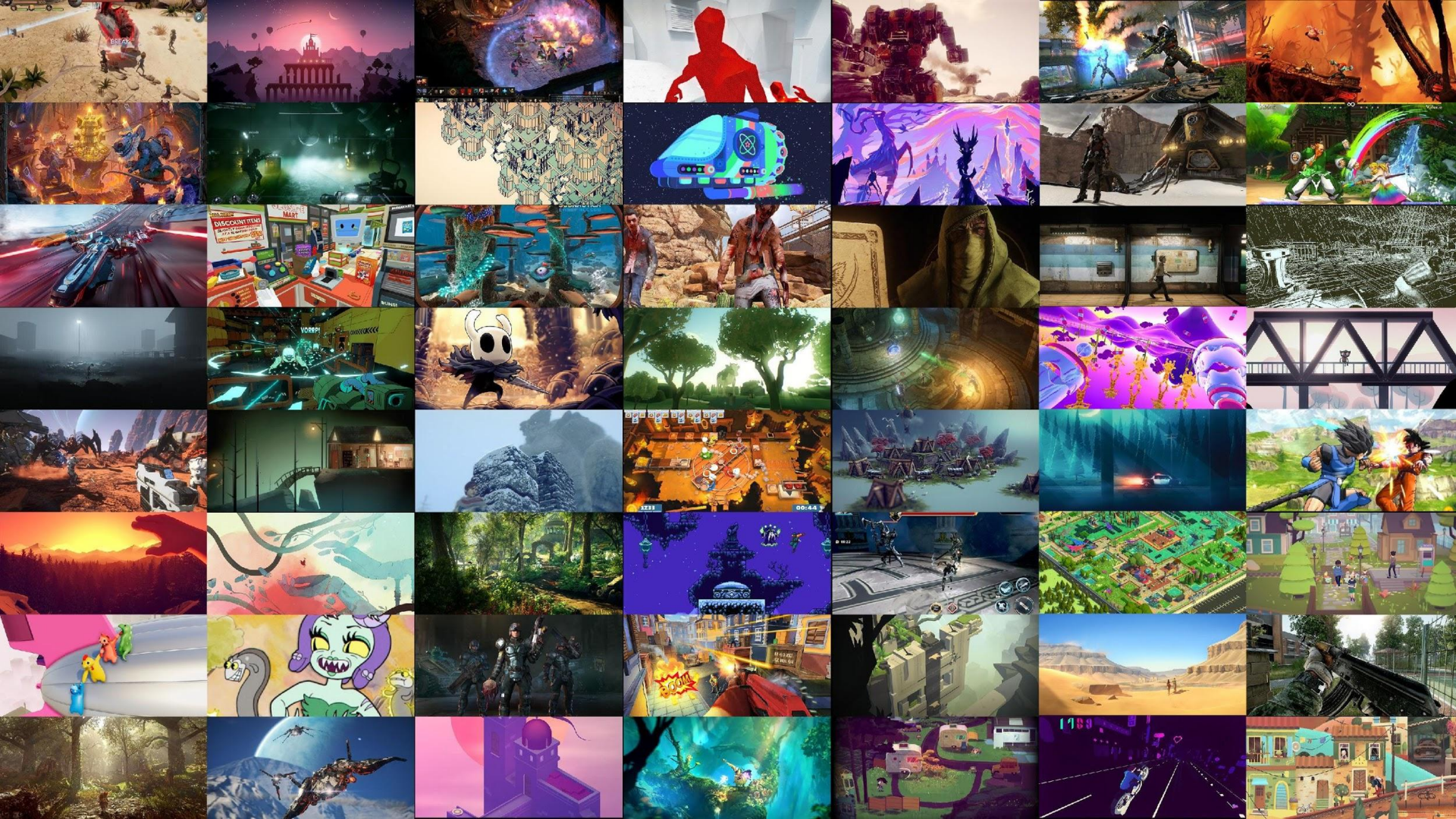
Successfully Use Deep Reinforcement Learning in Testing and NPC Development

Jeffrey Shih, Lead Product Manager, AI @ Unity
Ervin Teng Ph.D, Researcher, AI @ Unity
Robin Lindh Nilsson, Co-Founder, Carry Castle

GAME DEVELOPERS CONFERENCE
MARCH 16–20, 2020 | #GDC20

Why Unity @ GDC ML Summit



Why are so many studios trying DRL?

Why are so many studios trying DRL?

Ship games faster

Why are so many studios trying DRL?

Ship games faster

Less code

Why are so many studios trying DRL?

Ship games faster

Less code

Spend less on testing

Why are so many studios trying DRL?

Ship games faster

Less code

Spend less on testing

But maintain quality!!

Why are so many studios trying DRL?

Ship games faster

Less code

Spend less on testing

But maintain quality!!

We want our free lunch /

Have our cake and eat it too /

[INSERT YOUR FAVORITE IDIOM HERE]

Why are so many studios trying DRL?

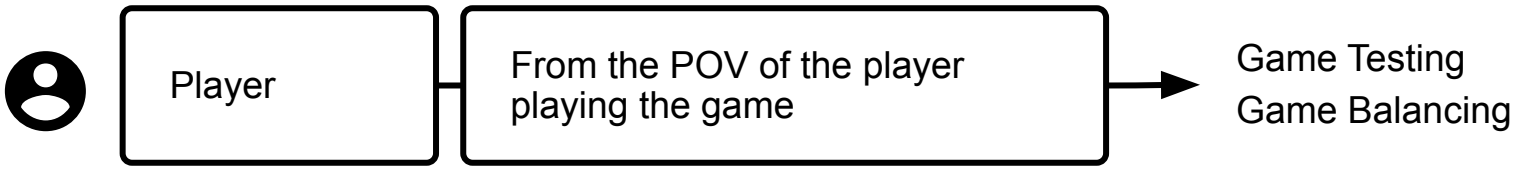
A lot of promise applying DRL in gaming....

…. But we all should be mindful of the challenges

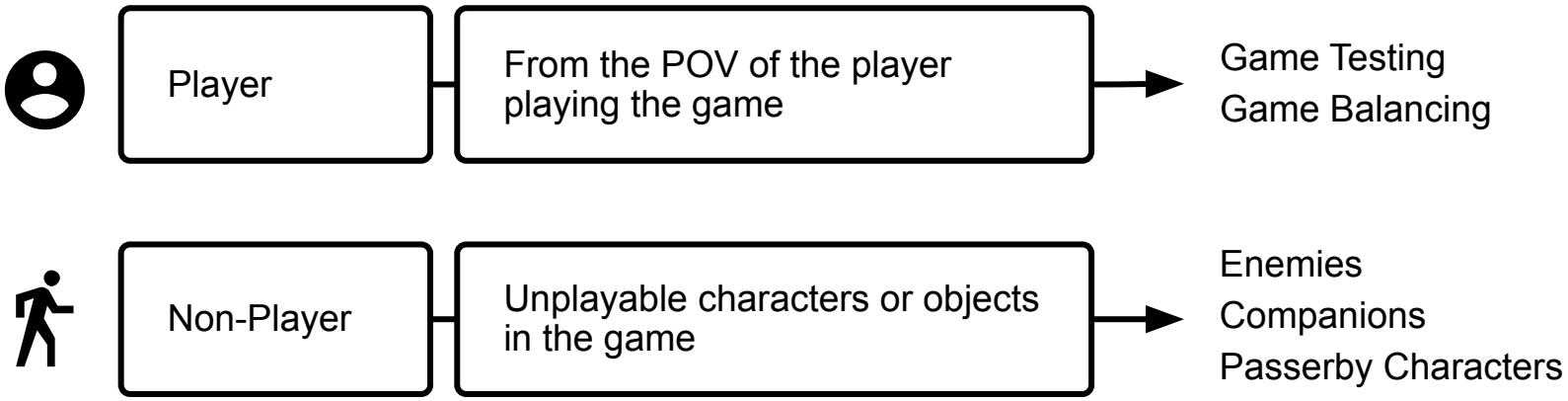
^ This is what we'll be talking about today

How are studios using DRL?

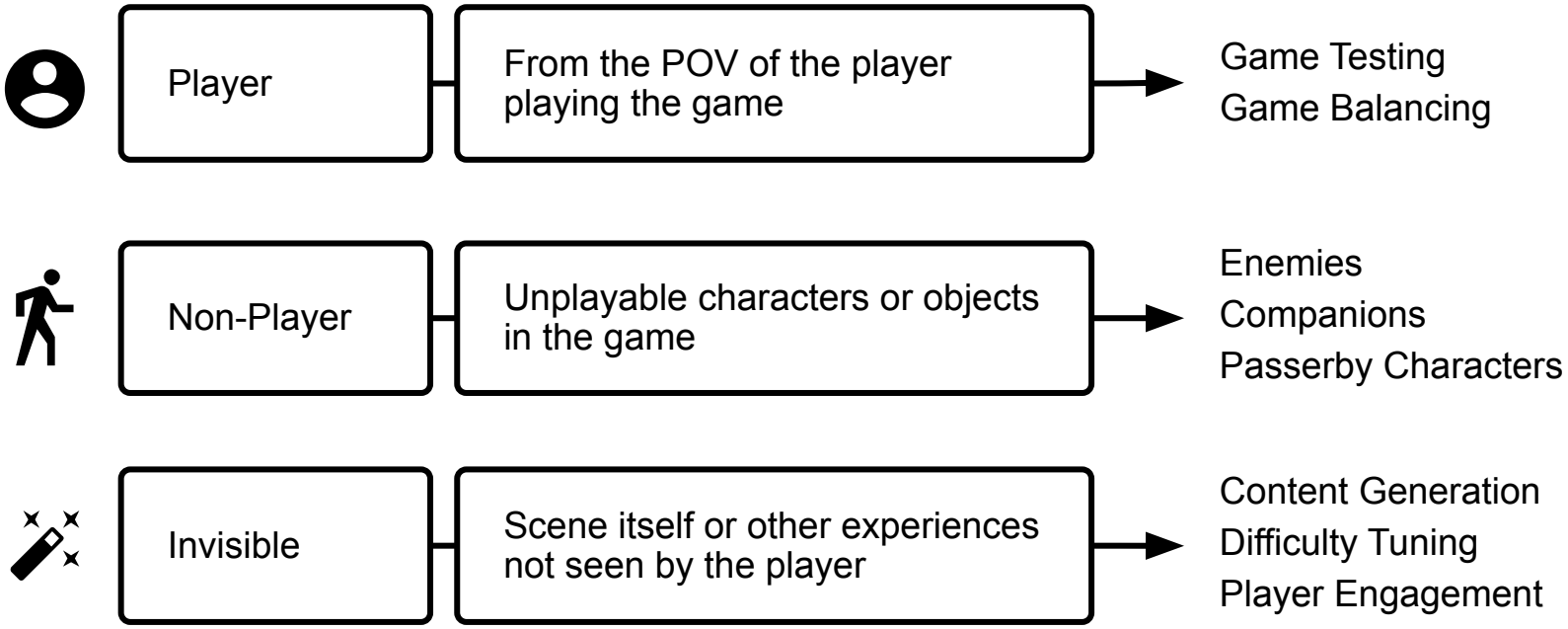
How are studios using DRL?



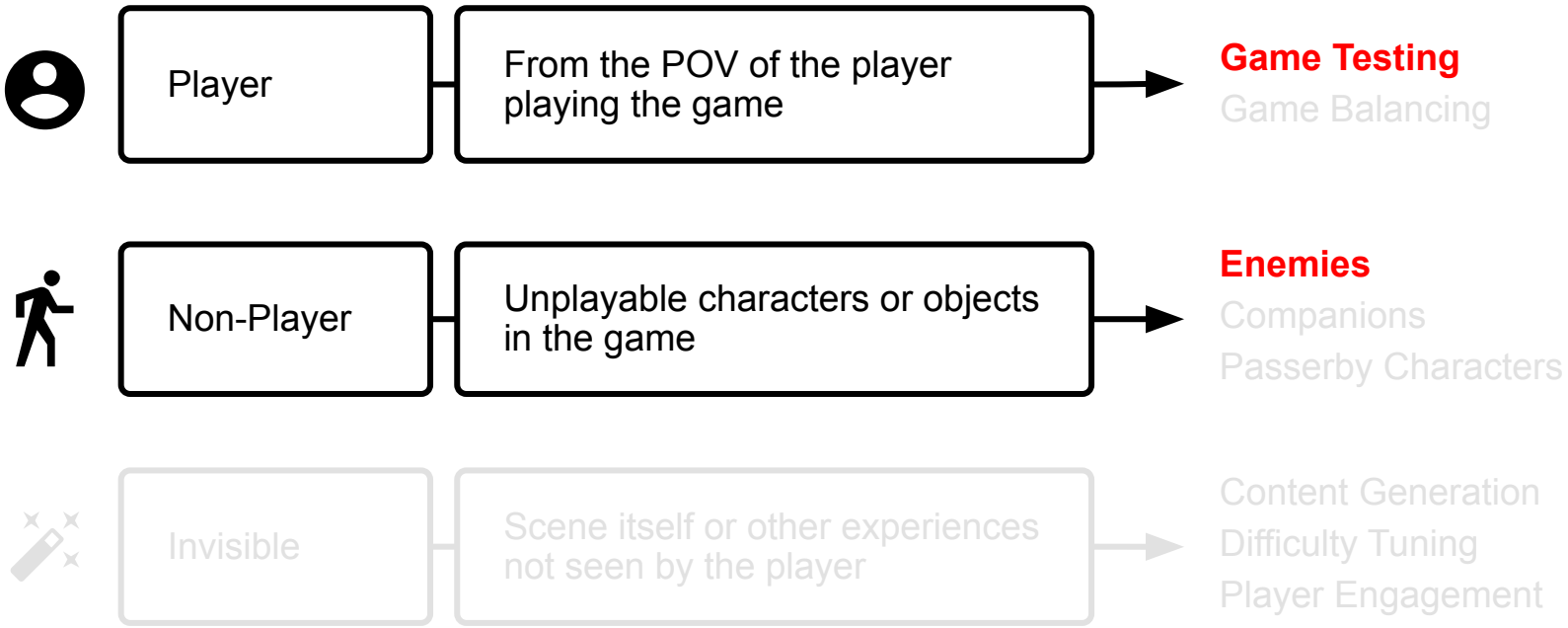
How are studios using DRL?



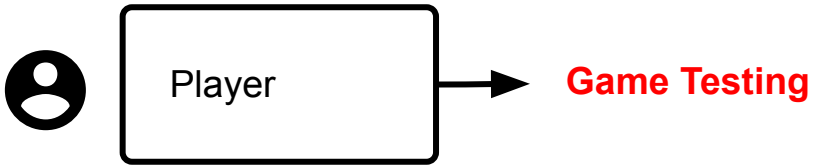
How are studios using DRL?



How are studios using DRL?

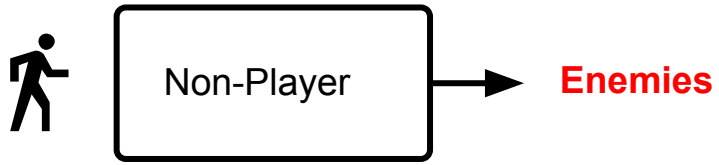


Most common use case



Test new levels or content using RL - Generalization of player bots to test never before seen levels or content

Most common use case



Natural looking enemies using RL - Make enemies looks and feel real without having to code

Today, we will focus on these common use cases

1. **Test new levels or content using RL** (Game Testing)
2. **Natural looking enemies using RL** (Enemy NPCs)

… There are many more but these are the most common in testing and NPC creation

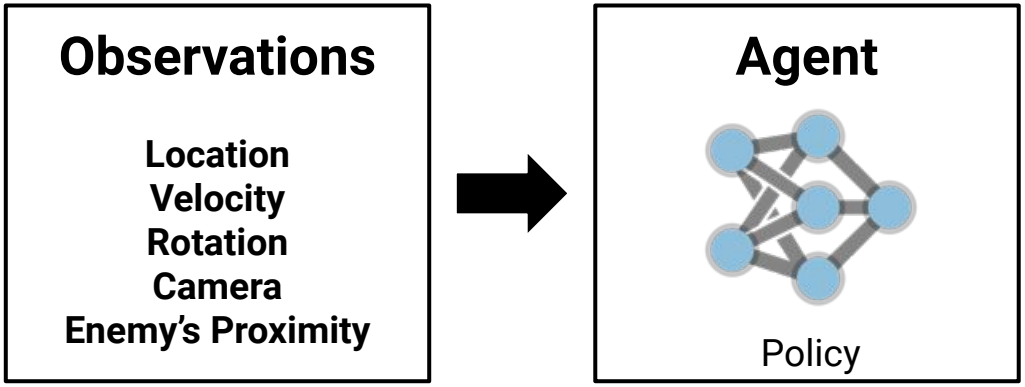
Reinforcement learning in a nutshell

Reinforcement learning in a nutshell

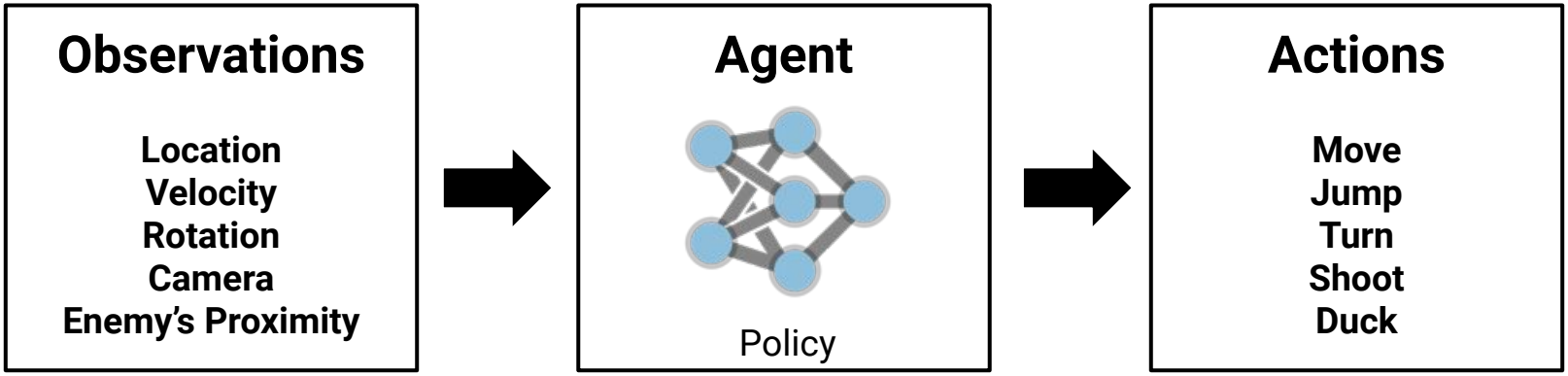
Observations

Location
Velocity
Rotation
Camera
Enemy's Proximity

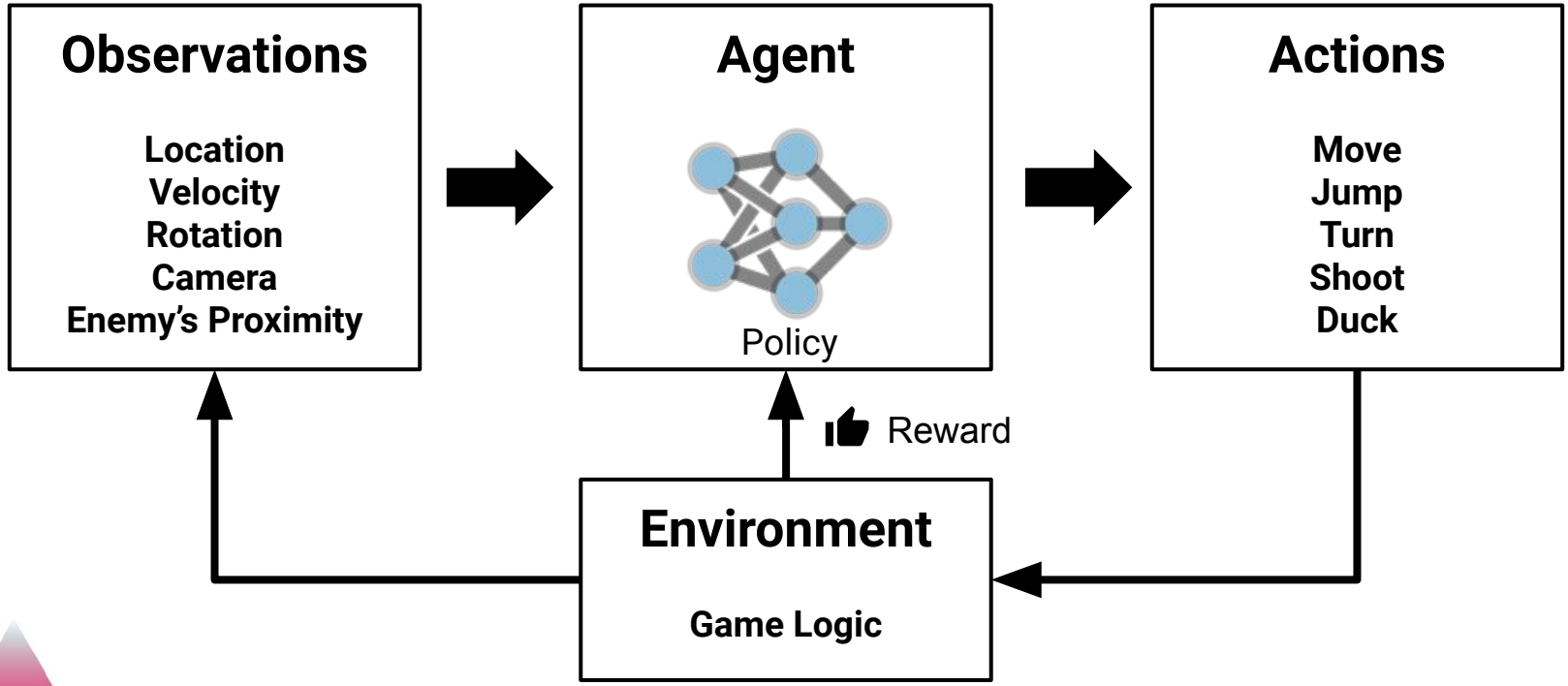
Reinforcement learning in a nutshell



Reinforcement learning in a nutshell



Reinforcement learning in a nutshell



Test new levels or content using RL

What does a testing bot need to do?

- Be able to play and win the game
- *But also*:
  - Deal with new changes for game-balancing
  - Play new, previously unseen levels to evaluate them
  - Solve levels in a *human-like* way

Example: JamCity/Snoopy Pop

Snoopy Pop is a bubble shooter game

- Levels are randomized
- Hundreds of levels
- Train a bot that can solve new levels as they are created

Example: JamCity/Snoopy Pop

What makes this hard?

Example: JamCity/Snoopy Pop

What makes this hard?

- Sheer number of procedurally generated levels
- Randomization of bubble colors

Example: JamCity/Snoopy Pop

What makes this hard?

- Sheer number of procedurally generated levels
- Randomization of bubble colors
- **Introduction of new level elements every 10-30 levels!**

Example: JamCity/Snoopy Pop

Approach 1 - Train a different agent for each level

- Too much training time **(expensive)**; evaluation of new levels not comparable

Approach 2 - Train on each level sequentially

- Agent might forget how to solve earlier levels

Approach 3 - Use player demonstrations for imitation learning

- Need player data for every level and every random bubble configuration

A few effective approaches

Domain randomization

- Encourages agent to be robust to variations in the environment

Combining demonstrations with RL

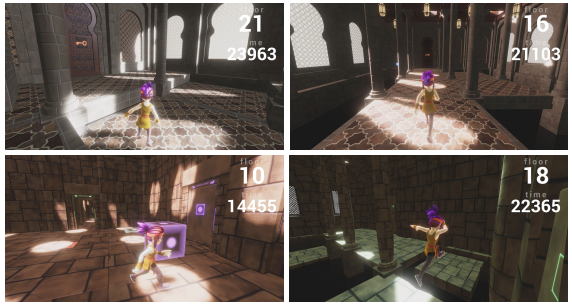
- Speeds up RL, generalizes past situations in demonstrations

Domain randomization

Variation of the Environment

Random Level Sampling

Procedural Generation



Domain randomization

Additional papers on domain randomization

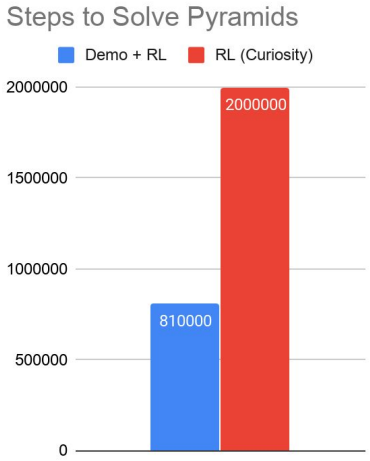
- Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World (Tobin et. al, 2017)
  - https://arxiv.org/abs/1703.06907
- Illuminating Generalization in Deep Reinforcement Learning through Procedural Level Generation (Justesen et. al, 2018)
  - https://arxiv.org/abs/1806.10729
- Obstacle Tower: A Generalization Challenge in Vision, Control, and Planning (Juliani et. al, 2019)
  - https://arxiv.org/abs/1806.10729

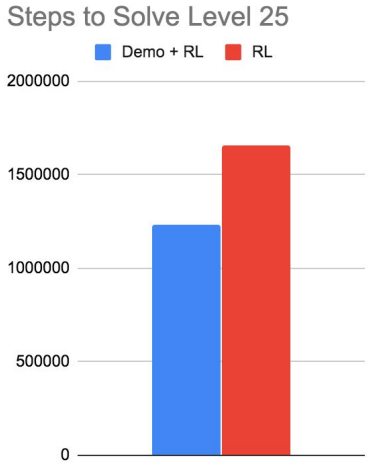
Using demonstrations to guide RL

- S*parse rewards* are less ambiguous, e.g. “win” or “lose”
  - The sparser the reward the harder to solve - i.e. *exploration problem*
  - RL is *guided random search*
- Demonstrations can be used to solve the exploration problem
  - Mix Reinforcement Learning with Imitation Learning
  - Speed up training significantly, result in more “human-like” behavior

Using demonstrations to guide RL

- Speeding up training using demonstrations





Using demonstrations to guide RL

- Solving the exploration problem: Obstacle Tower Challenge

Read more at Alex Nichol's blog: https://blog.aqnichol.com/2019/07/24/competing-in-the-obstacle-tower-challenge/

Using demonstrations to guide RL

Additional papers on using demonstrations and RL

- Overcoming Exploration in Reinforcement Learning with Demonstrations (Nair *et. al*, 2017)
  - https://arxiv.org/pdf/1709.10089.pdf
- Making Efficient Use of Demonstrations to Solve Hard Exploration Problems (Gulcehre and Paine *et. al*, 2019)
  - https://arxiv.org/pdf/1909.01387.pdf
- Imitation Learning with Concurrent Actions in 3D Games (Harmer *et. al*, 2018)
  - https://arxiv.org/abs/1803.05402
- Using Imitation Learning to Boostrap Starcraft 2 (Deepmind, 2019)
  - https://deepmind.com/blog/article/alphastar-mastering-real-time-strategy-game-starcraft-ii

How do we mitigate cost?

RL (even guided by IL) is *expensive and time-consuming*. Often requires > millions of steps

- Modern games aren't ATARI
  - Physics, complex 3D rendered scenes
- Techniques such as domain randomization are *even more expensive*
  - Increases the number of states agent experiences

How do we mitigate cost?

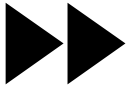
RL (even guided by IL) is *expensive and time-consuming*. Often requires > millions of steps

- Modern games aren't ATARI
  - Physics, complex 3D rendered scenes
- Techniques such as domain randomization are *even more expensive*
  - Increases the number of states agent experiences

Speedup comes in two forms

- Sample throughput
- Sample efficiency

Increasing sample throughput

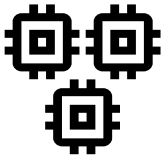


Game speedup

(+) Computation costs stay the same

(-) Limited by physics accuracy, rendering quality

(-) Can introduce unintended bugs



Parallel simulations and distributed training

(+) Does not require game modifications

(-) Proportionally increases computation costs

Increasing sample efficiency

Using demonstrations

(+) Minimal change in algorithm

(-) Must produce and record demonstrations

Using a more sample-efficient algorithm (e.g. *off-policy, such as DQN, DDPG or SAC*)

(+) Does not require any “additional work” by human

(-) May require a more complex network/more computation spent on training

Using RL for testing - final thoughts

Domain Randomization → Encourage agent to generalize across levels.

Demonstrations → Guide RL in the right direction. Can reduce training times, produce better behaviors

Long Training Times → Parallel computing, demonstrations, sample-efficient RL algorithms can decrease training times.

Carry Castle

Robin Lindh Nilsson

Engineer

Monster Trainer

Per Fornander

Technical Artist

Painter Trainer


Source
of
Madness

Source of Madness

- Action Rogue-lite
- Procedural physics world
- Procedural monsters

How to make our monsters behave correctly?

How to do this with only 3 people?

Challenges

- The controlling physics of monsters is very unique
- Millions of variations of procedurally generated monsters
- Needs to look natural
- Less coding footprint and performance considerations
- Only 3 people

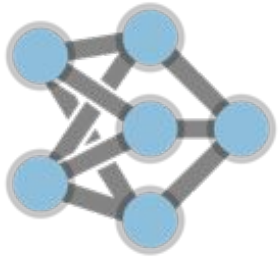
Lots of different variations and joints

So we turned to RL as a solution

RL setup for Source of Madness

Observations

Nearby environment
State of own body



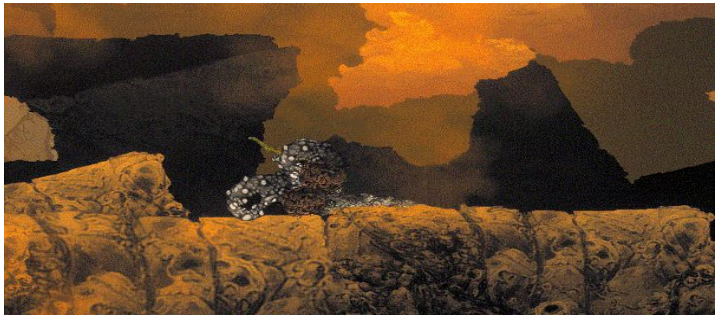
Actions

Muscle forces
Grabbing

Reward = Speed towards target + Extra Reward(s)

Structuring the proper rewards

- Overall, the goal was to move toward the player
- But we gave them smaller extra reward
  - “Stay close to the ground” to learn **tumbling**
  - “Stay above ground” to learn **jumping**
- Much better variation in less time



Tumbling example

Jumping example

RL setup for Source of Madness

- Different body structures
- Various ways to walk

Quadruped

Spiders

Lessons learned

- WHY does it not work?
  - A lot of potential reasons
  - Don’t start guessing randomly

Lessons learned

- Common pitfalls in games:
  - Reward function
  - Handling of Actions

Coded by hand = Easy to make mistakes

Check that they’re working as intended!

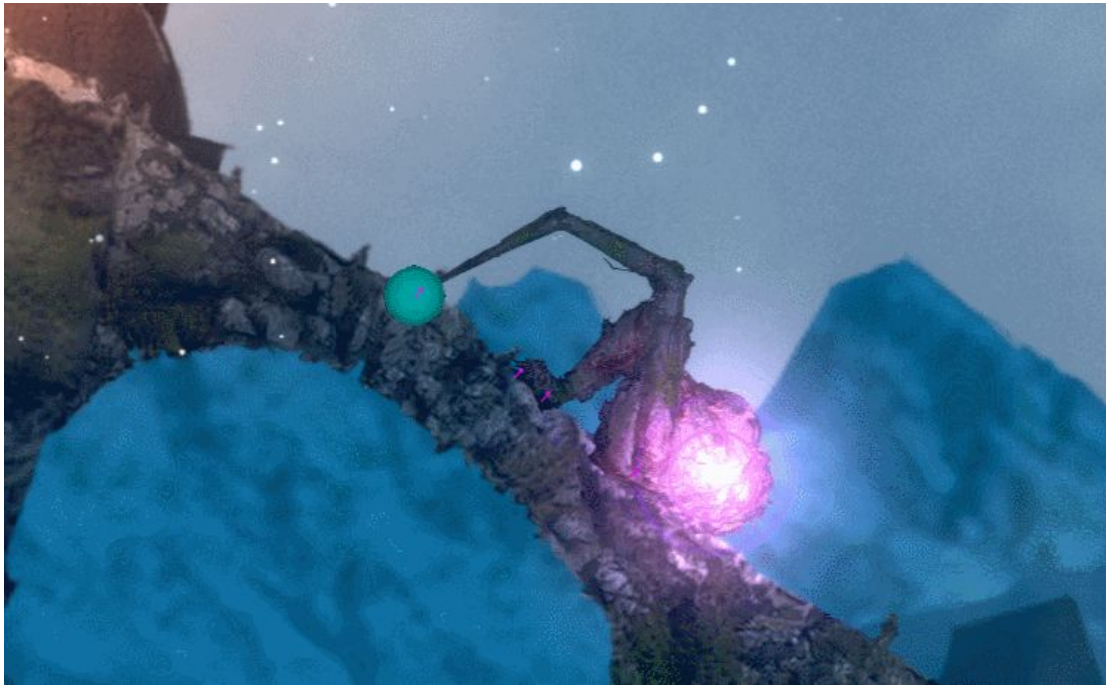
Lessons learned

- Otherwise you'll waste hours
- Takes time to find the mistake
- Might even work fairly well, but not optimal

Lessons Learned - Visualizations

Lessons Learned - Balancing Rewards

Lessons Learned - Handling of Actions

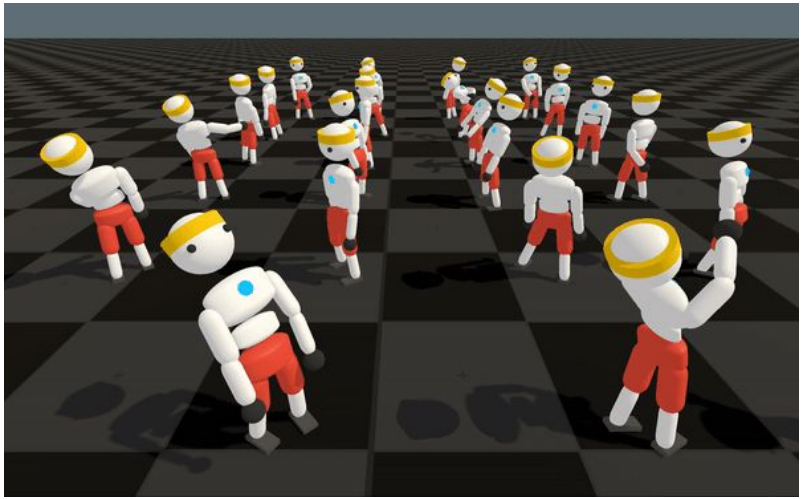


sourceofmadness.com

Robin Lindh Nilsson

robin@carrycastle.se

Thank you!



Additional Information
(Point Camera)

tinyurl.com/unitygdcml2020

Getting started (WIP)

- What can you do to get started “starter kit”

GRAVEYARD SLIDES

[Optional] Case Study II - Using Multi-agent, self play for iLLOGIKA Rogue Racers

[Optional] Case Study III - Snoopy Pop and RL, how to think about RL for abstracting to new levels

[Optional] Case Study IV - “Bigger game”

How does it compliment the first case? (is it just a repeat?)

How it’s deployed, implementation?

TBD - need to figure out if we can use.

GRAVEYARD SLIDES

Why is Unity at the ML Summit

Successfully Use Deep Reinforcement Learning in Testing and NPC Development

Jeffrey Shih, Lead PM AI @ Unity
Ervin Teng, Ph.D, Research AI @ Unity
Robin Lindh Nilsson, Co-Founder Carry Castle

Introductions

Jeff & Ervin from Unity. Joined by Robin at Carry Castle

Why is this talk important: Pressure to deliver games on time, new experiences expected from gamers.

We know of the problems. Hear from Ervin,

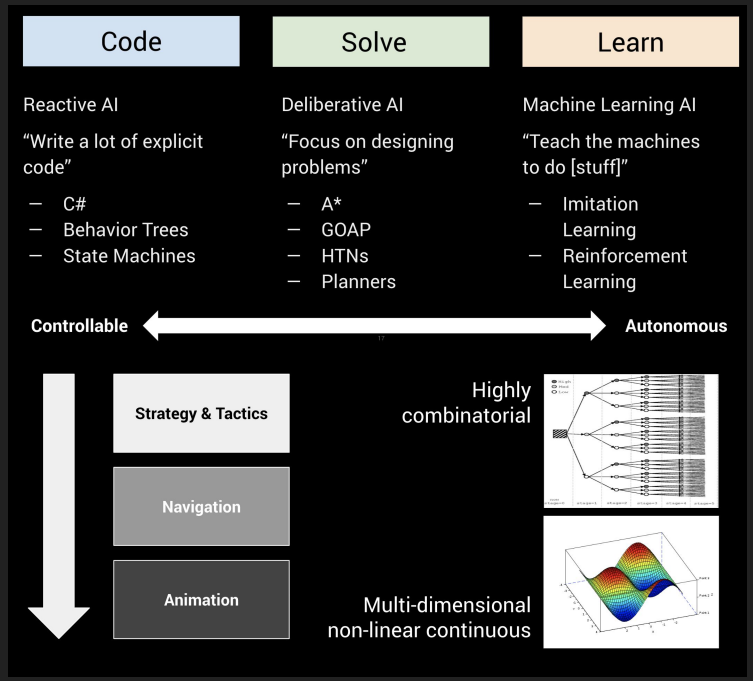
Placeholder

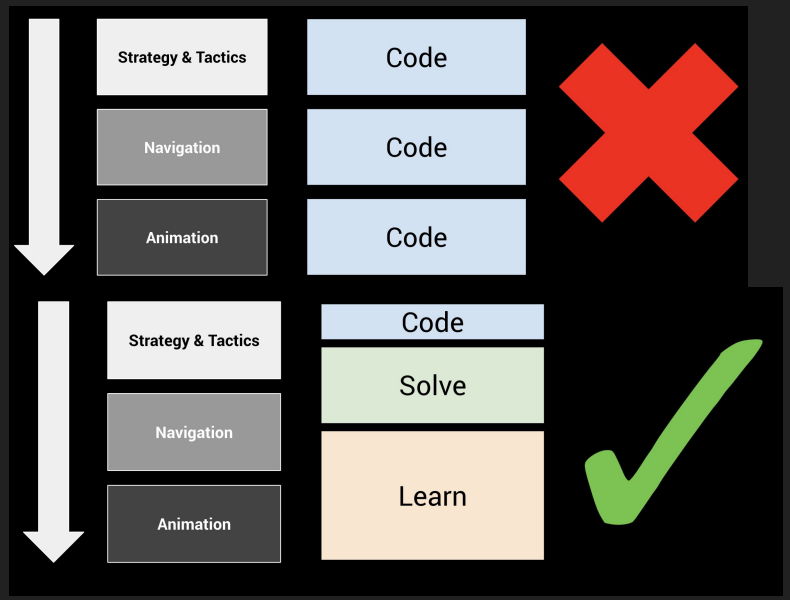
Talk about why Unity is at ML Summit. Because we see across

Olivier Opening / Introductions: 1. Unity at ML Summit is going provide some transverse / wider view of AI in games. All companies (small games, large game companies). Then go to “why is this important”.

Existing solutions and approaches

Single slide to illustrate this point





Using Deep Learning

Transition to Ervin

What is RL / What is IL - in one slide. setup slides to give overview of both techniques (perhaps cut this)

More technical - More on the RL side. Pick an Axis so we can anchor to it.

Message as IL to make RL more efficient (because RL takes a ton of time for exploration).

Make sure to pick the angle - Applying this to real games, what do we see. Use our examples (iLLOGIKA, Carry Castle). Finish with “is this solved, not solved” what else we need to do.

What can you do with RL / IL - The World Today

Testing is done manually and through expensive soft launch

Game AI’s take a lot of programming effort and iterations

What can you do with RL / IL - The World Tomorrow

Before putting the game out in the real world, understand if it works and is enjoyable without spending a dollar

- Testing Bots - Enables smaller budget studios to perform thorough testing
  - Cost-effective solution for playtesting
  - Create distribution of different player behaviors

New kinds of game experiences we haven’t even thought of (that is outside the bounds of normal programming)

- NPC - Enables smaller studios to achieve much more in their Game’s AI
  - Less coding, more experimenting
  - Richer player experiences

What we see from studios trying RL / IL

A Unity, we see the horizontal. We see studios ranging from FPS to Match 3.

The difference between IL bot vs RL bot

- RL tries find the best and most optimal way to achieve a reward. Therefore, tend to get “super-human” abilities
  - See it used a lot for QA, functional, and stress testing
- IL tries to mimic the expert. Therefore, tend to get “human-like” abilities
  - See it used a lot for general playtesting

Common Challenges across all studios

Implementing RL

Implementing IL

[LINK TO DOC, Long List] - https://docs.google.com/document/d/1yH22GZTQKiBEJnFHD-W5C36qkPgSMcJpkHO145En-tU/edit

Why are studios doing this - the business case

Everyone is trying to ship the games faster, spend less money on testing but maintain quality.

Testing automation using RL (Ervin)

- Current solutions usually break or do not work well when new levels or content is introduced
- Studios require a wide breadth of different player profiles for testing

NPC development (Robin)

- In some cases, a developer cannot actually code all the physics and animations of NPCs (Carry Castle)
- Matchmaking can be challenging, you need compelling opponent bots (TBD)

Coding everything is not good business (reference the 3 learn, code, solve)

Reference world today / world tomorrow in doc

What we will be talking about today

Based on what we see, here is some the most common use cases. (Emphasize there a LOT of other use cases)

Testing automation using RL (Ervin)

- Generalization - see new content that previously unseen
- Varying patterns of behaviors - degree of skill (from novice to super human)

NPC development (Robin) (alot of use cases, but the one thats “slam dunk”)

- Animations to look “real” (Carry Castle)
- Player bots for matchmaking in a multiplayer game (TBD)

What we will be talking about today

What have we learned in these areas

If you are planning to do RL in games, make sure you consider these things

STRUCTURE OF REMAINING CONTENT

	Best practices, techniques approaches. What needs to be in place (AI env)	State of research (citations, paper)	Common challenges studios face	Examples with studios
Generalized bots			Across all applications	
Varying skill bots				
Behaviors / ties into gameplay (physically animated, etc..) for NPC				
Player bots for				

End this with summary

Example: JamCity/Snoopy Pop

What makes this hard?

- Hundreds of levels
- Randomization of bubble colors
- Introduction of new level elements

Example: JamCity/Snoopy Pop

What makes this hard?

- Hundreds of levels
- Randomization of bubble colors
- Introduction of new level elements

RL in Testing: Bots Robust to Game Changes

How can we create bots that are robust to new content (e.g., play through new levels)?

- Scripted/programmed bots are specific to the level they were created for

An RL Solution

- Domain randomization to train adaptable agents
- In games - procedural generation or random level selection
  - Generalization through procedural levels - https://arxiv.org/pdf/1806.10729.pdf
  - Adaptability through domain randomization - https://openai.com/blog/solving-rubiks-cube/

RL in Testing: Bots that Play Like Humans

How can we create bots that play like human players at different skill levels?

- Scripted/programmed bots - must be handcrafted per skill level
- Pure RL bots will try to be the “best”

An RL Solution

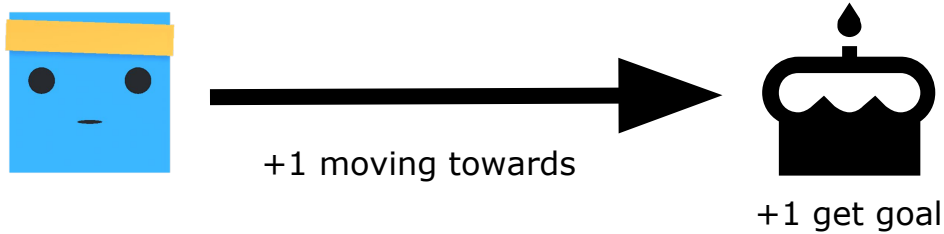
- Imitation learning (with or without reinforcement learning)
- In games - player data is often readily accessible
  - IL to evaluate level difficulty
    https://medium.com/techking/human-like-playtesting-with-deep-learning-92adafffe921
  - IL to train a baseline, RL to improve past it -
    https://deepmind.com/blog/article/alphastar-mastering-real-time-strategy-game-starcraft-ii

Game and Level Robustness

- Training requires 1000’s+ of playthroughs
- Agents are very good at finding exploits!

Observation, Action, and Reward Definitions

- Rewards often contain local minima, agent will exploit!
  - “Dense” rewards - easier to learn, more likely to have minima
  - “Sparse” rewards - harder to learn, more reflective of actual goal

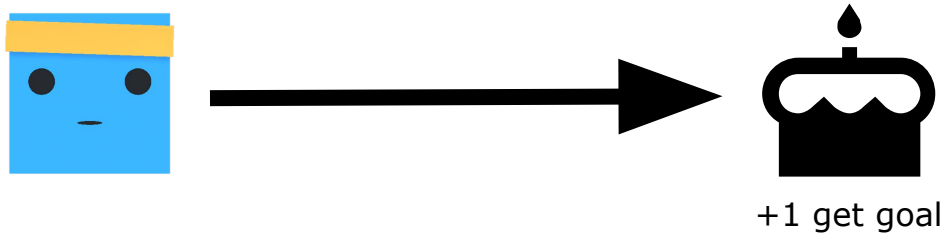


Observation, Action, and Reward Definitions

- Rewards often contain local minima, agent will exploit!
  - “Dense” rewards - easier to learn, more likely to have minima
  - “Sparse” rewards - harder to learn, more reflective of actual goal

Observation, Action, and Reward Definitions

- Rewards often contain local minima, agent will exploit!
  - “Dense” rewards - easier to learn, more likely to have minima
  - “Sparse” rewards - harder to learn, more reflective of actual goal



Observation, Action, and Reward Definitions

- Rewards often contain local minima, agent will exploit!
  - “Dense” rewards - easier to learn, more likely to have minima
  - “Sparse” rewards - harder to learn, more reflective of actual goal

+1 get goal

- Visual observations - useful for research, less useful for games

Long Iteration Times

- Agents take *hundreds of thousands* to *millions* of actions to learn
  - Trial and error difficult
  - Lots of hyperparameters to tweak

Long Iteration Times

- Agents take *hundreds of thousands* to *millions* of actions to learn
  - Trial and error difficult
  - Lots of hyperparameters to tweak

- Two ways to improve training time:

Sample Throughput

- Game speedup (be careful of physics!)
- Multiple copies of the game
- Distributed training across machines

Sample Efficiency

- Sample-efficient RL algorithms (off-policy)
- Imitation learning

Case Study I - How an indie studio (Carry Castle) is leveraging IL and RL (Action Rogue-lite)

Transition to Robin

1. Introduction of Carry Castle and Source of Madness
2. Why Machine Learning
3. Inspiration! Show cool monsters, trained with different rewards
4. Challenge: Balancing the reward function
    a. Show gif of real-time reward visualization
5. Challenge: Combining ML with coded logic
    a. Ignore ML on some muscles while the limb is attacking. Control attack with script.

CARRY CASTLE

Robin Lindh Nilsson

Engineering

Monster Trainer

Per Fornander

Technical Artist

Painter Trainer

Source of Madness

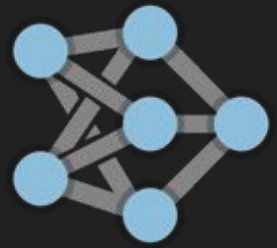
- Action Rogue-lite
- Procedural physics world
- Millions of procedural monsters

How to do this with
only 3 people?

Machine Learning Monsters

Observations

Nearby environment
State of own body



Reinforcement Learning

Actions

Muscle forces
Grabbing

Reward = Speed towards target + Extra Reward(s)

Machine Learning!

Reward close to ground

Reward above ground

Lessons Learned

- WHY does it not work?
- Common pitfalls in games:
  - Reward function
  - Handling of Actions
  - ~~Observations~~ Maybe include something around 2 person studio or skip it (easy to sit and watch the monsters). Better to use gizmo instead of “visual”. Unintended examples maybe?

Visualizations!

Balancing Rewards

[Multi-reward gizmo improved gif]

Handling of Actions

[GIF: gizmo for grabbing limbs]

Summary

What can you do to get started “starter kit”

Overview of best practices

Delete?? Natural looking enemies using RL

- Carry Castle - we are a three person studio!
- For us, we need solutions for our NPCs since we don’t have all the resources of a bigger studio
- We want our NPCs to look natural and fit well in our game

SLIDE OUTLINES

TODO: Finalize on outline

TODO: Create material

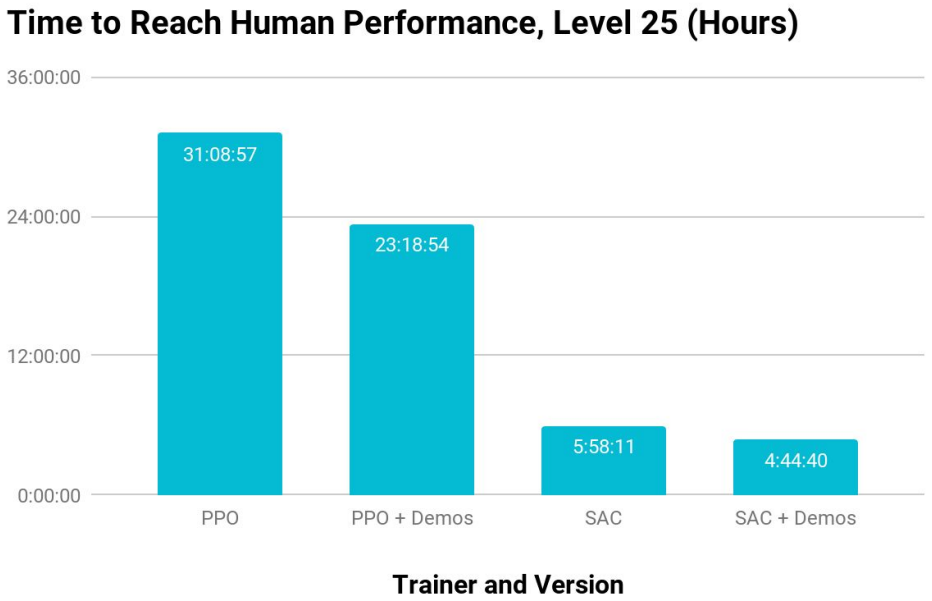
TODO: Polish / Move to GDC Template / Create QR Code and Doc

TODO: Rehearse

Timing: 27 minutes

Audience - might not be the most up to date, more game devs

Improving sample efficiency



Read more here: https://blogs.unity3d.com/2019/11/11/training-your-agents-7-times-faster-with-ml-agents/