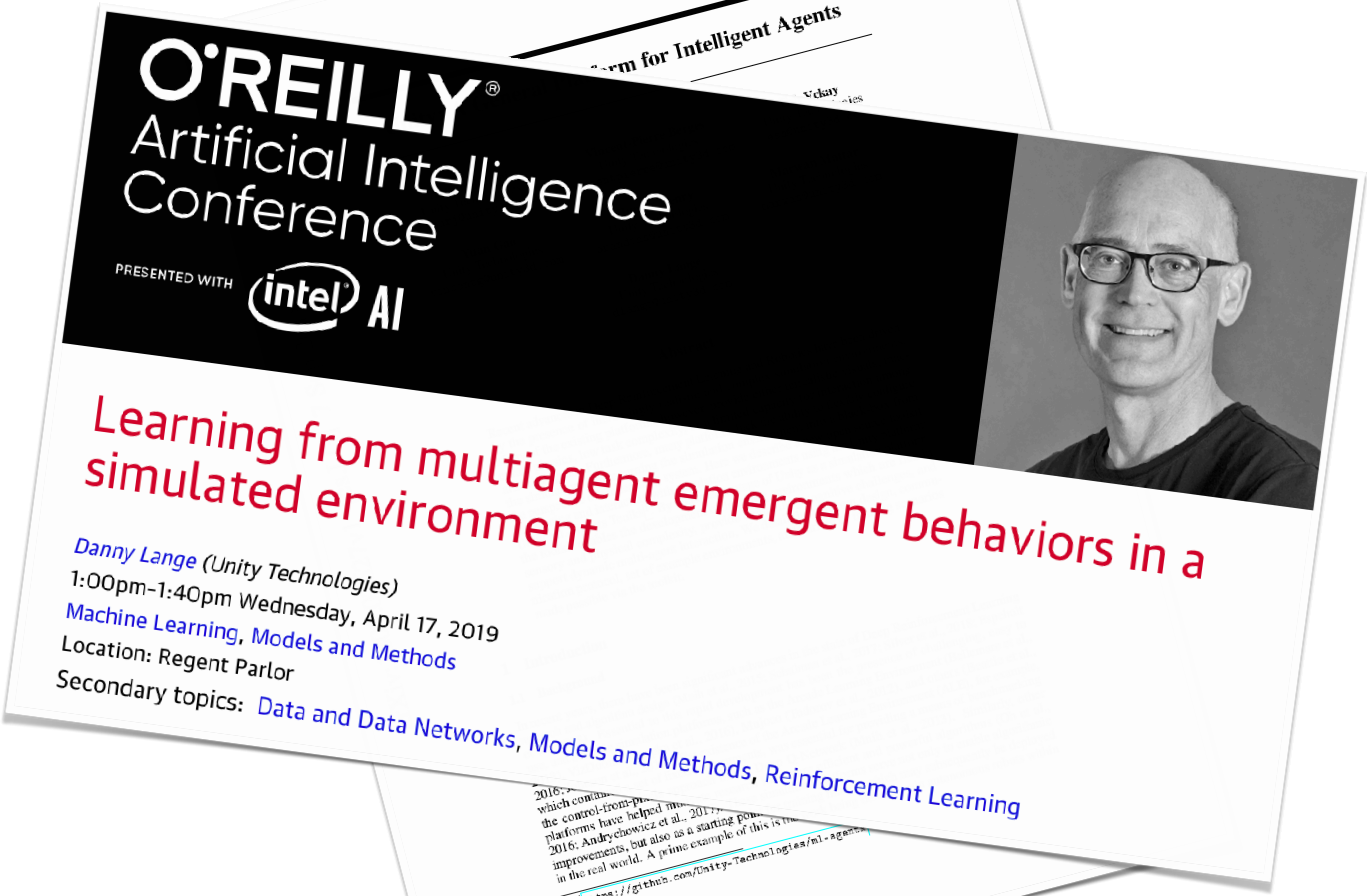
orm for Intelligent Agents

Vekay

ies

2016;
which conta
the control-from-p
platforms have helped m
2016; Andrychowicz et al., 2017
improvements, but also as a starting po
in the real world. A prime example of this is th

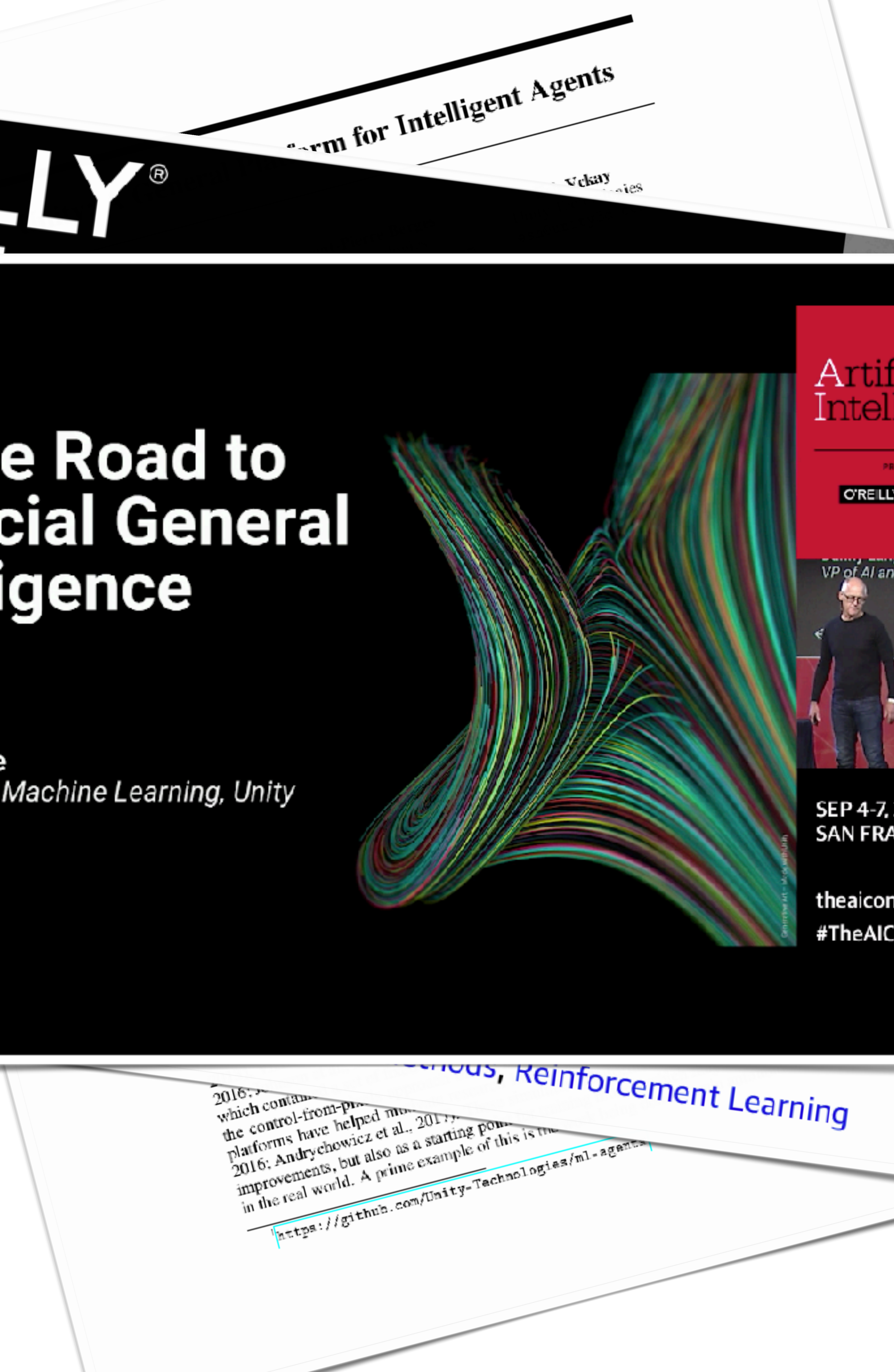
https://github.com/Unity-Technologies/ml-agents



O'REILLY®
Artificial Intelligence Conference
PRESENTED WITH intel AI

Learning from multiagent emergent behaviors in a simulated environment

Danny Lange (Unity Technologies)
1:00pm–1:40pm Wednesday, April 17, 2019
Machine Learning, Models and Methods
Location: Regent Parlor
Secondary topics: Data and Data Networks, Models and Methods, Reinforcement Learning

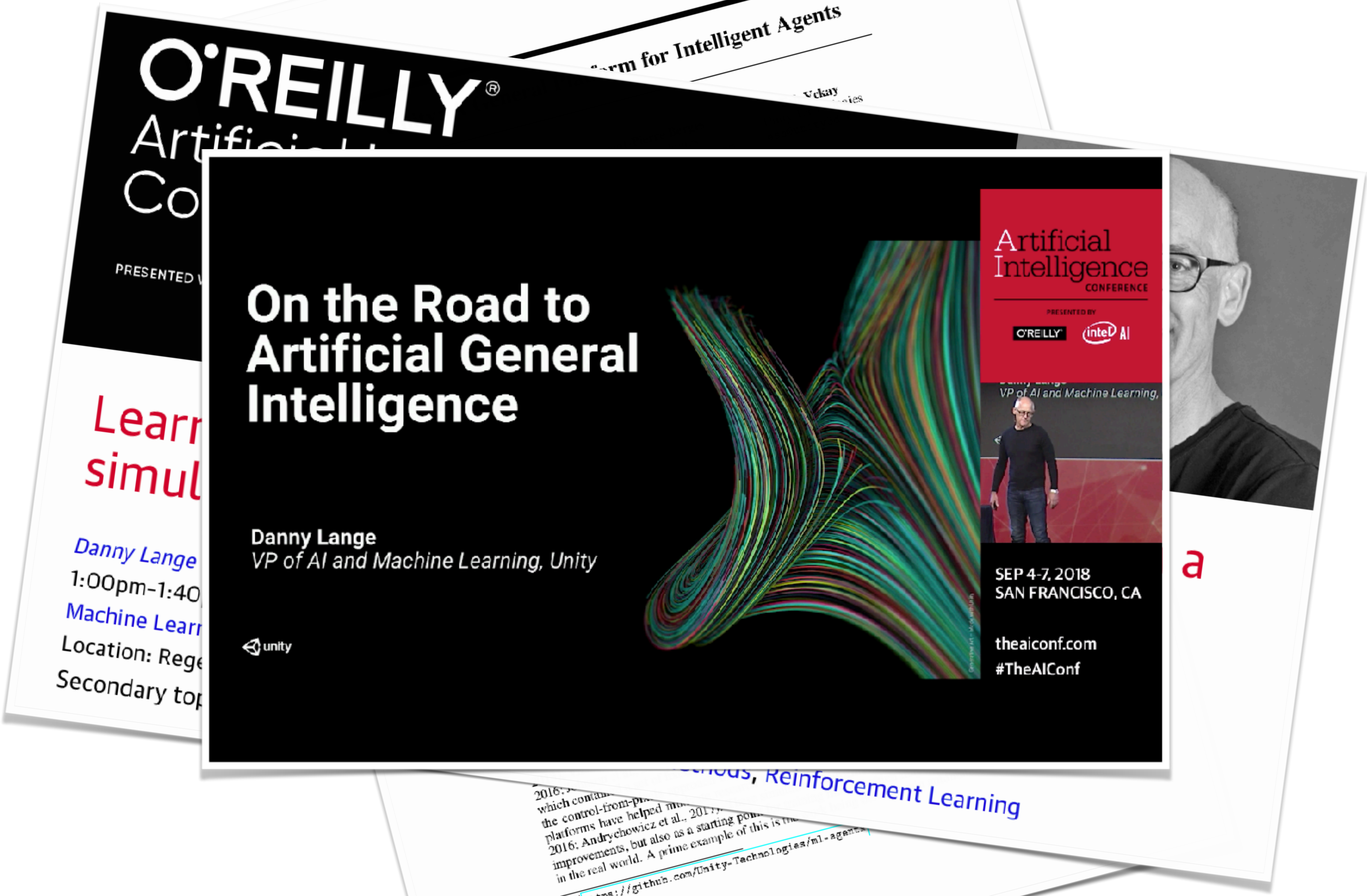


…rm for Intelligent Agents

…Vckay
…ies

2016;…
which contain…
the control-from-p…
platforms have helped m…
2016; Andrychowicz et al., 2017;…
improvements, but also as a starting poi…
in the real world. A prime example of this is th…

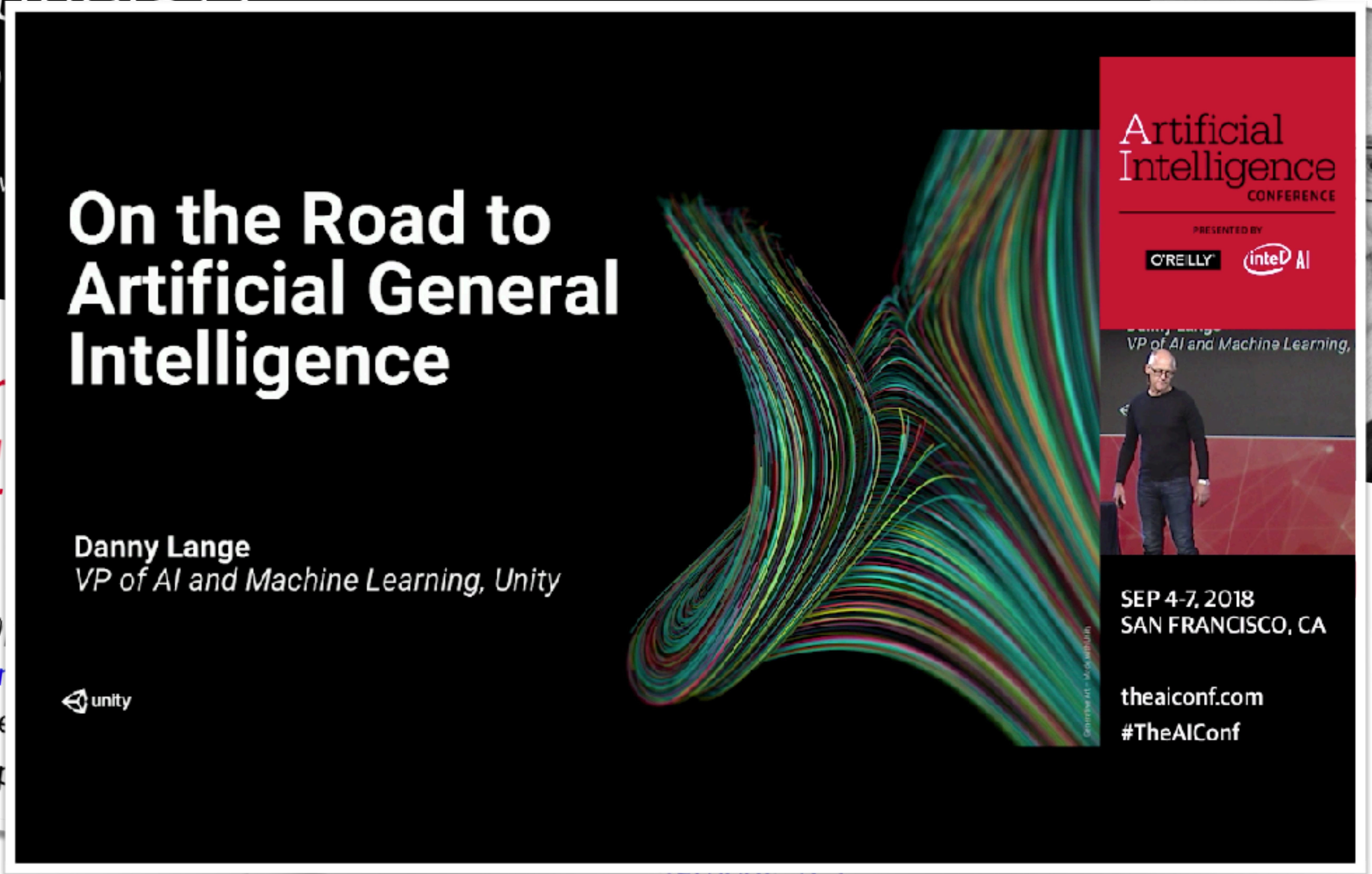
https://github.com/Unity-Technologies/ml-agents



O'REILLY®
Artifici…
Co…
PRESENTED …

Learn…
simul…
…a

Danny Lange
1:00pm–1:40…
Machine Learn…
Location: Rege…
Secondary top…
…ethods, Reinforcement Learning



On the Road to Artificial General Intelligence

Danny Lange
VP of AI and Machine Learning, Unity

unity

Artificial Intelligence CONFERENCE
PRESENTED BY
O'REILLY intel AI

VP of AI and Machine Learning,

SEP 4-7, 2018
SAN FRANCISCO, CA

theaiconf.com
#TheAIConf

Basics of Unity

ML-Agents Fundamentals

The Process

Live Demo

So What?

Basics of Unity

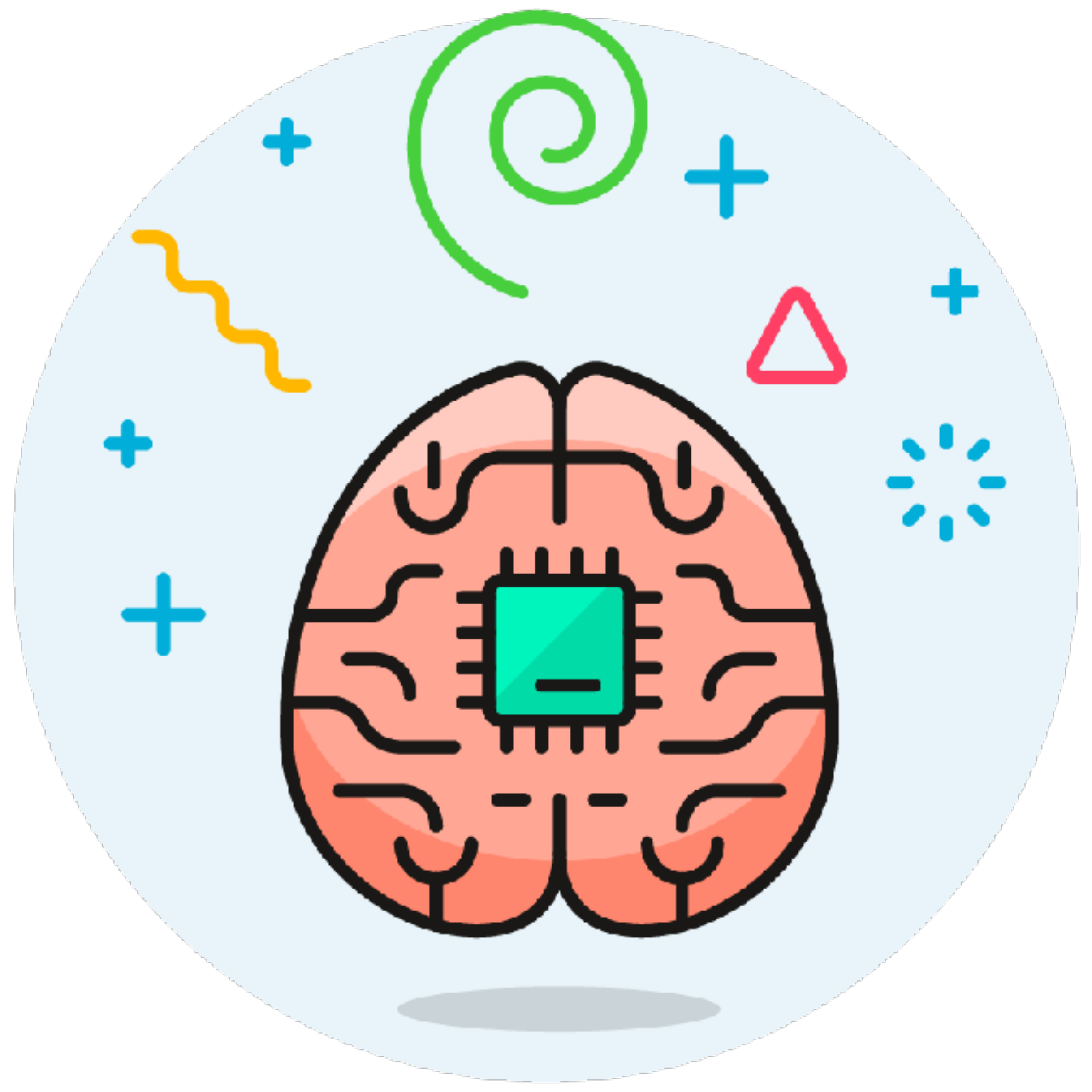
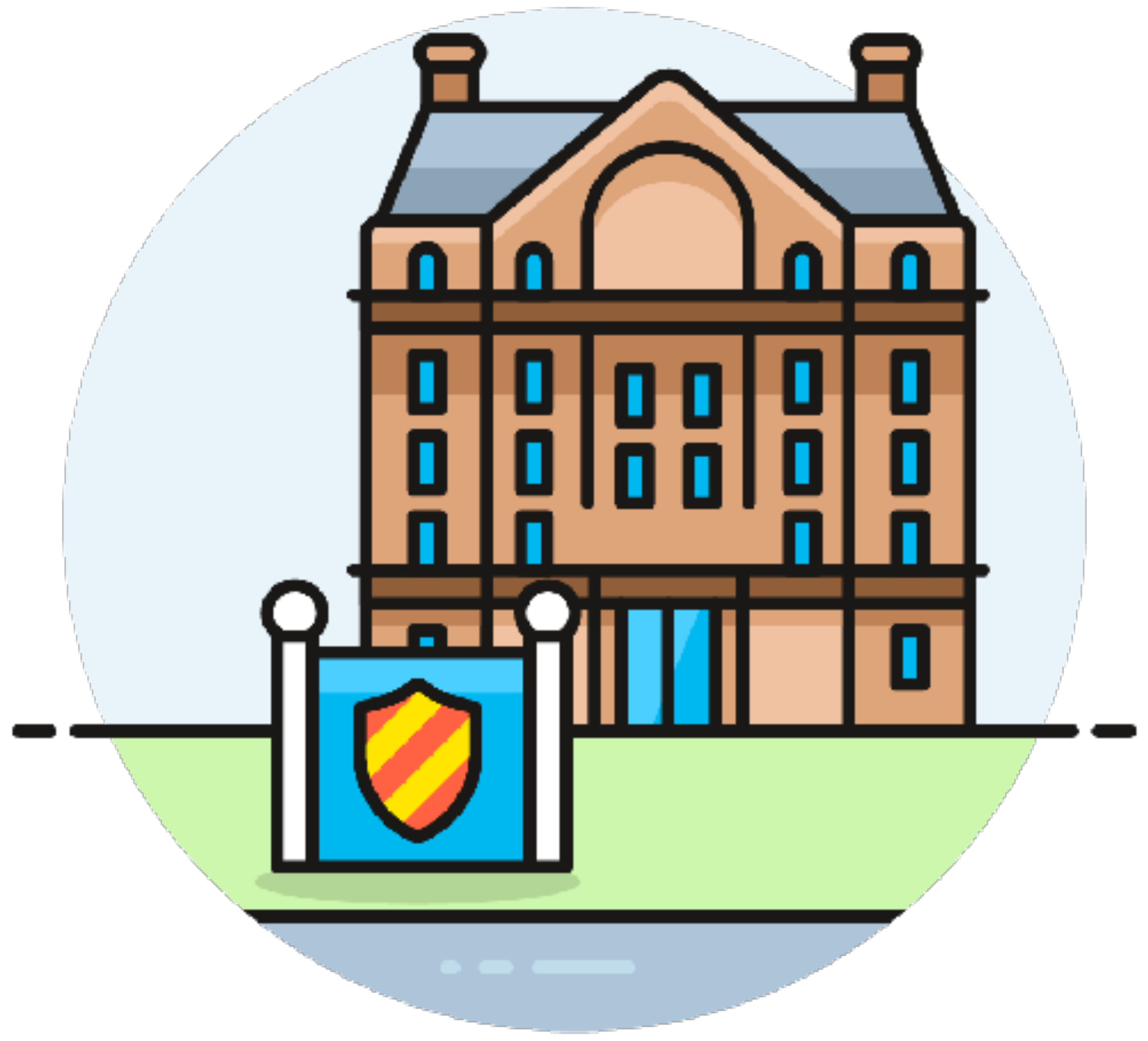
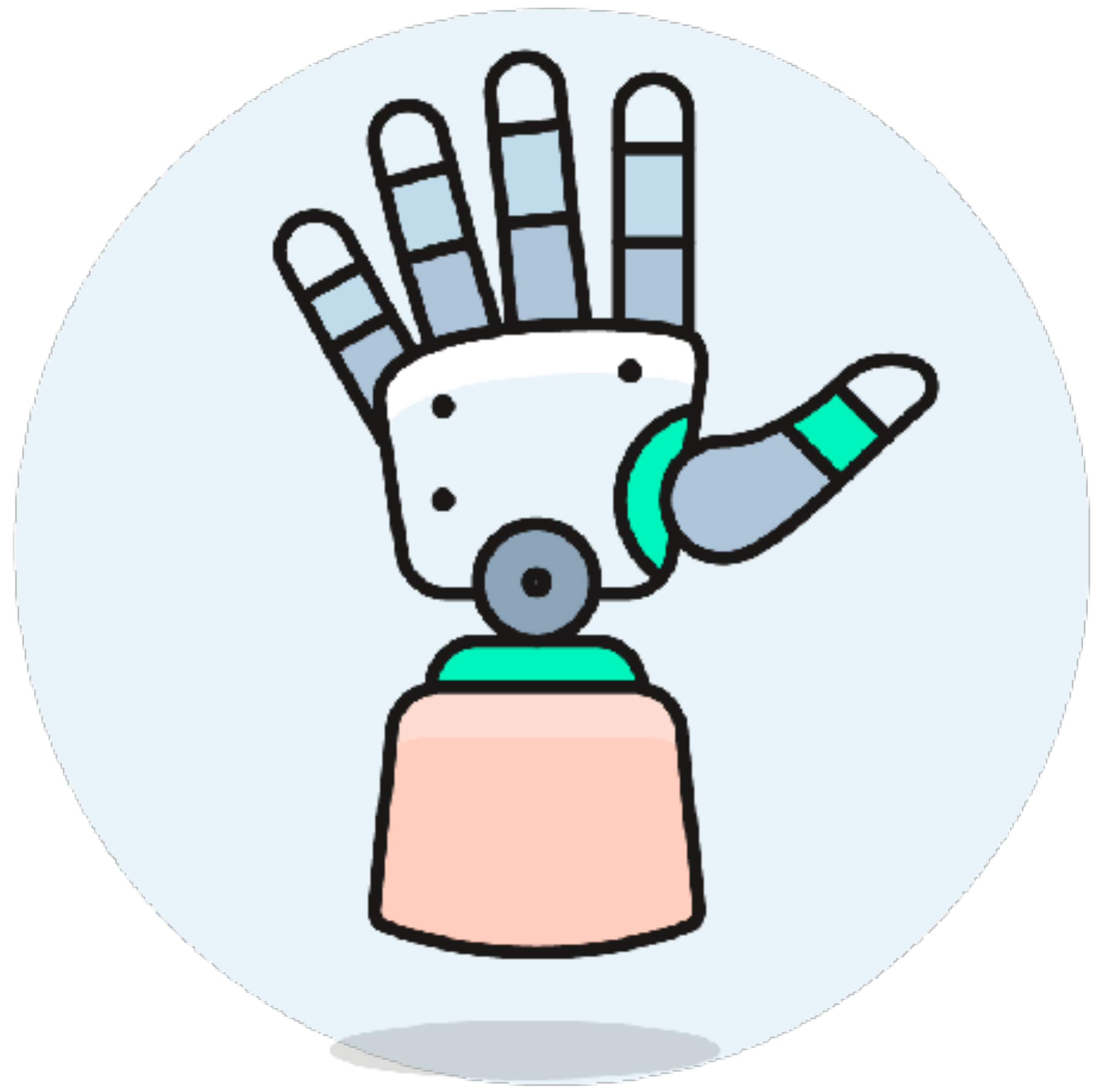
Live Demo

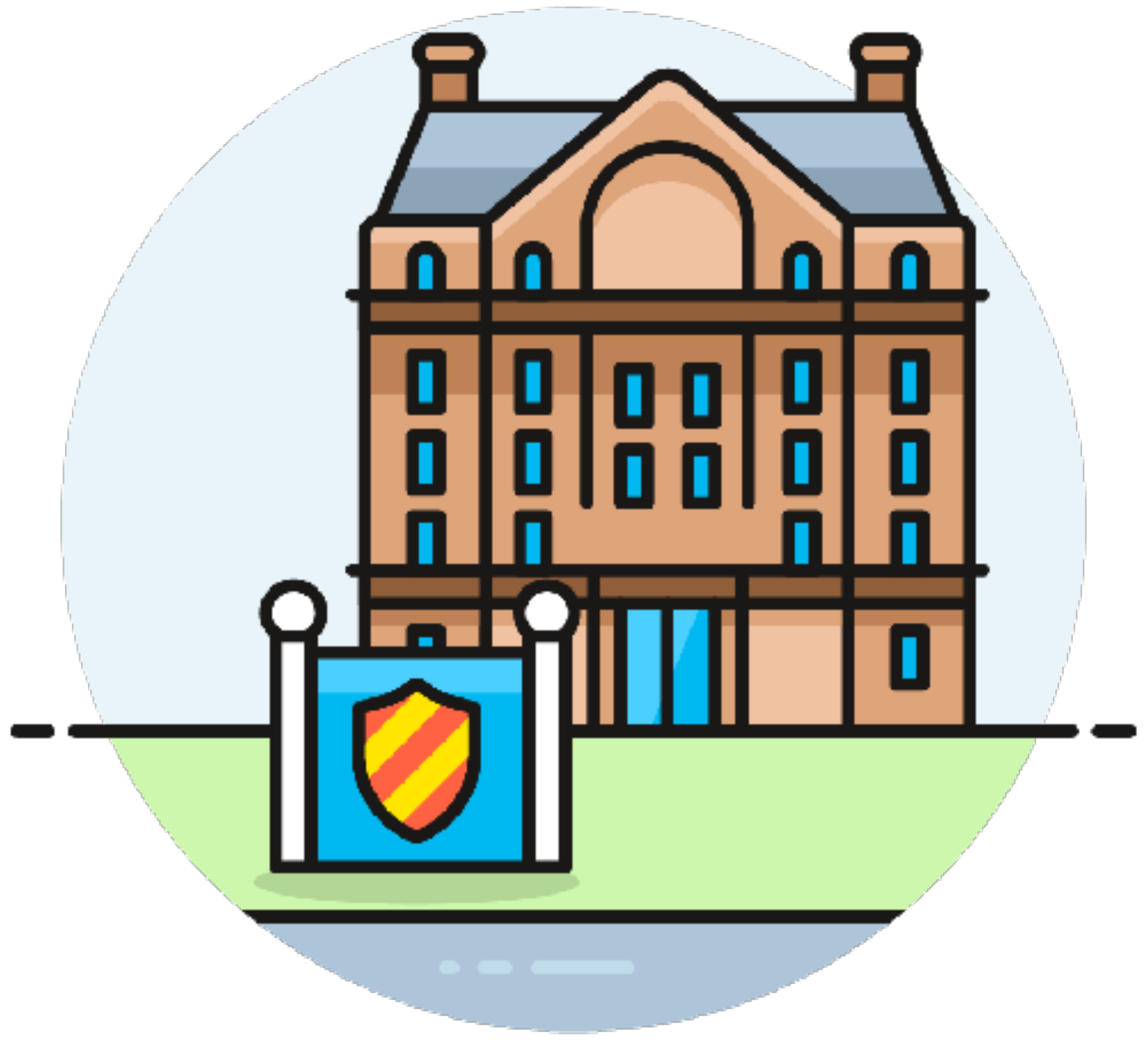
ML-Agents Fundamentals

https://github.com/Unity-Technologies/ml-agents/

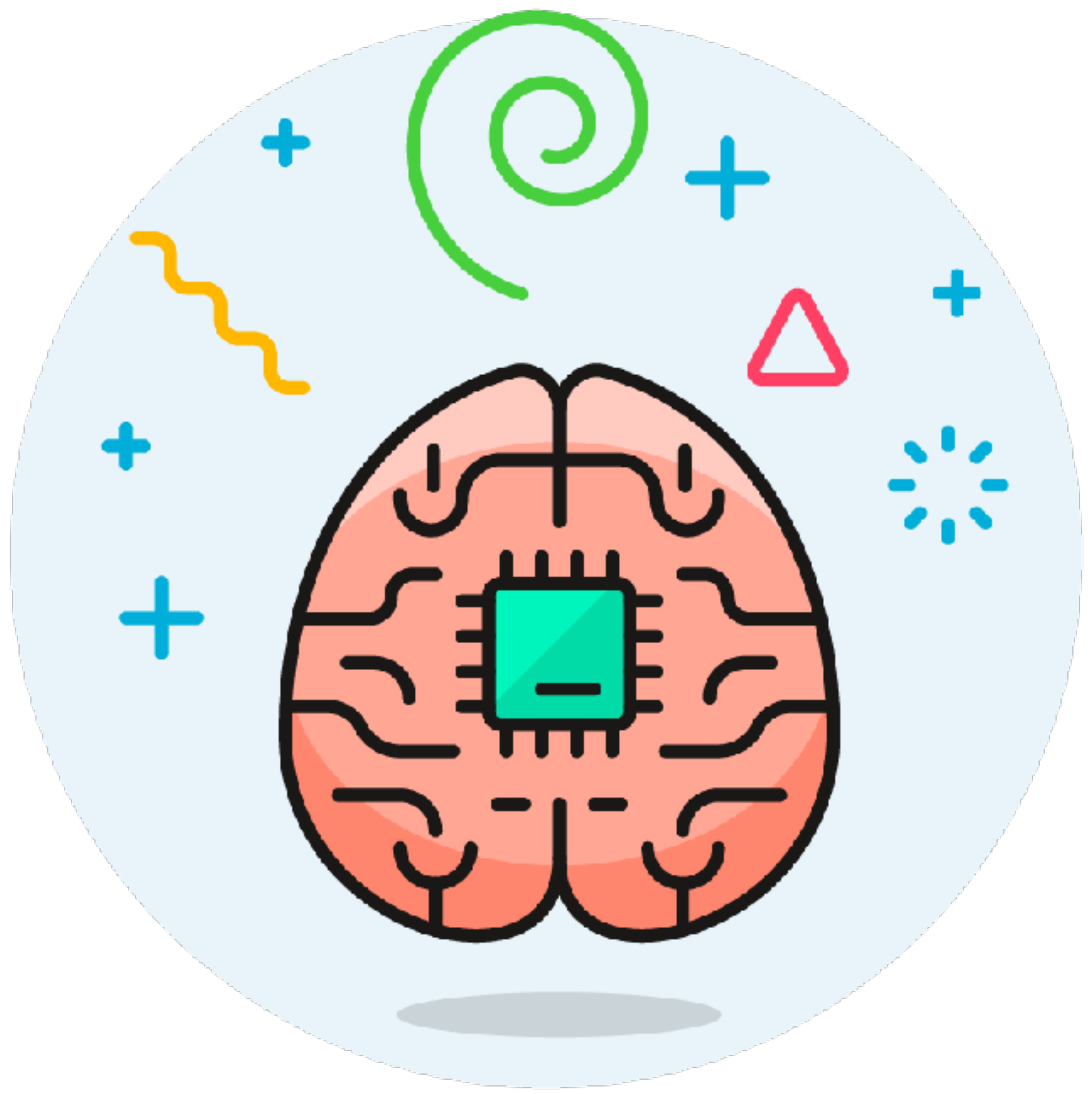
"The ML-Agents toolkit is mutually beneficial for both game developers and AI researchers as it provides a central platform where advances in AI can be evaluated on Unity's rich environments and then made accessible to the wider research and game developer communities."

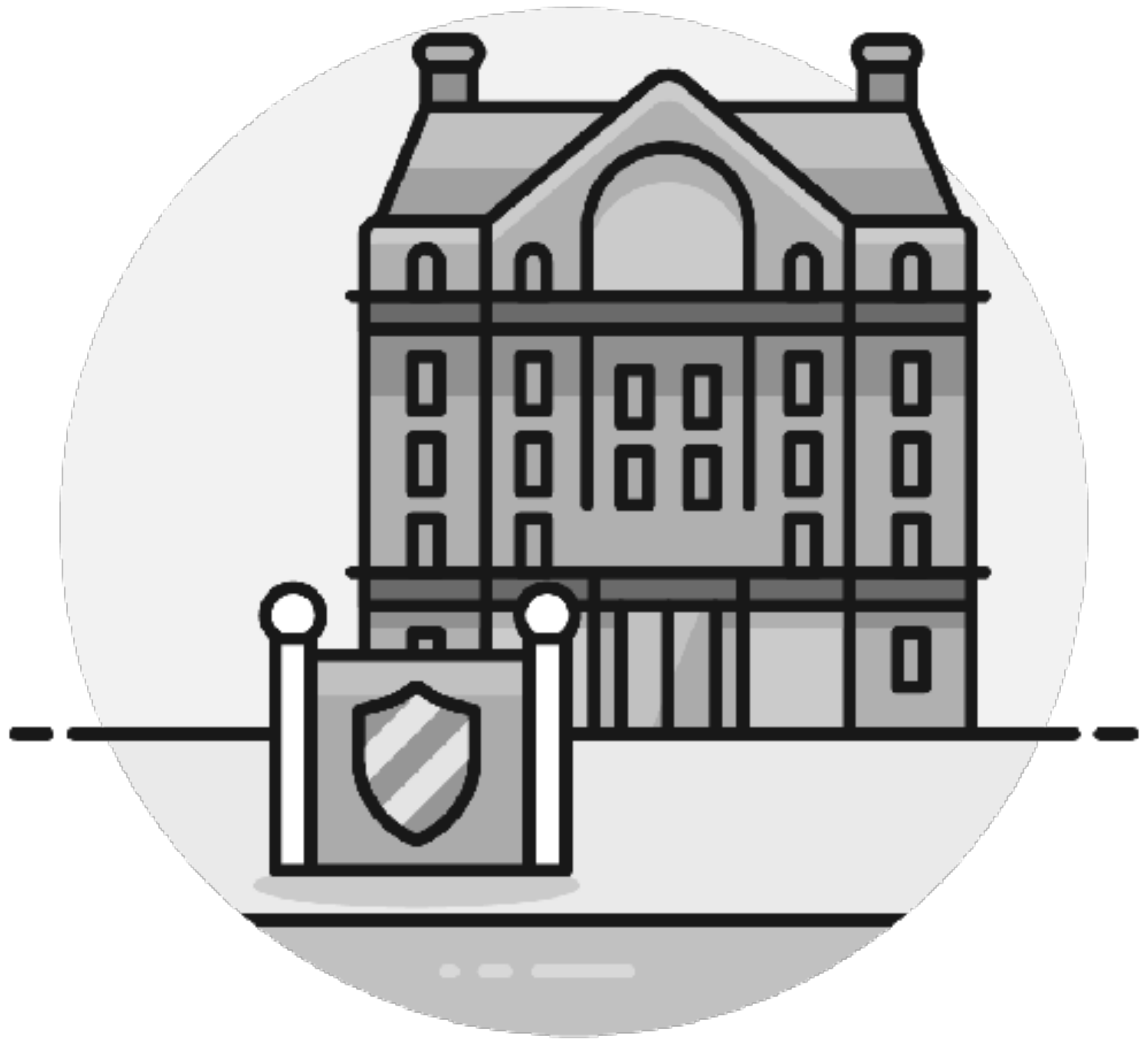
–Unity ML-Agents Toolkit Overview

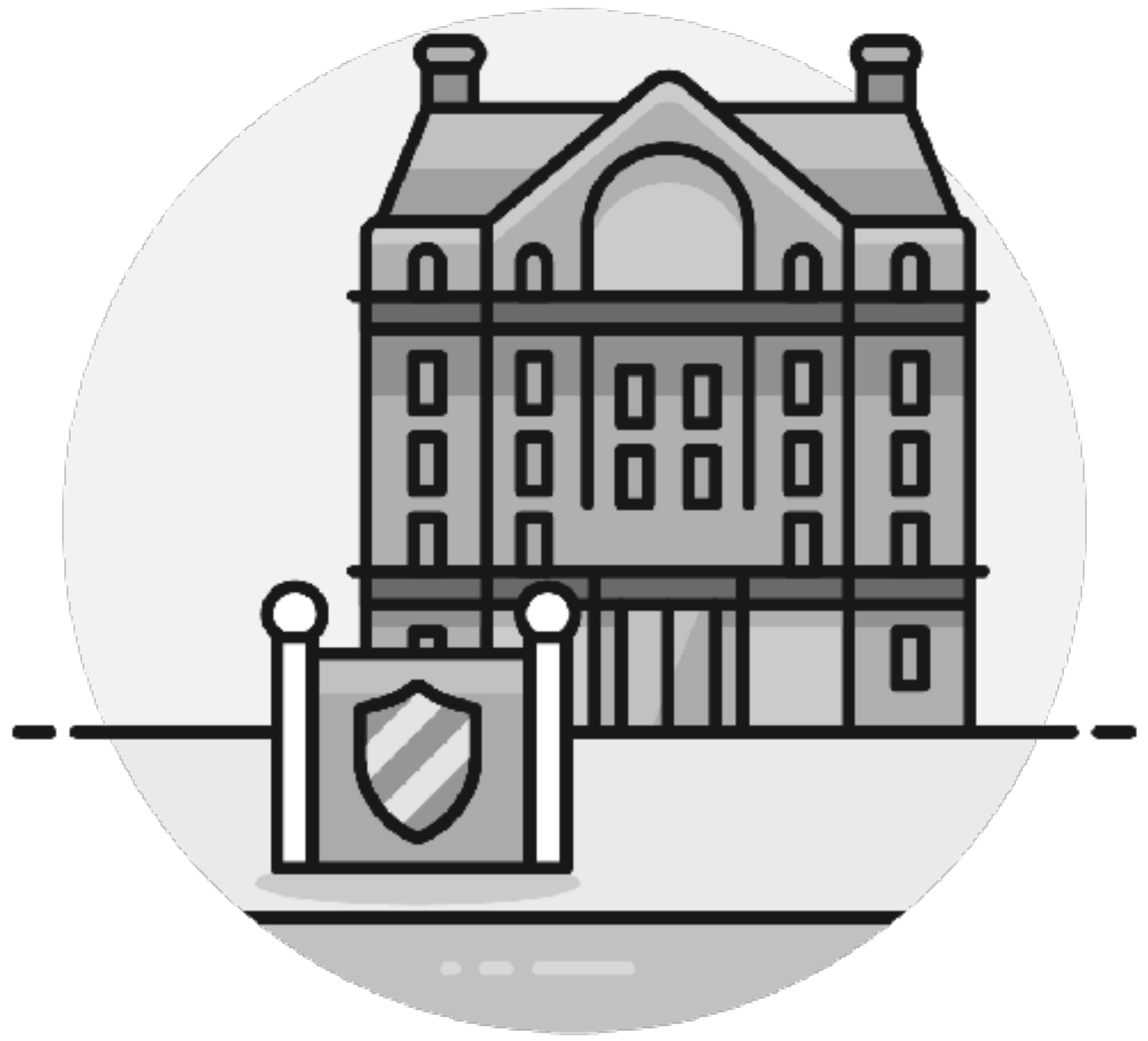


Academy

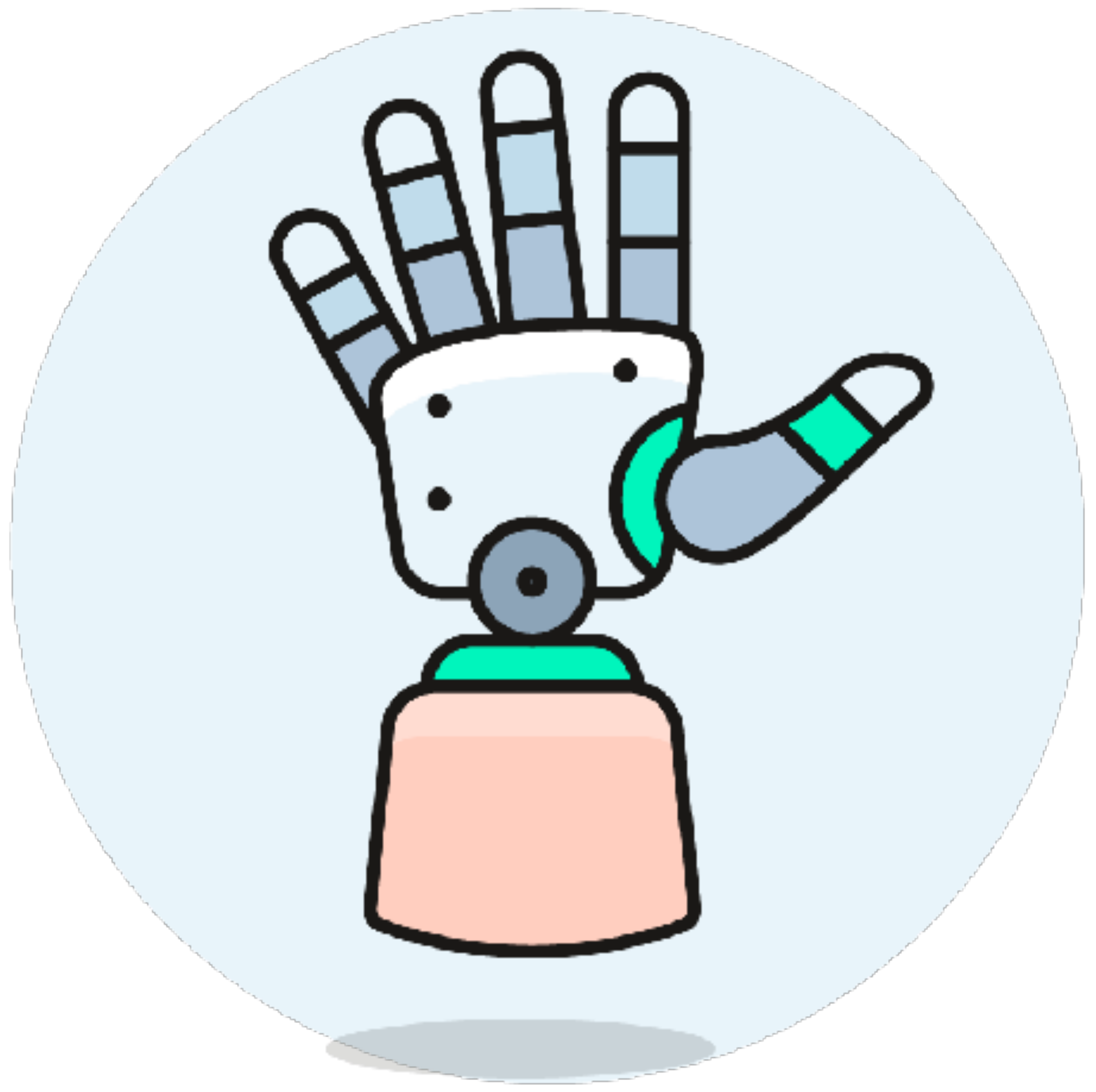

Brain


Academy

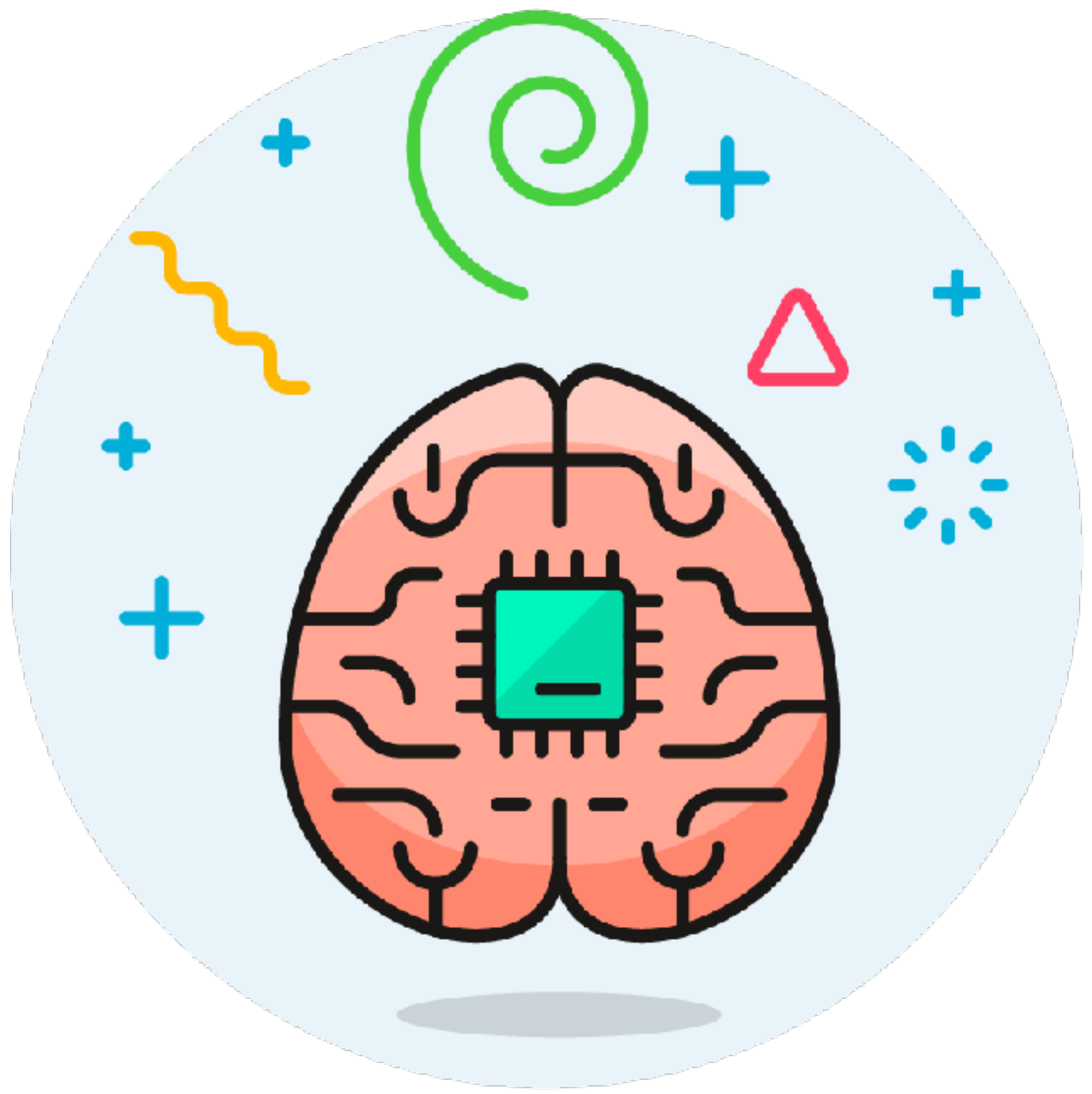
Brain

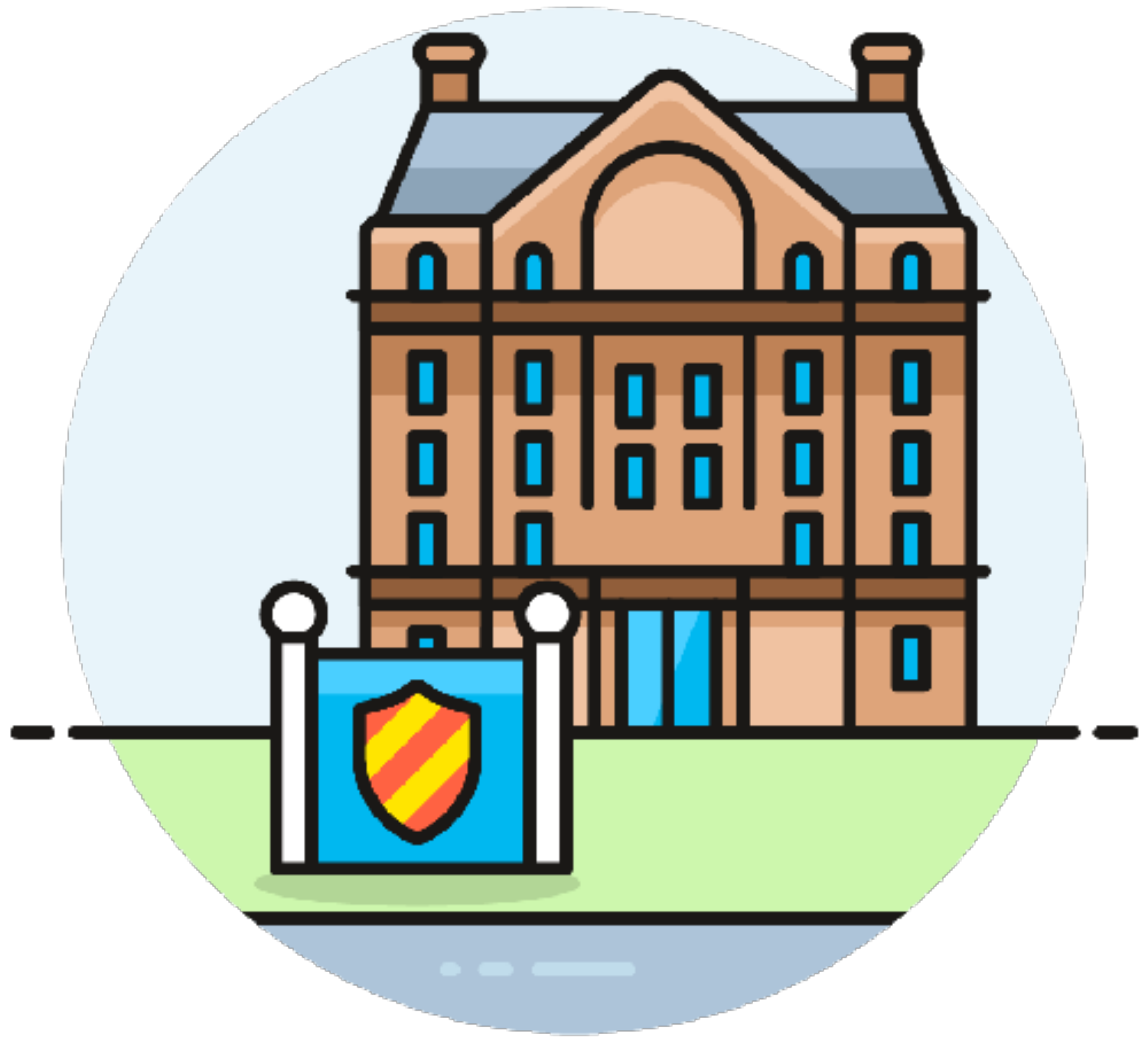


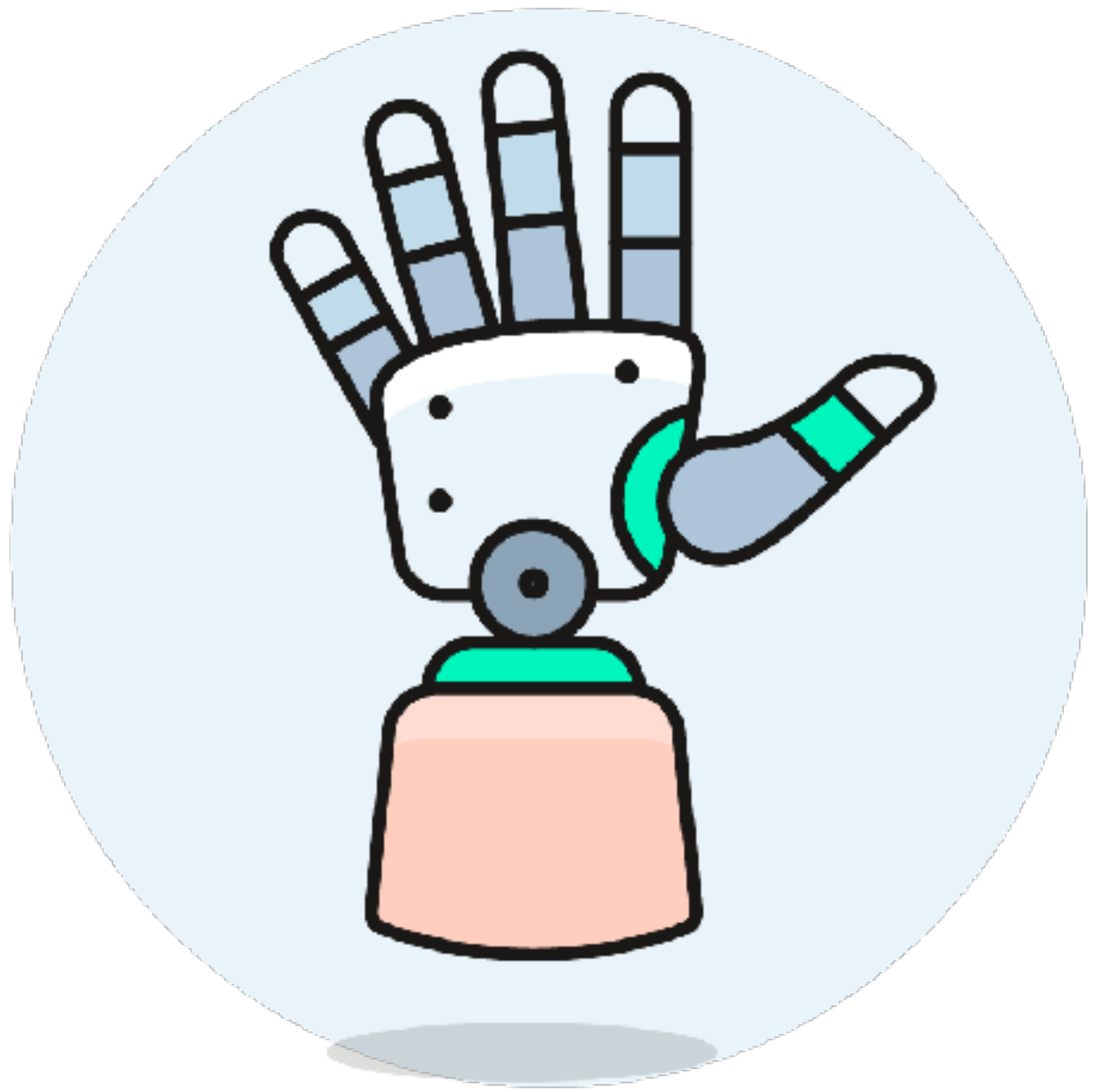
Academy

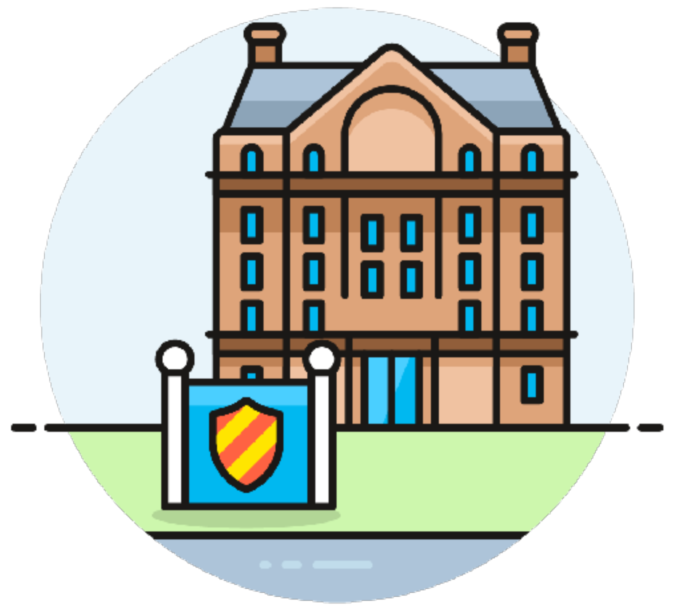


Agent


Brain

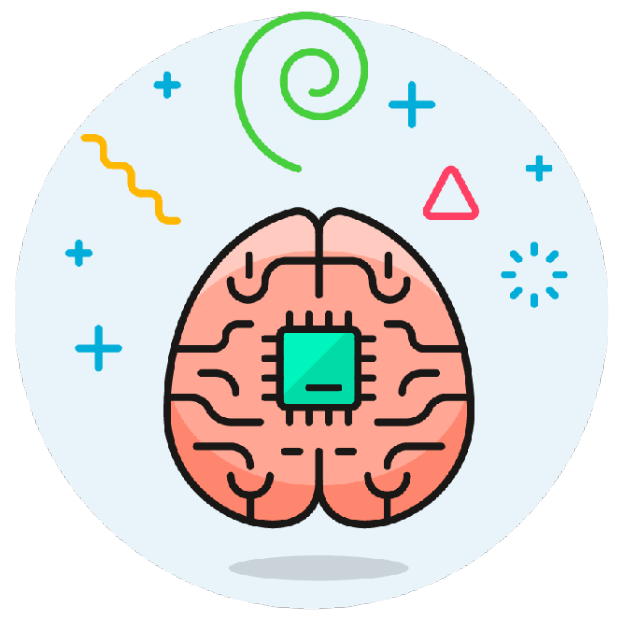

Academy


Agent



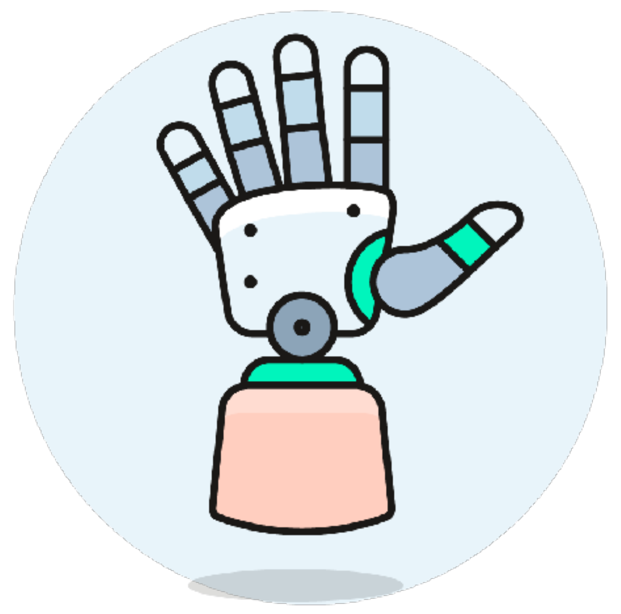
Academy

- Orchestrates the observations and decision making process
- Sets environment-wide parameters, like speed and rendering quality
- Talks to the external communicator
- Make sure agent(s) and brain(s) are in sync
- Coordinates everything



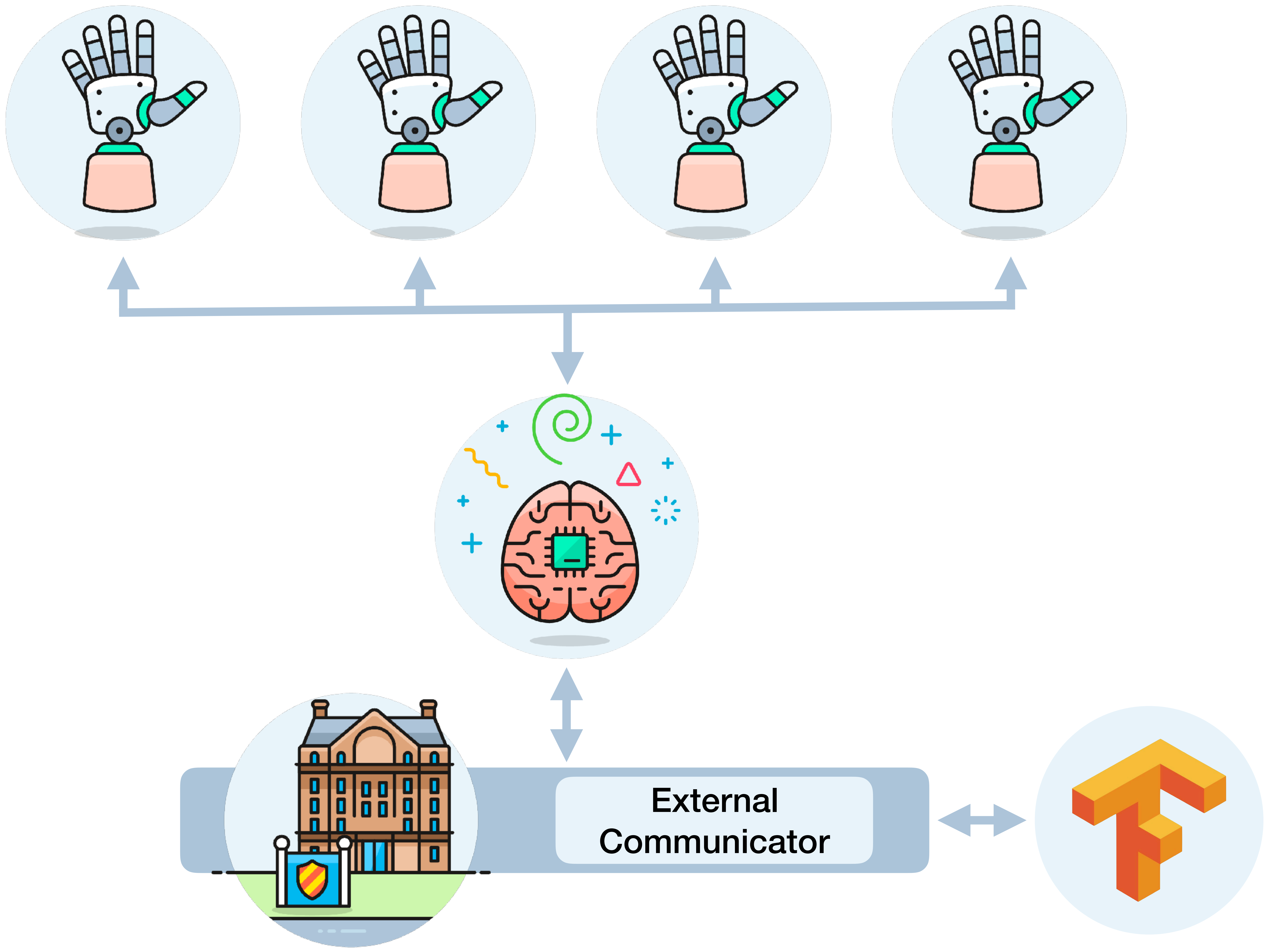
Brain

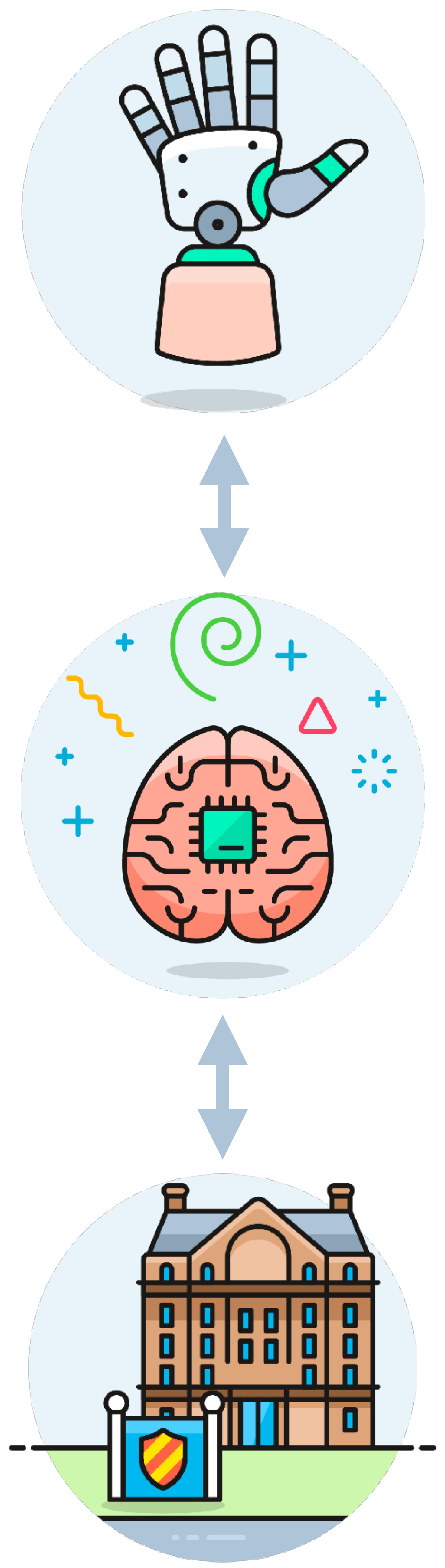
- Holds logic for the Agent’s decision making
- Determines which action(s) the Agent should take at each instance
- Receives observations from the Agent
- Receives rewards from the Agent
- Returns actions to the Agent
- Can be controlled by a human, a training process, or an inference process



Agent

- Attached to a Unity Game Object
- Generates observations
- Performs actions (that it’s told to do by a brain)
- Assigns rewards
- Linked to one Brain


External Communicator



None of these concepts are new

Some might have new names

Training Methods

Reinforcement Learning

Imitation Learning

Neuroevolution

… and many other learning methods

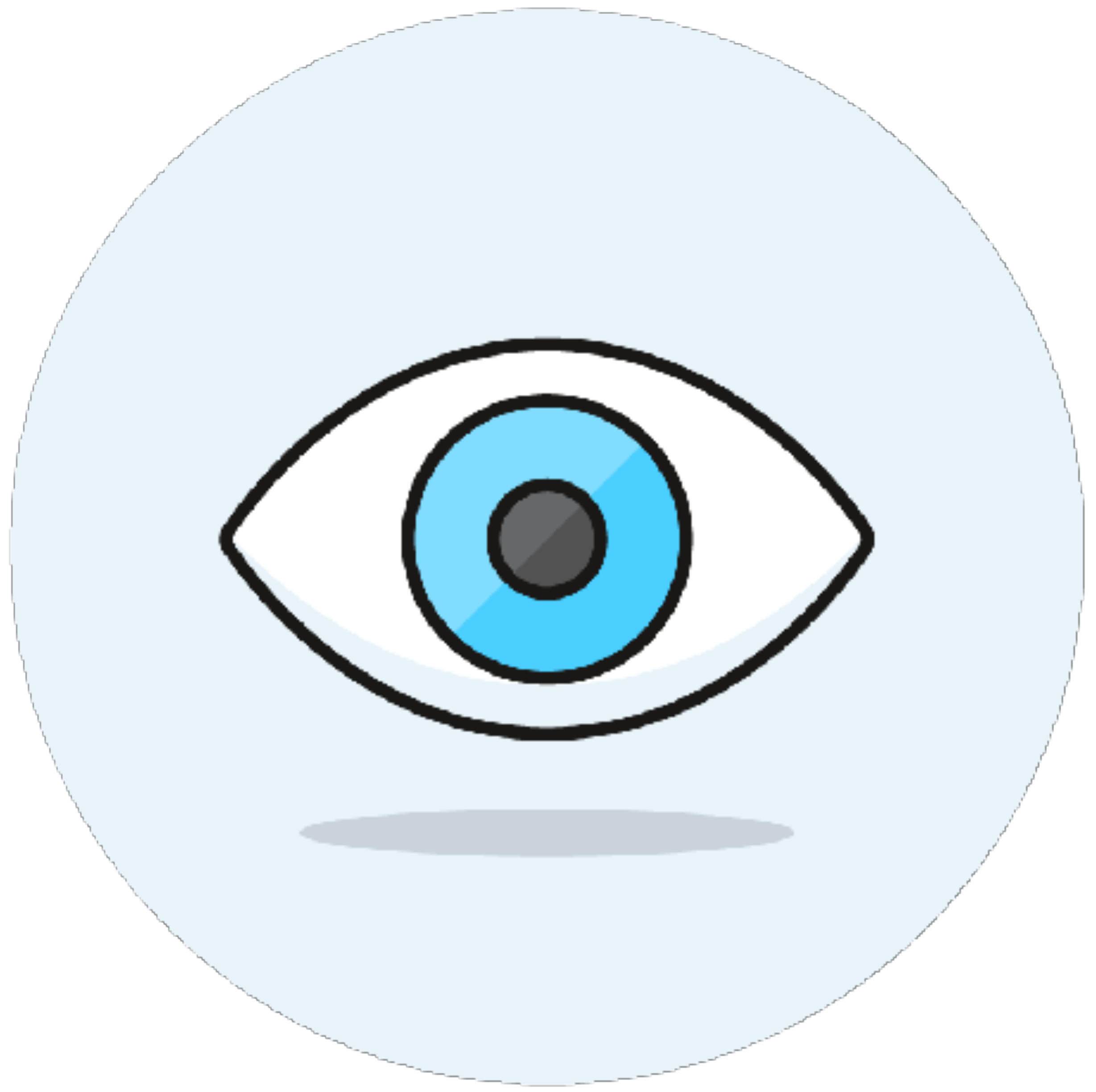
Reinforcement Learning

- Signals from rewards
- Trial and error
- Simulate at high speeds
- Agent becomes optimal

Imitation Learning

- Learning through demonstrations
- No rewards
- Simulate in real-time (mostly)
- Agent becomes human-like

Actions



Observations

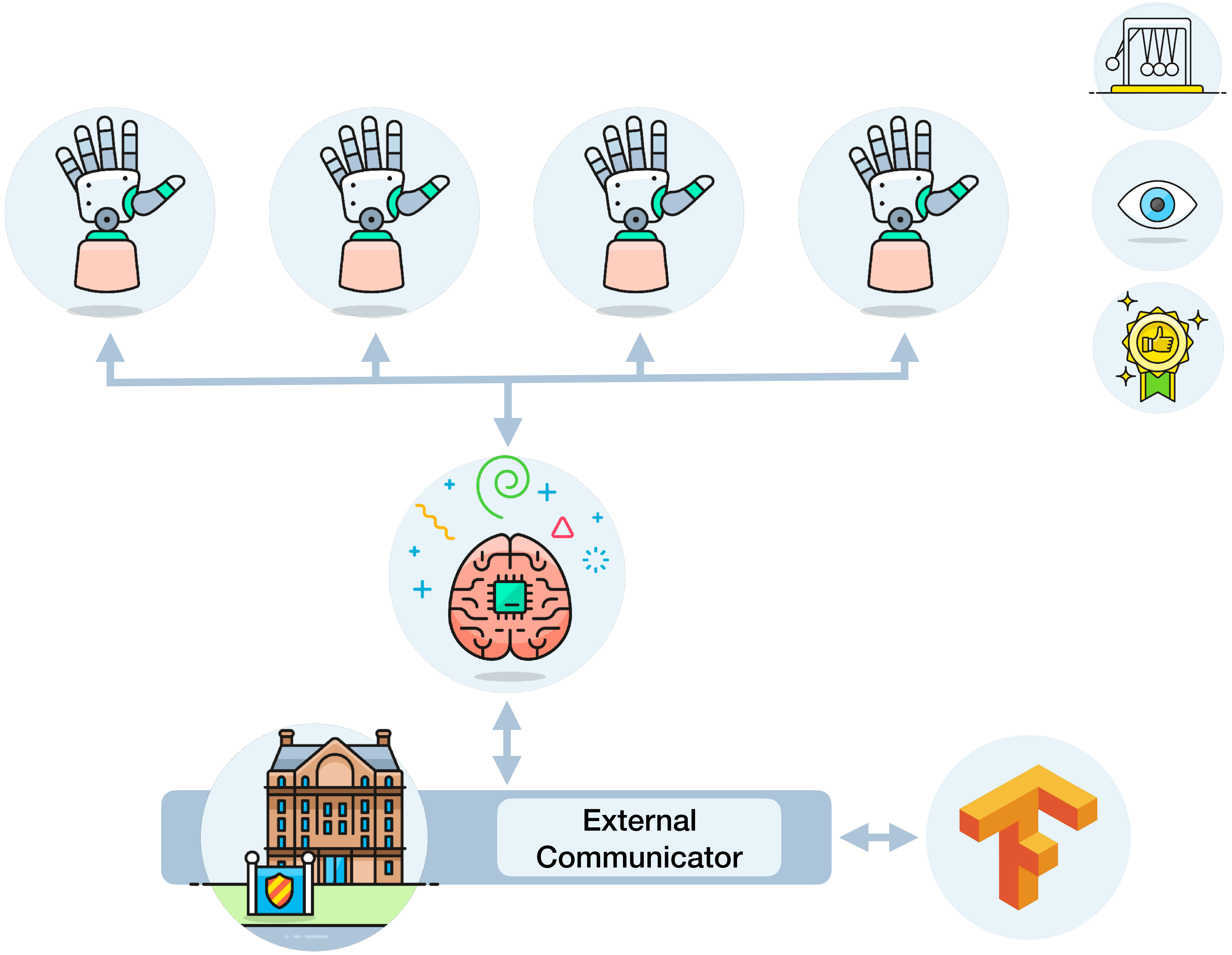
Rewards

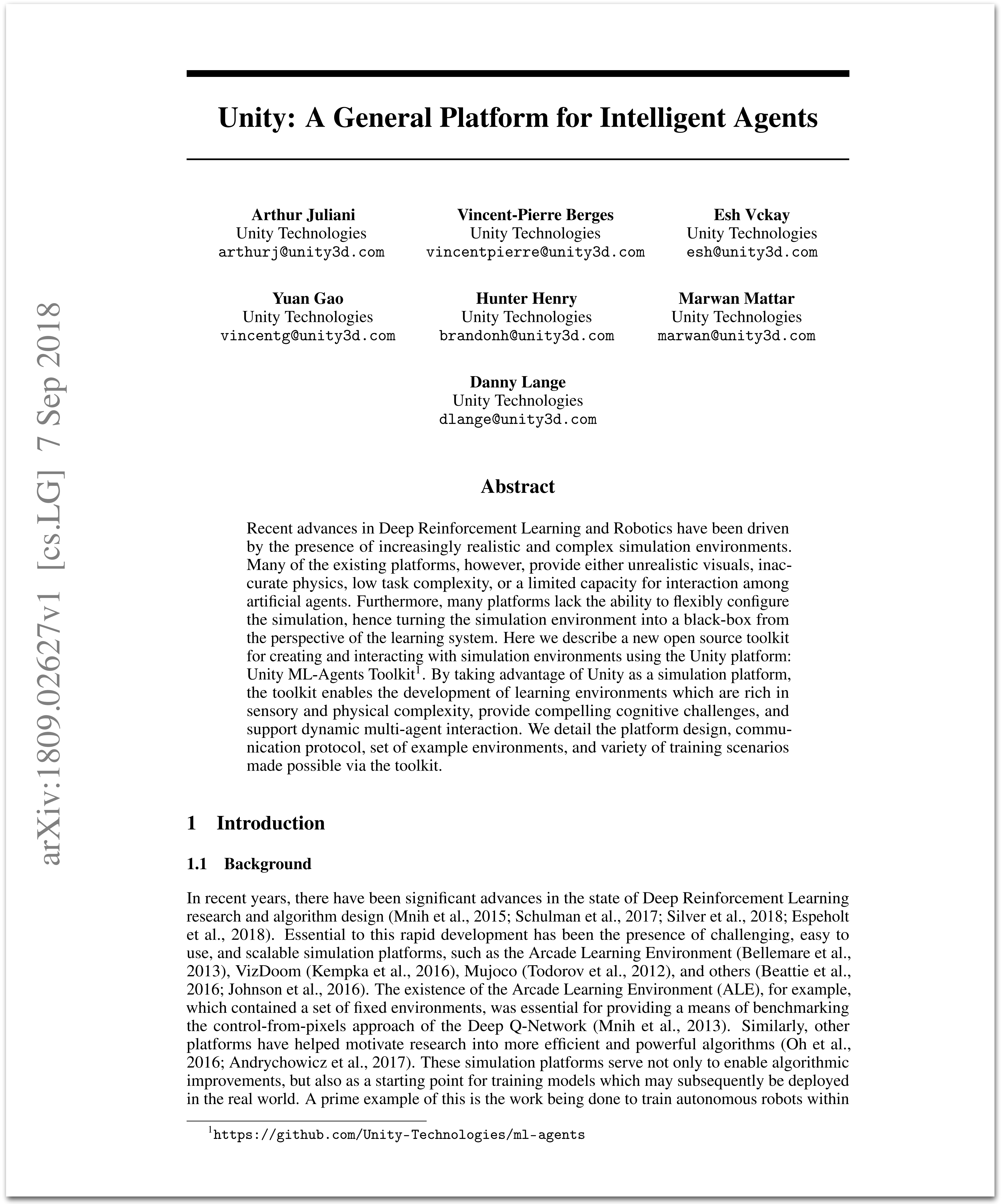
Reinforcement Learning

- Signals from rewards
- Trial and error
- Simulate at high speeds
- Agent becomes optimal

Imitation Learning

- Learning through demonstrations
- No rewards
- Simulate in real-time (mostly)
- Agent becomes human-like


External Communicator



Unity: A General Platform for Intelligent Agents

Arthur Juliani
Unity Technologies
arthurj@unity3d.com

Vincent-Pierre Berges
Unity Technologies
vincentpierre@unity3d.com

Esh Vckay
Unity Technologies
esh@unity3d.com

Yuan Gao
Unity Technologies
vincentg@unity3d.com

Hunter Henry
Unity Technologies
brandonh@unity3d.com

Marwan Mattar
Unity Technologies
marwan@unity3d.com

Danny Lange
Unity Technologies
dlange@unity3d.com

Abstract

Recent advances in Deep Reinforcement Learning and Robotics have been driven by the presence of increasingly realistic and complex simulation environments. Many of the existing platforms, however, provide either unrealistic visuals, inaccurate physics, low task complexity, or a limited capacity for interaction among artificial agents. Furthermore, many platforms lack the ability to flexibly configure the simulation, hence turning the simulation environment into a black-box from the perspective of the learning system. Here we describe a new open source toolkit for creating and interacting with simulation environments using the Unity platform: Unity ML-Agents Toolkit[1]. By taking advantage of Unity as a simulation platform, the toolkit enables the development of learning environments which are rich in sensory and physical complexity, provide compelling cognitive challenges, and support dynamic multi-agent interaction. We detail the platform design, communication protocol, set of example environments, and variety of training scenarios made possible via the toolkit.

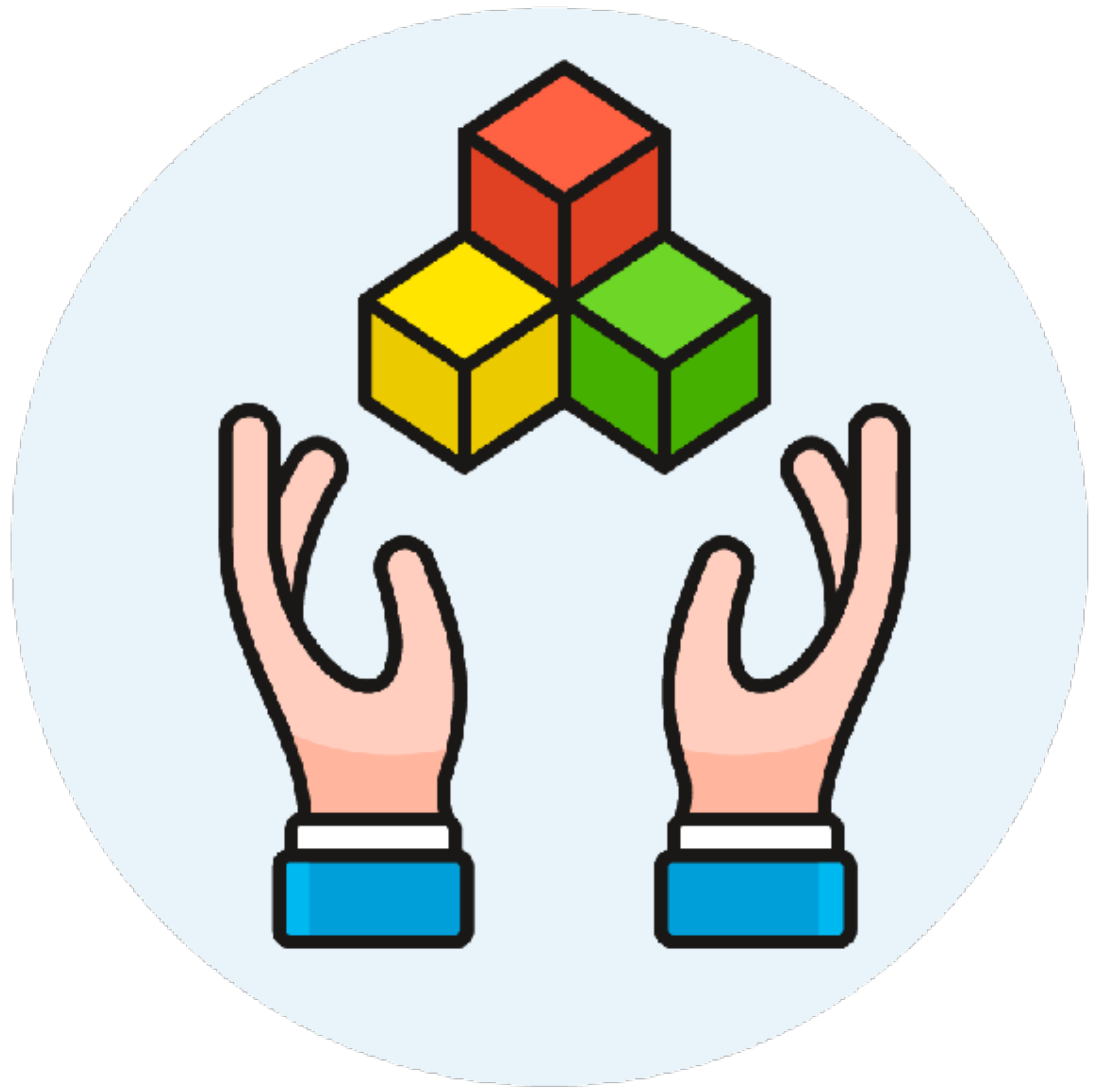
1 Introduction

1.1 Background

In recent years, there have been significant advances in the state of Deep Reinforcement Learning research and algorithm design (Mnih et al., 2015; Schulman et al., 2017; Silver et al., 2018; Espeholt et al., 2018). Essential to this rapid development has been the presence of challenging, easy to use, and scalable simulation platforms, such as the Arcade Learning Environment (Bellemare et al., 2013), VizDoom (Kempka et al., 2016), Mujoco (Todorov et al., 2012), and others (Beattie et al., 2016; Johnson et al., 2016). The existence of the Arcade Learning Environment (ALE), for example, which contained a set of fixed environments, was essential for providing a means of benchmarking the control-from-pixels approach of the Deep Q-Network (Mnih et al., 2013). Similarly, other platforms have helped motivate research into more efficient and powerful algorithms (Oh et al., 2016; Andrychowicz et al., 2017). These simulation platforms serve not only to enable algorithmic improvements, but also as a starting point for training models which may subsequently be deployed in the real world. A prime example of this is the work being done to train autonomous robots within

[1]https://github.com/Unity-Technologies/ml-agents

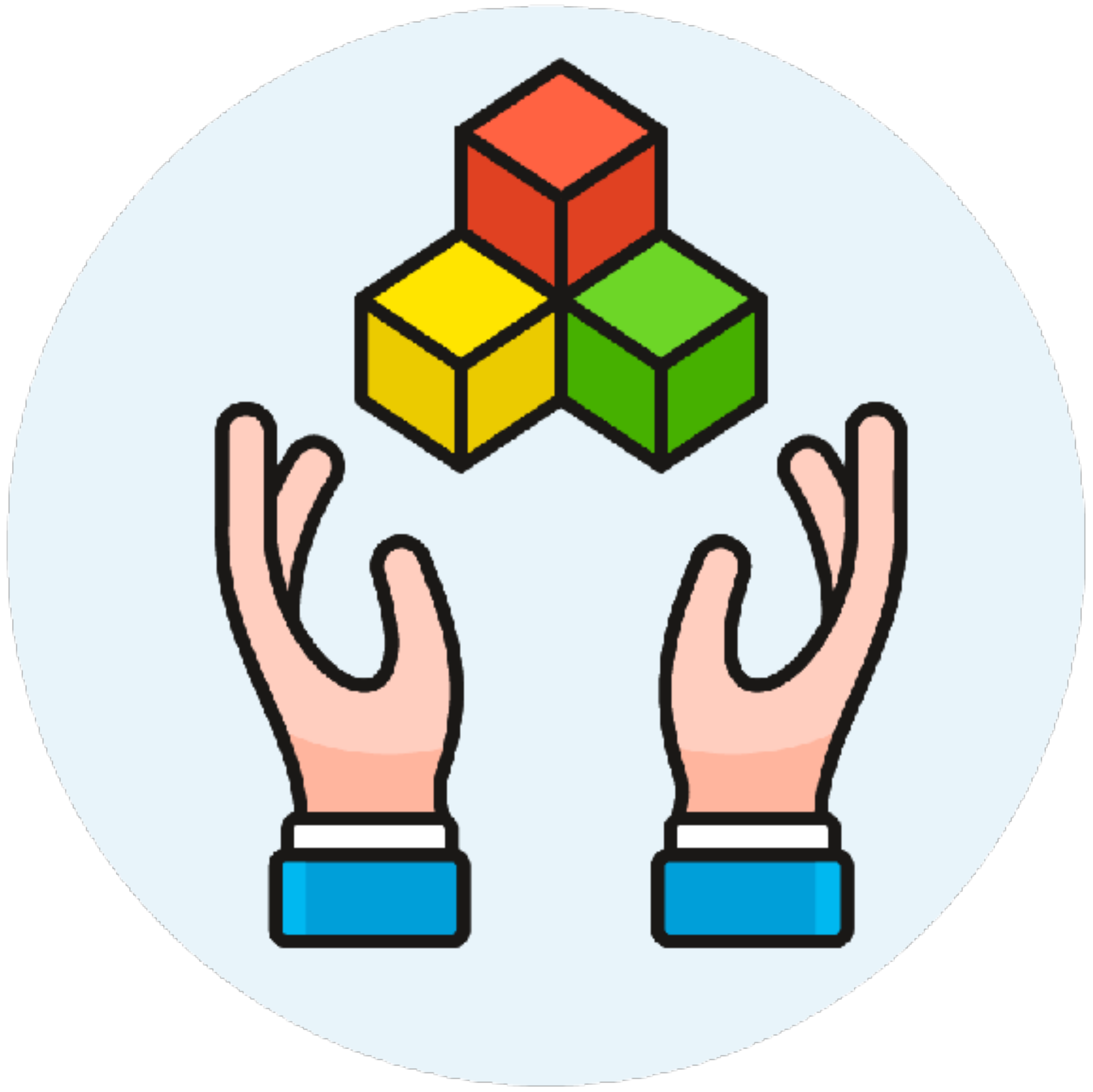
https://arxiv.org/abs/1809.02627



The Process

Imitation Learning

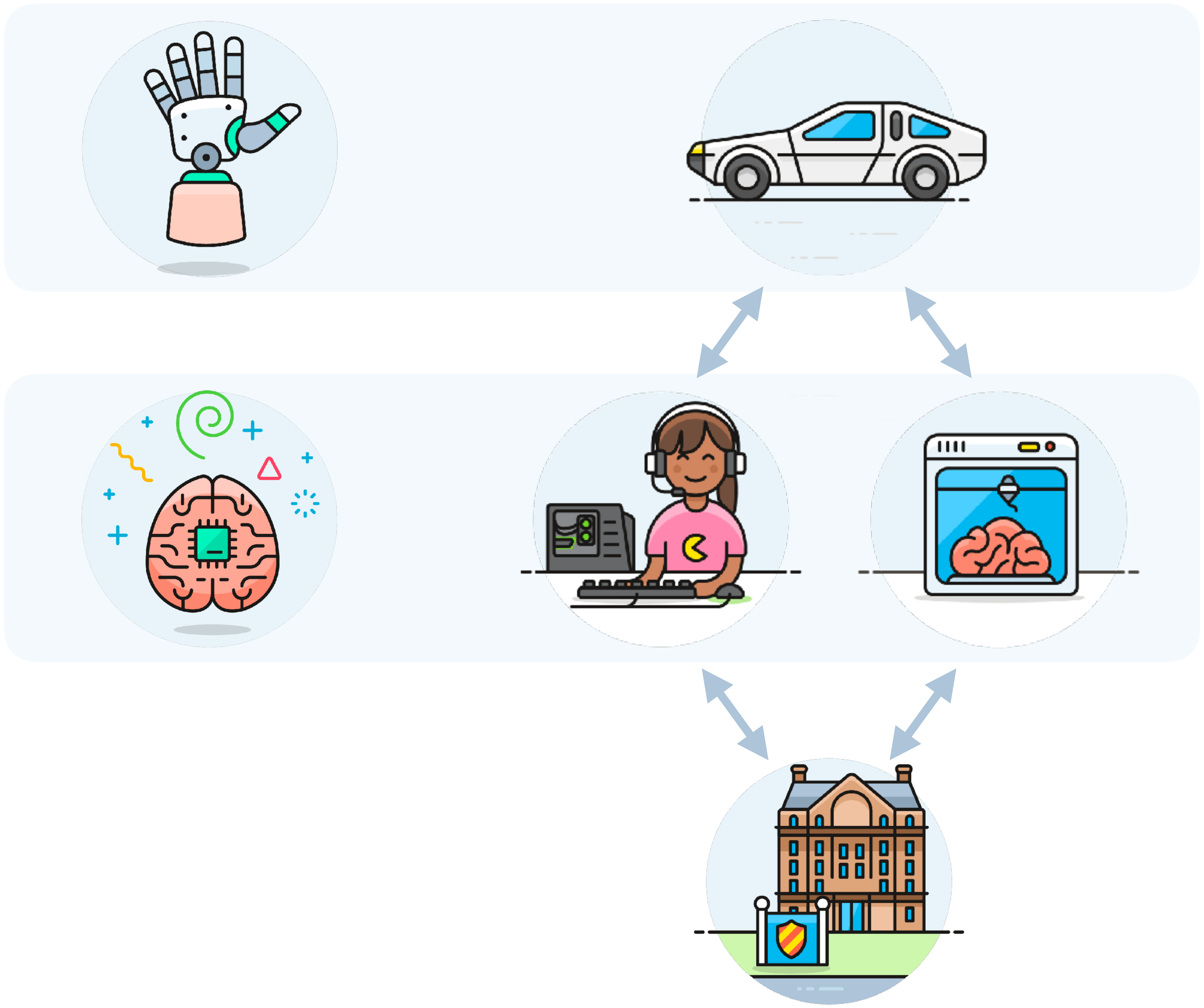
Step by Step



- Pick a task
- Create an environment
- Create/identify the agent
- Create an academy
- Pick a learning/training method
- Create observations, rewards, and actions
- Pick algorithms, tune, and train

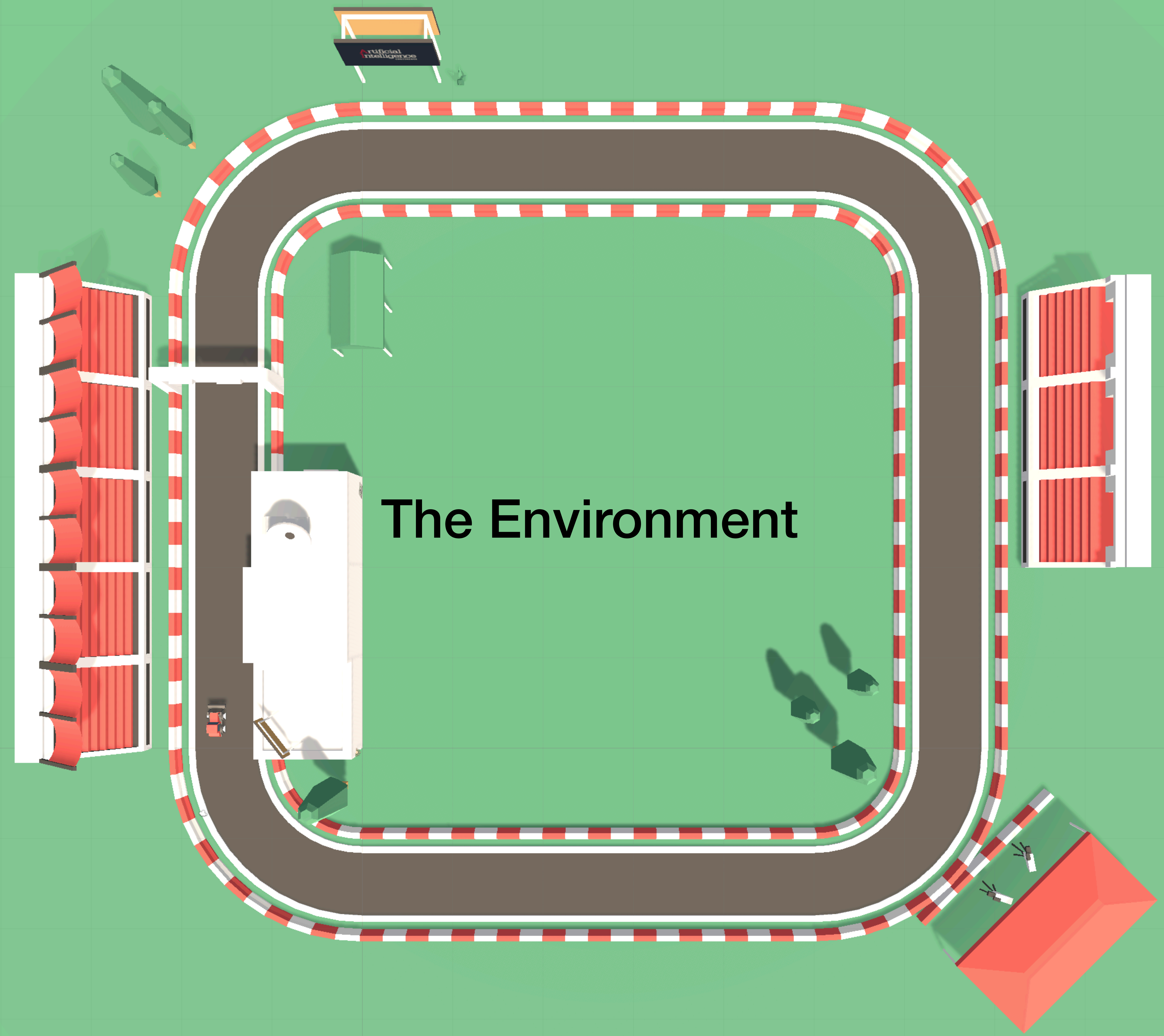
Step by Step

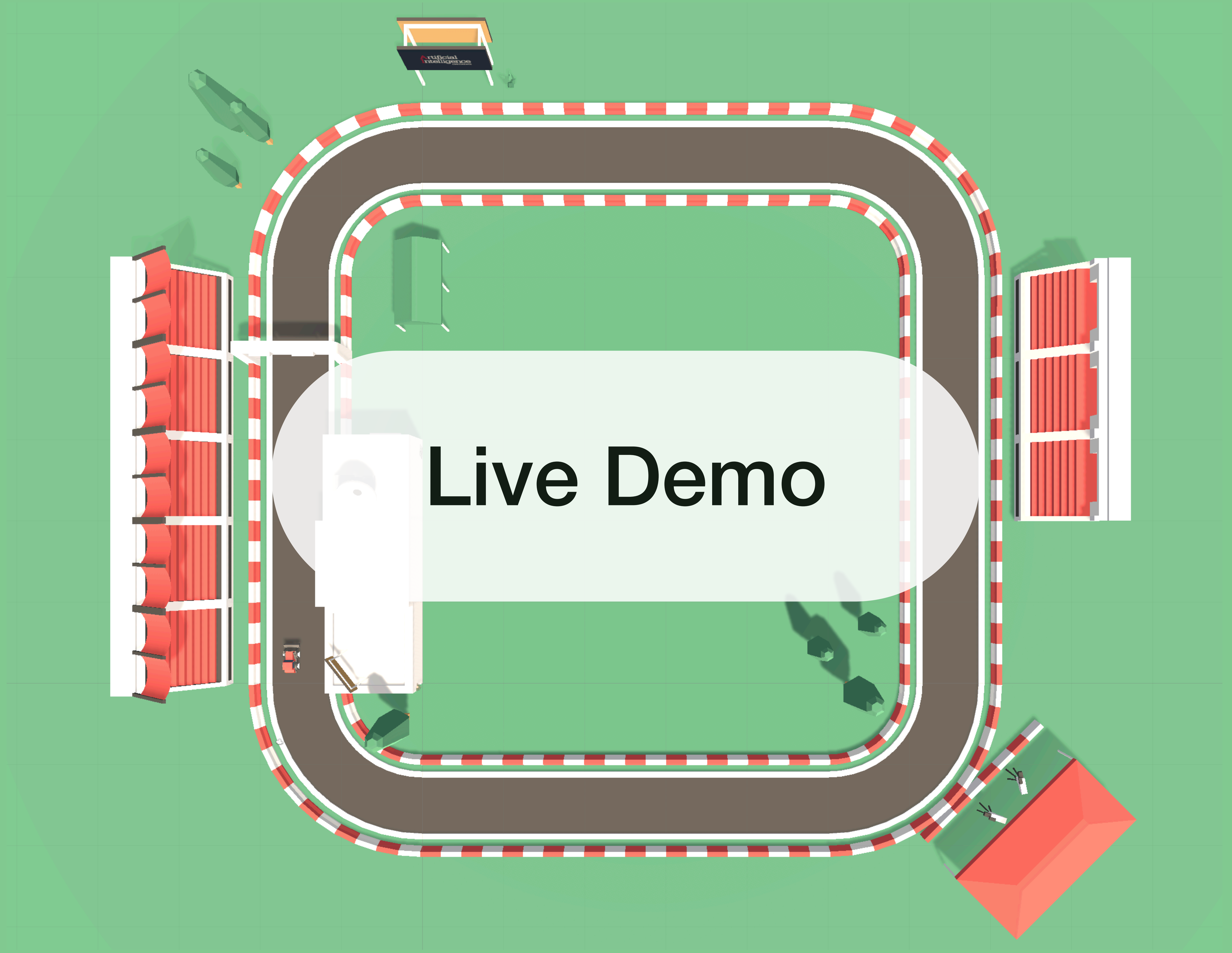
A car that drives by itself	• Pick a task
Cartoony race track	• Create an environment
Our self-driving car	• Create/identify the agent
A bog-standard Academy	• Create an academy
Imitation Learning	• Pick a learning/training method
Raycasts, Modify transform	• Create observations, rewards, and actions
Train!	• Pick algorithms, tune, and train



The Environment

The Environment



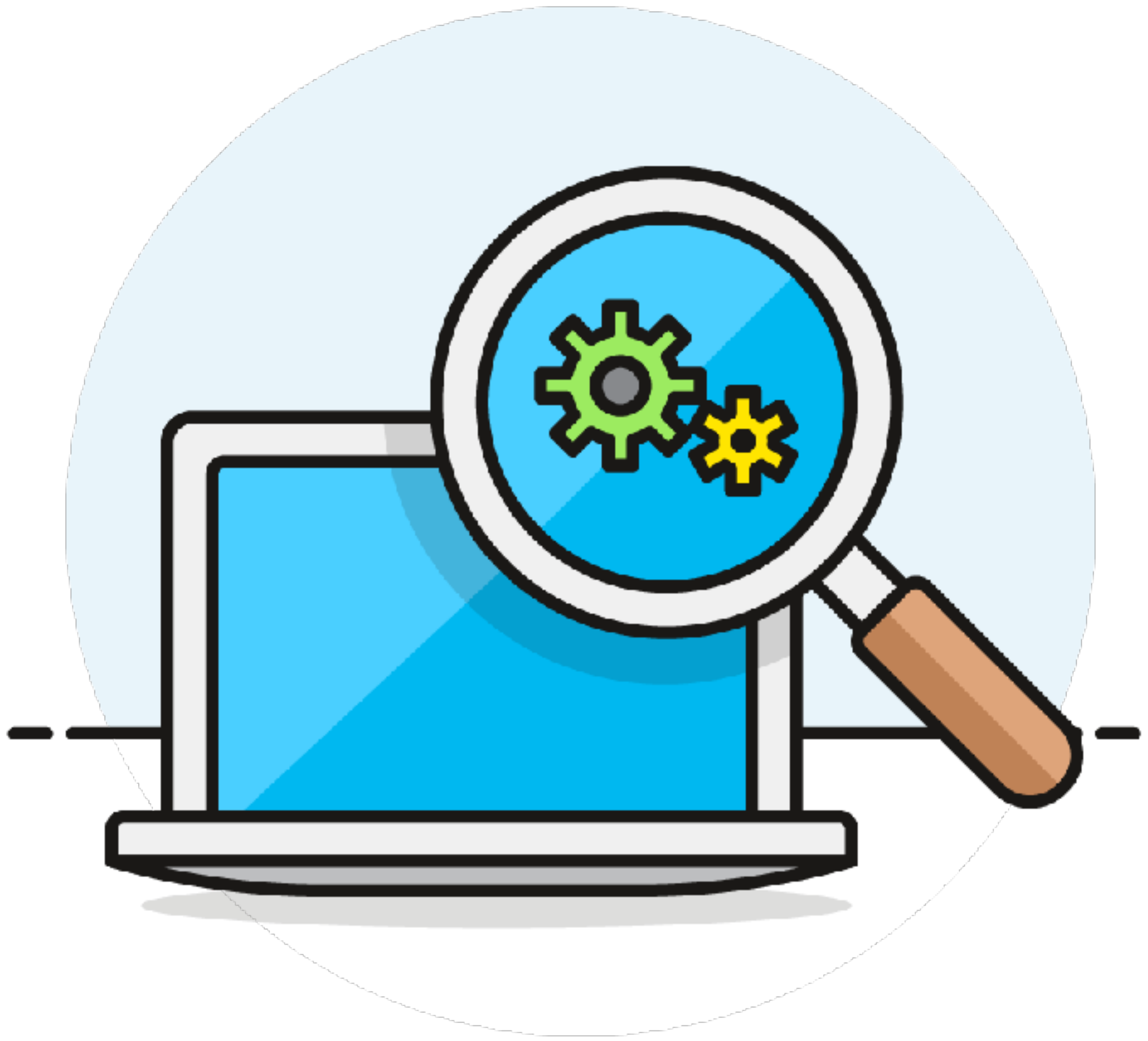


Live Demo

Imitation learning

Imitation Learning

- Learning through demonstrations
- No rewards
- Simulate in real-time (mostly)
- Agent becomes human-like



So What?

Reinforcement Learning

- Signals from rewards
- Trial and error
- Simulate at high speeds
- Agent becomes optimal

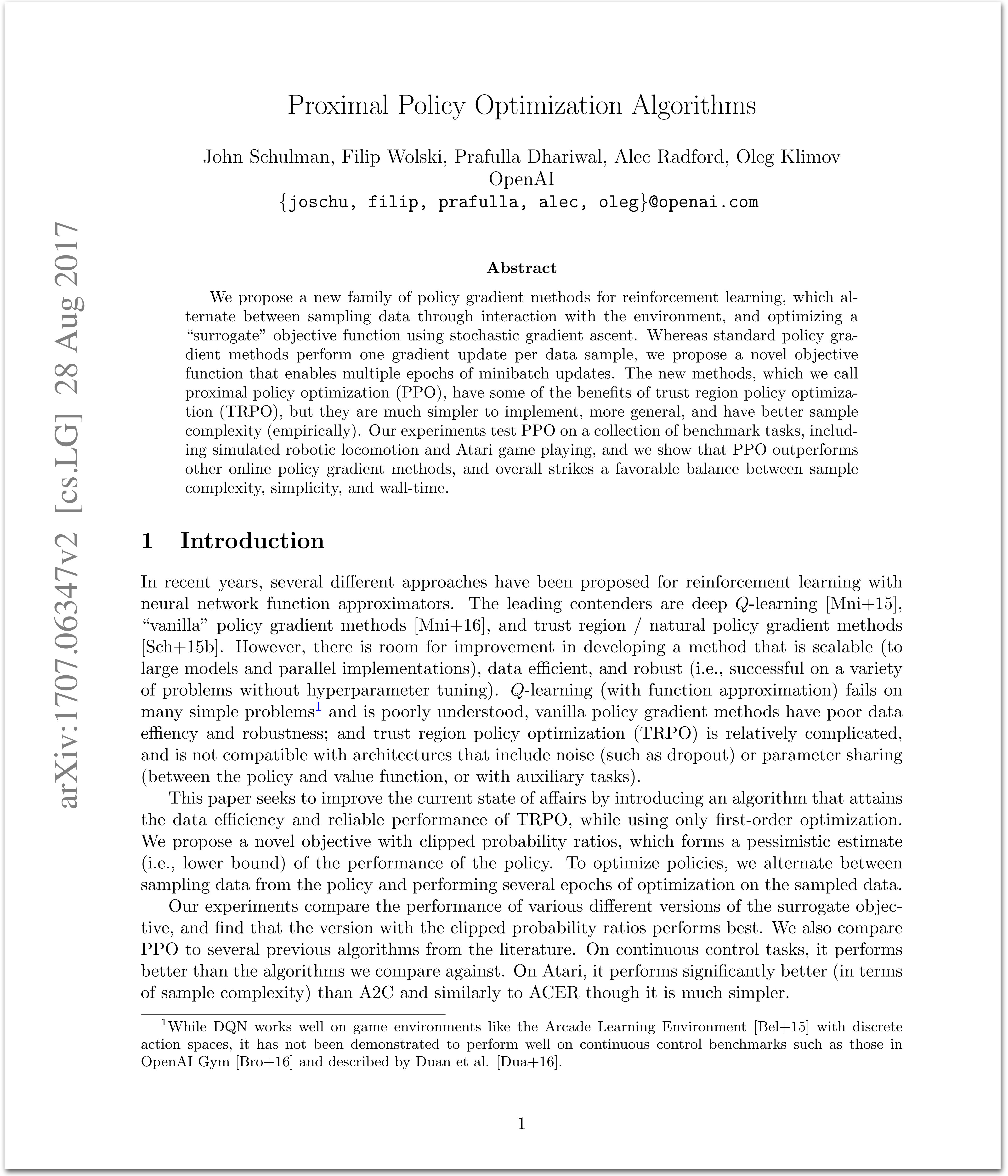
Imitation Learning

- Learning through demonstrations
- No rewards
- Simulate in real-time (mostly)
- Agent becomes human-like

Rewards in Actions

Actions

Rewards



Proximal Policy Optimization Algorithms

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov
OpenAI
{joschu, filip, prafulla, alec, oleg}@openai.com

Abstract

We propose a new family of policy gradient methods for reinforcement learning, which alternate between sampling data through interaction with the environment, and optimizing a "surrogate" objective function using stochastic gradient ascent. Whereas standard policy gradient methods perform one gradient update per data sample, we propose a novel objective function that enables multiple epochs of minibatch updates. The new methods, which we call proximal policy optimization (PPO), have some of the benefits of trust region policy optimization (TRPO), but they are much simpler to implement, more general, and have better sample complexity (empirically). Our experiments test PPO on a collection of benchmark tasks, including simulated robotic locomotion and Atari game playing, and we show that PPO outperforms other online policy gradient methods, and overall strikes a favorable balance between sample complexity, simplicity, and wall-time.

1 Introduction

In recent years, several different approaches have been proposed for reinforcement learning with neural network function approximators. The leading contenders are deep Q-learning [Mni+15], "vanilla" policy gradient methods [Mni+16], and trust region / natural policy gradient methods [Sch+15b]. However, there is room for improvement in developing a method that is scalable (to large models and parallel implementations), data efficient, and robust (i.e., successful on a variety of problems without hyperparameter tuning). Q-learning (with function approximation) fails on many simple problems[1] and is poorly understood, vanilla policy gradient methods have poor data effiency and robustness; and trust region policy optimization (TRPO) is relatively complicated, and is not compatible with architectures that include noise (such as dropout) or parameter sharing (between the policy and value function, or with auxiliary tasks).

This paper seeks to improve the current state of affairs by introducing an algorithm that attains the data efficiency and reliable performance of TRPO, while using only first-order optimization. We propose a novel objective with clipped probability ratios, which forms a pessimistic estimate (i.e., lower bound) of the performance of the policy. To optimize policies, we alternate between sampling data from the policy and performing several epochs of optimization on the sampled data.

Our experiments compare the performance of various different versions of the surrogate objective, and find that the version with the clipped probability ratios performs best. We also compare PPO to several previous algorithms from the literature. On continuous control tasks, it performs better than the algorithms we compare against. On Atari, it performs significantly better (in terms of sample complexity) than A2C and similarly to ACER though it is much simpler.

[1]While DQN works well on game environments like the Arcade Learning Environment [Bel+15] with discrete action spaces, it has not been demonstrated to perform well on continuous control benchmarks such as those in OpenAI Gym [Bro+16] and described by Duan et al. [Dua+16].

https://arxiv.org/abs/1707.06347

TensorFlow

“That seems more useful.”

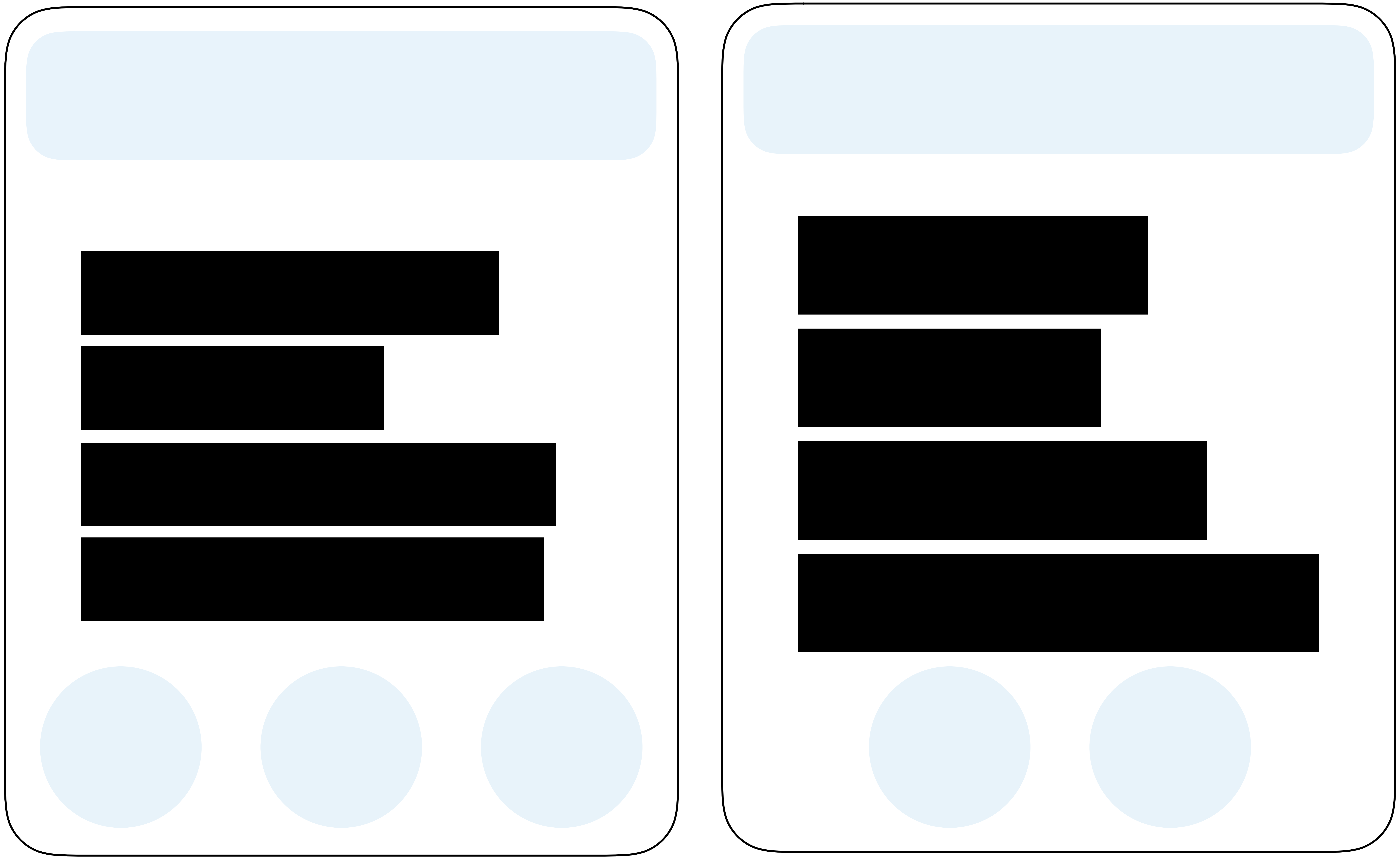
–You, probably.

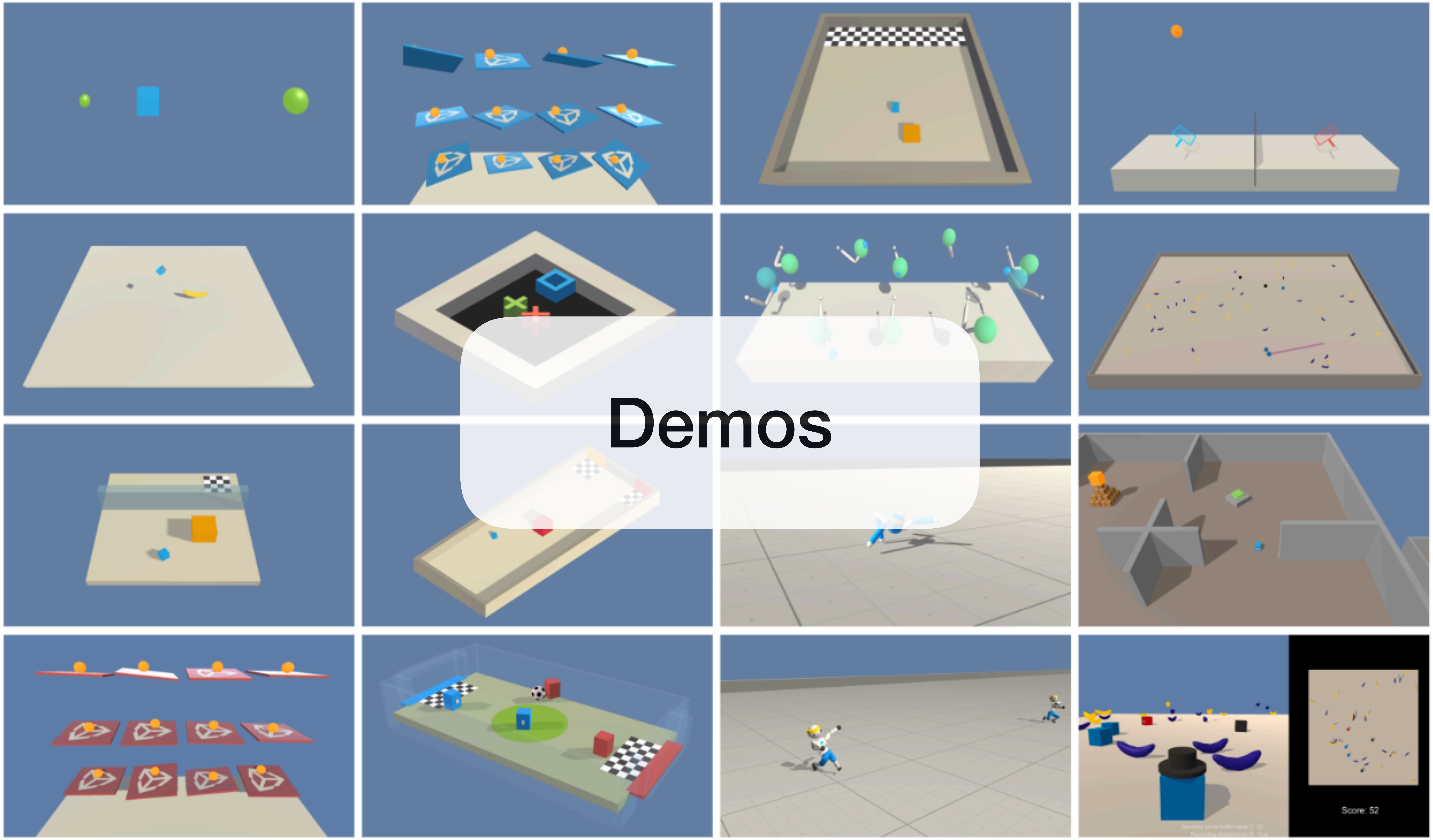
Reinforcement Learning

- Signals from rewards
- Trial and error
- Simulate at high speeds
- Agent becomes optimal

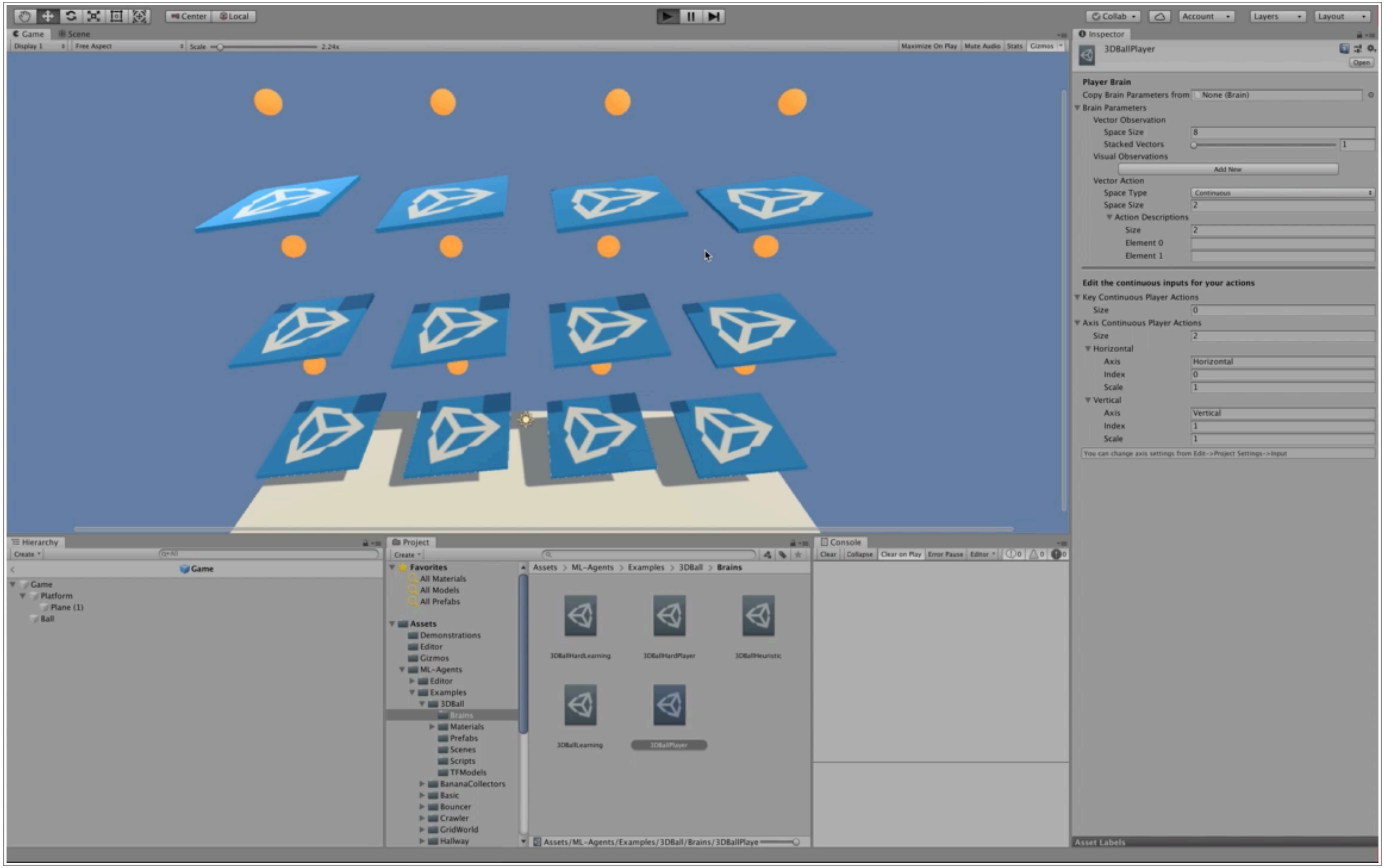
Imitation Learning

- Learning through demonstrations
- No rewards
- Simulate in real-time (mostly)
- Agent becomes human-like





Demos



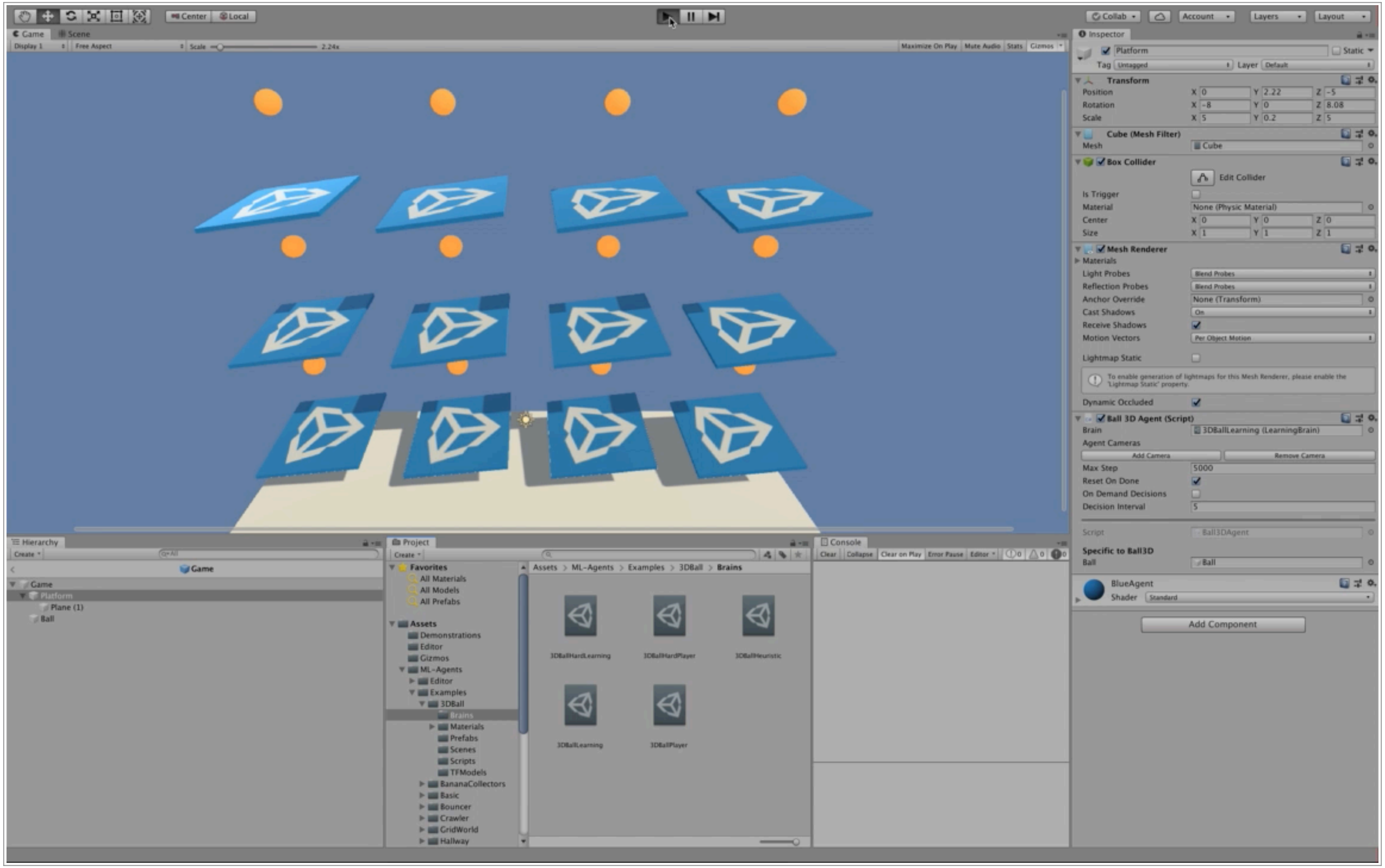
3DBallPlayer

Player Brain

```
Copy Brain Parameters from   None (Brain)
▼ Brain Parameters
    Vector Observation
        Space Size            8
        Stacked Vectors       1
    Visual Observations
                  Add New
    Vector Action
        Space Type            Continuous
        Space Size            2
        ▼ Action Descriptions
            Size              2
            Element 0
            Element 1
```

Edit the continuous inputs for your actions

```
▼ Key Continuous Player Actions
    Size                      0
▼ Axis Continuous Player Actions
    Size                      2
    ▼ Horizontal
        Axis                  Horizontal
        Index                 0
        Scale                 1
    ▼ Vertical
        Axis                  Vertical
        Index                 1
        Scale                 1
You can change axis settings from Edit->Project Settings->Input
```



Platform — Static
Tag: Untagged — Layer: Default

Transform

	X	Y	Z
Position	0	2.22	-5
Rotation	-8	0	8.08
Scale	5	0.2	5

Cube (Mesh Filter)
Mesh: Cube

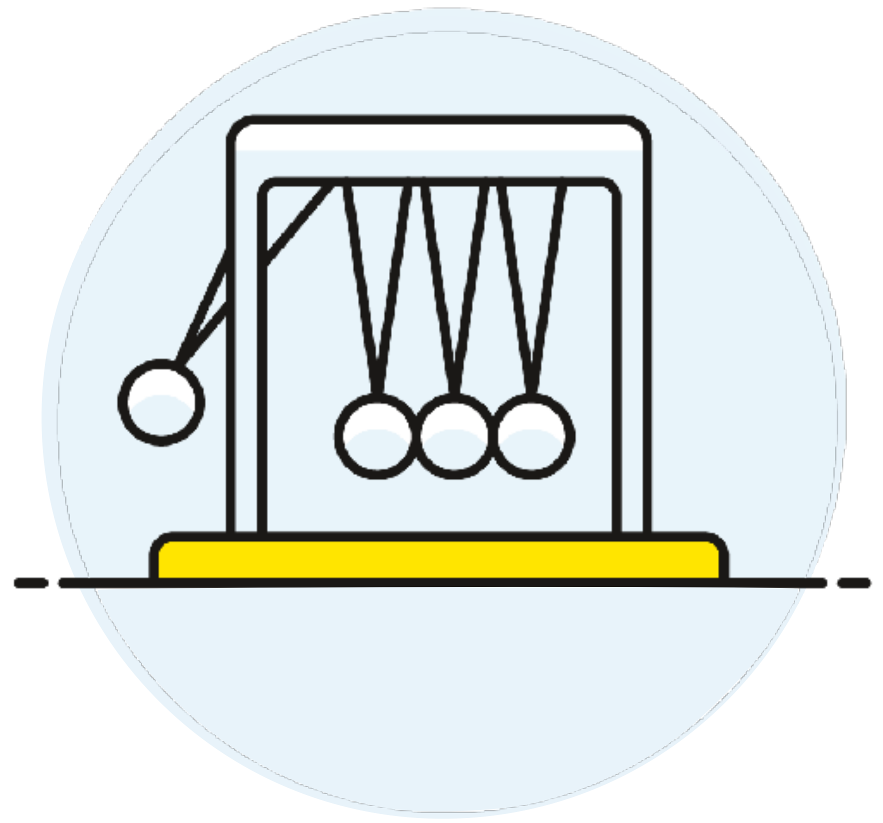
Box Collider
Edit Collider
Is Trigger: ☐
Material: None (Physic Material)
Center: X 0 Y 0 Z 0
Size: X 1 Y 1 Z 1

Mesh Renderer
Materials
Light Probes: Blend Probes
Reflection Probes: Blend Probes
Anchor Override: None (Transform)
Cast Shadows: On
Receive Shadows: ✓
Motion Vectors: Per Object Motion
Lightmap Static: ☐
To enable generation of lightmaps for this Mesh Renderer, please enable the 'Lightmap Static' property.
Dynamic Occluded: ✓

Ball 3D Agent (Script)
Brain: 3DBallLearning (LearningBrain)
Agent Cameras
Add Camera — Remove Camera
Max Step: 5000
Reset On Done: ✓
On Demand Decisions: ☐
Decision Interval: 5
Script: Ball3DAgent
Specific to Ball3D
Ball: Ball

BlueAgent
Shader: Standard

Add Component



Actions

X-rotation

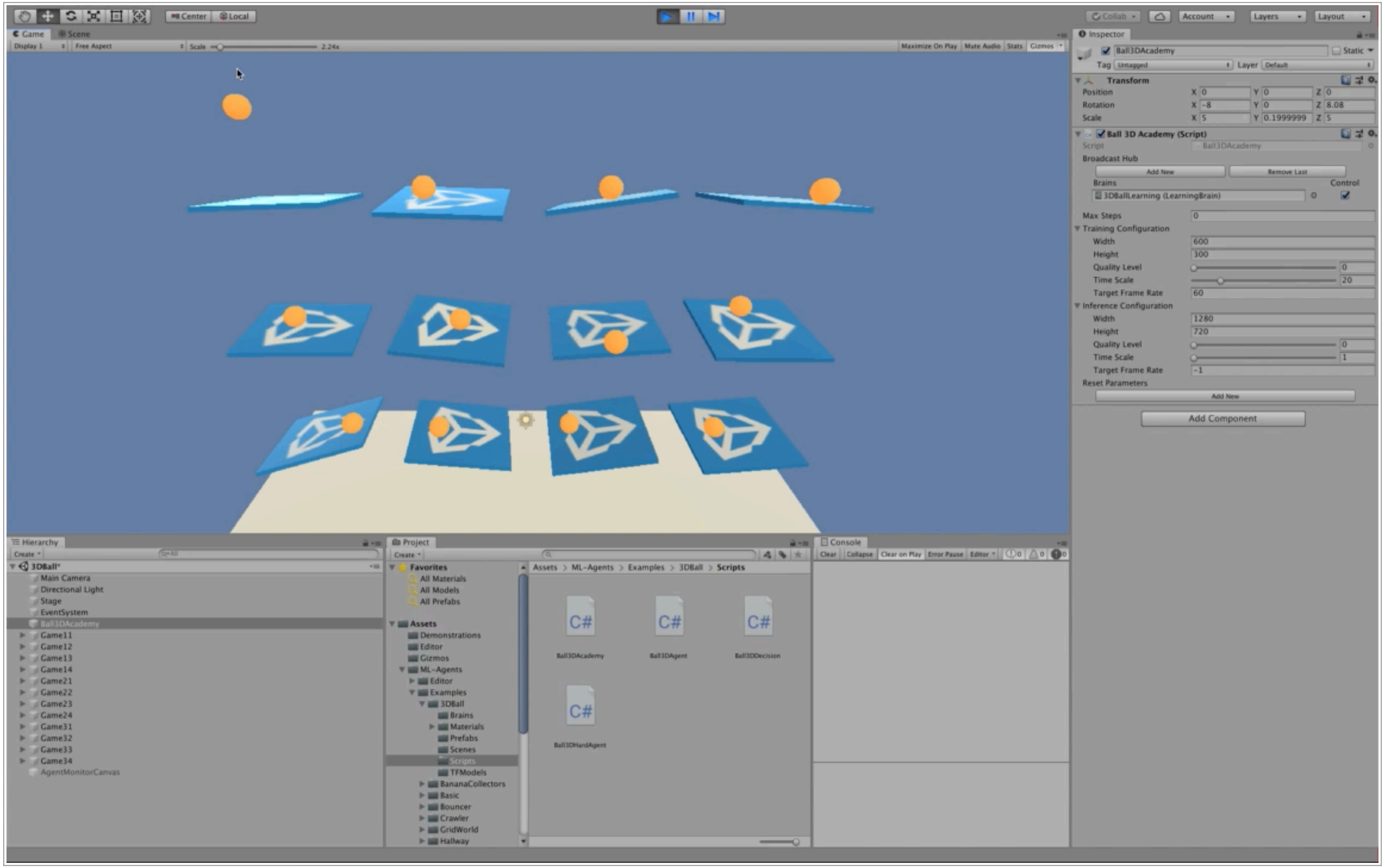
Z-rotation

Observations

```
AddVectorObs(gameObject.transform.rotation.z);
AddVectorObs(gameObject.transform.rotation.x);
AddVectorObs(ball.transform.position - gameObject.transform.position);
AddVectorObs(ballRb.velocity);
```

Rewards

```
if ((ball.transform.position.y - gameObject.transform.position.y) < -2f ||
    Mathf.Abs(ball.transform.position.x - gameObject.transform.position.x) > 3f ||
    Mathf.Abs(ball.transform.position.z - gameObject.transform.position.z) > 3f)
{
    Done();
    SetReward(-1f);
}
else
{
    SetReward(0.1f);
}
```



Inspector

Ball3DAcademy   Static
Tag Untagged   Layer Default

Transform
Position X 0 Y 0 Z 0
Rotation X -8 Y 0 Z 8.08
Scale X 5 Y 0.1999999 Z 5

Ball 3D Academy (Script)
Script Ball3DAcademy
Broadcast Hub
Add New   Remove Last
Brains   Control
3DBallLearning (LearningBrain) ✓

Max Steps 0
Training Configuration
Width 600
Height 300
Quality Level 0
Time Scale 20
Target Frame Rate 60
Inference Configuration
Width 1280
Height 720
Quality Level 0
Time Scale 1
Target Frame Rate -1
Reset Parameters
Add New

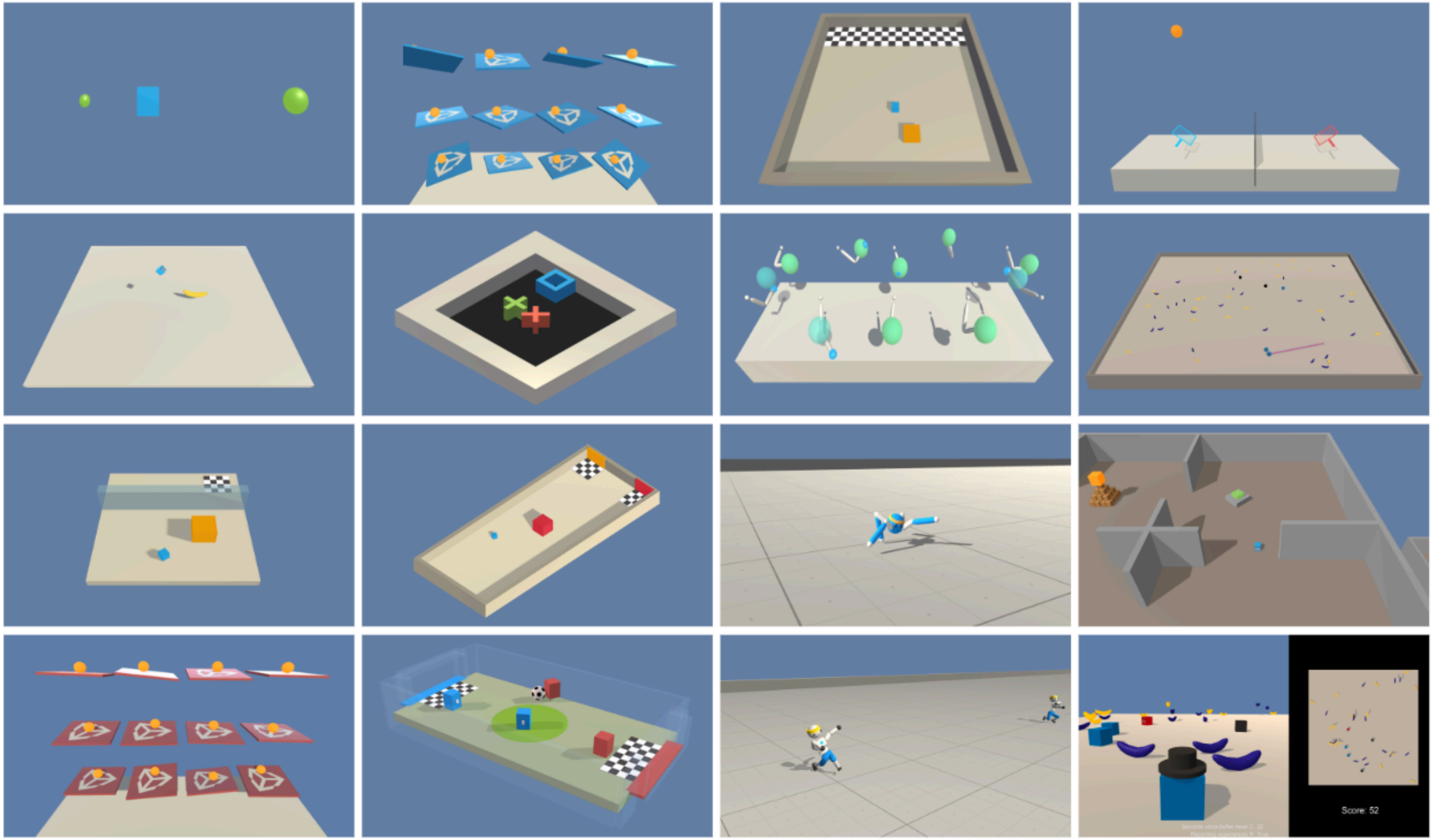
Add Component

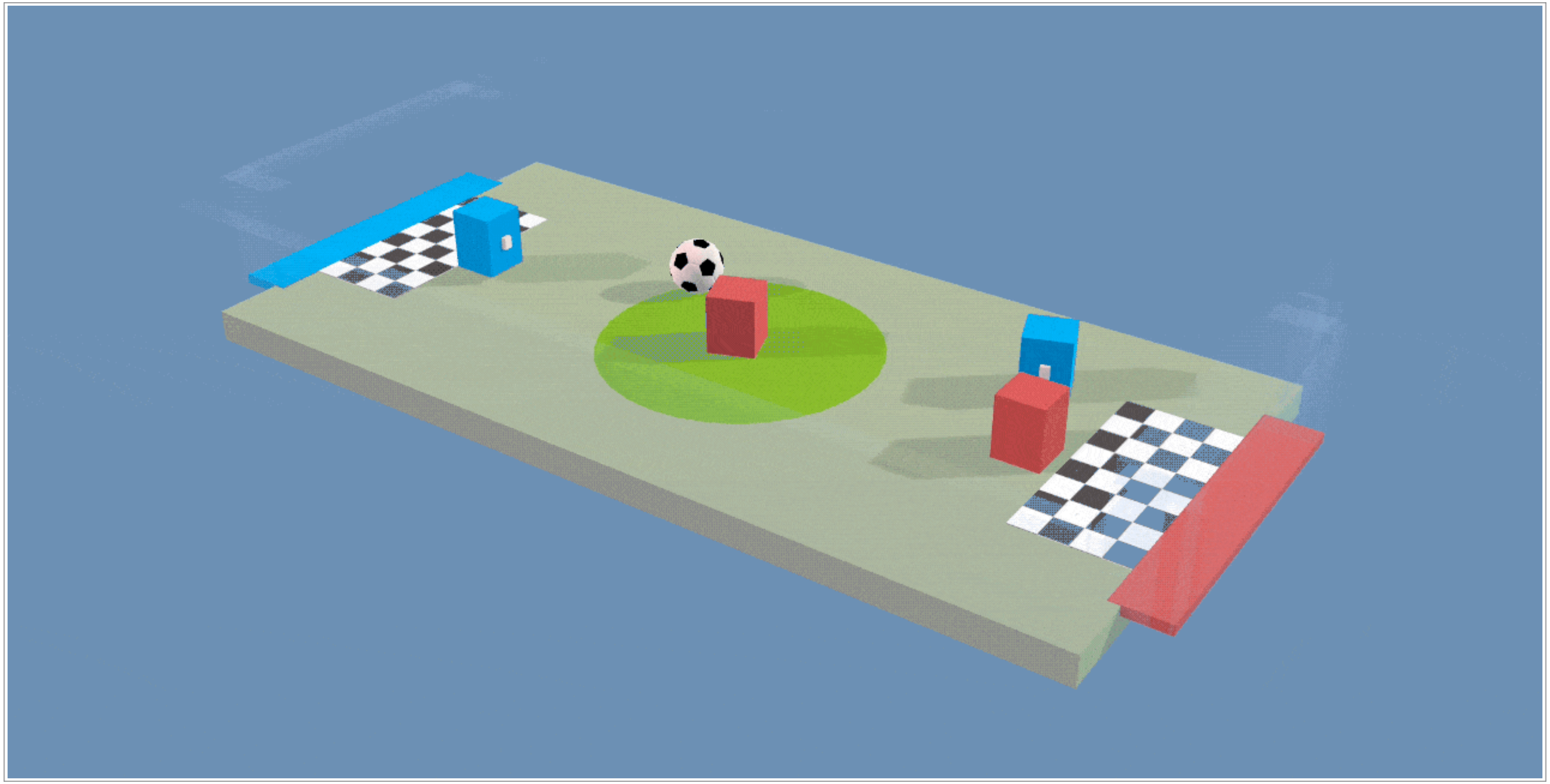
Hierarchy
Create
3DBall*
Main Camera
Directional Light
Stage
EventSystem
Ball3DAcademy
Game11
Game12
Game13
Game14
Game21
Game22
Game23
Game24
Game31
Game32
Game33
Game34
AgentMonitorCanvas

Project
Create
Favorites
All Materials
All Models
All Prefabs
Assets
Demonstrations
Editor
Gizmos
ML-Agents
Editor
Examples
3DBall
Brains
Materials
Prefabs
Scenes
Scripts
TFModels
BananaCollectors
Basic
Bouncer
Crawler
GridWorld
Hallway

Assets > ML-Agents > Examples > 3DBall > Scripts
Ball3DAcademy
Ball3DAgent
Ball3DDecision
Ball3DHardAgent

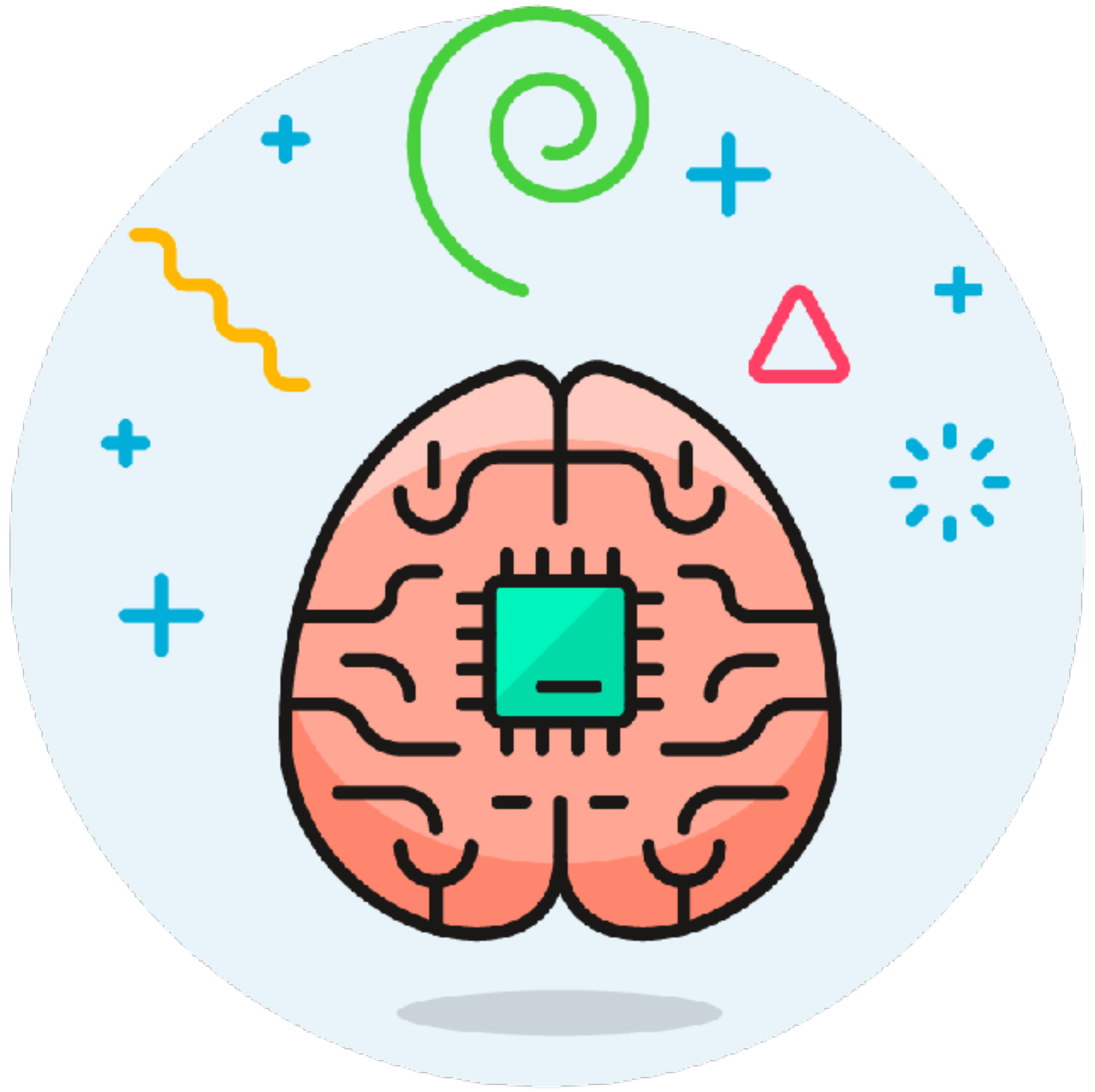
Console
Clear   Collapse   Clear on Play   Error Pause   Editor   0   0   0


Score: 52



Useful…?

- Training behaviours, rather than coding behaviours
- Exploring or training behaviours in physical, spatial, simulated scenarios
    - Self-driving cars
    - Warehouses, factories
- Low-risk, low-cost way to test visual, physical, cognitive machine learning problems
- "Free" visualisation!



A game engine is a controlled, self-contained spatial, physical environment that can (closely) replicate (enough of) the real world (to be useful).

(but it’s also useful for non-physical problems that you might be able to make a physical representation of and observe.)

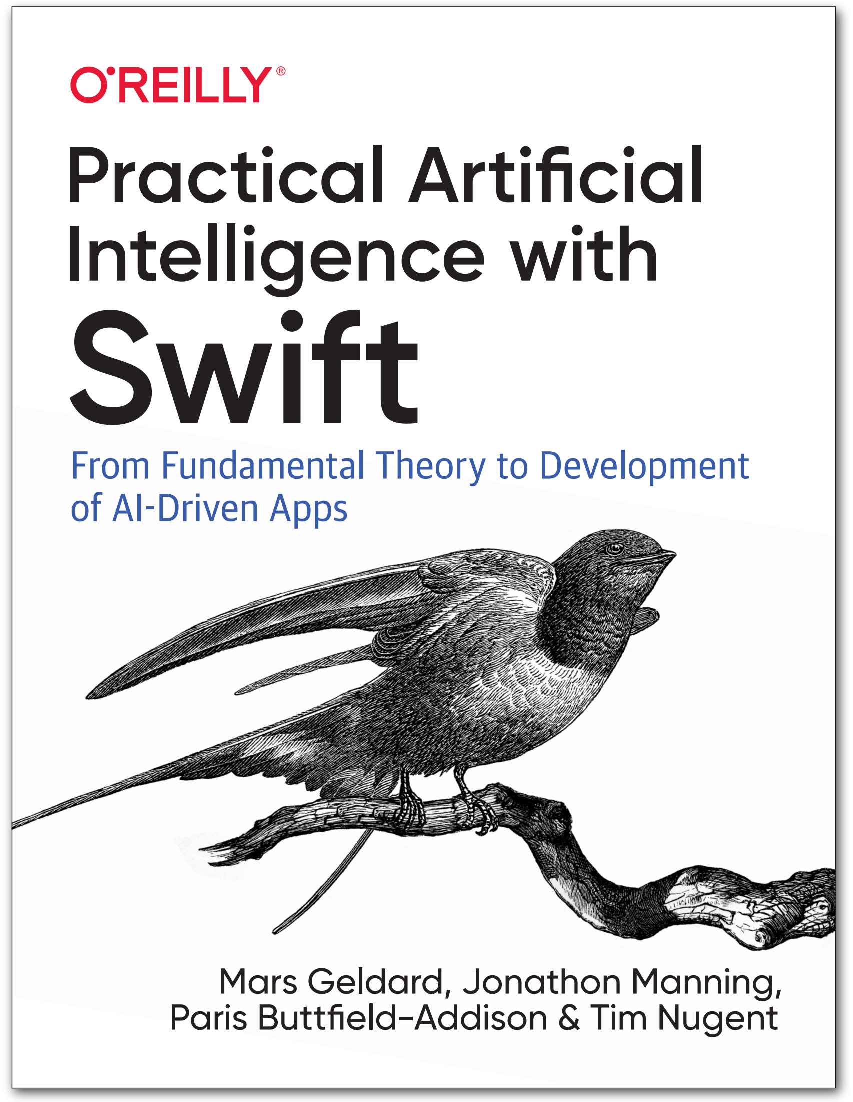
Thank you

@themartianlife

@the_mcjones

@parisba

Thank you



O'REILLY

Practical Artificial Intelligence with Swift

From Fundamental Theory to Development of AI-Driven Apps

Mars Geldard, Jonathon Manning, Paris Buttfield-Addison & Tim Nugent

@themartianlife @the_mcjones @parisba @aiwithswift

At #OSCON? Join us for a half-day tutorial on Unity Machine Learning!

https://lab.to/AIConfNYC2019