Creating Cooperative Character Behaviors using Deep Reinforcement Learning

Vincent-Pierre Berges - Senior Research Engineer, Unity Technologies
Markus Weiß - Co-Founder, Couch in the Woods

Contents

- Introduction
  - Why create cooperative behaviors
  - Reinforcement learning overview
  - Why use reinforcement learning in games
- How to use reinforcement learning for cooperative behaviors in games
  - Centralized learning, decentralized execution
  - Asynchronous decision making
  - Variable number of characters
  - Optimizing for the greater good
- Couch in the Wood presentation on how they used RL in NEON SHIFTER

Why Cooperating Characters?

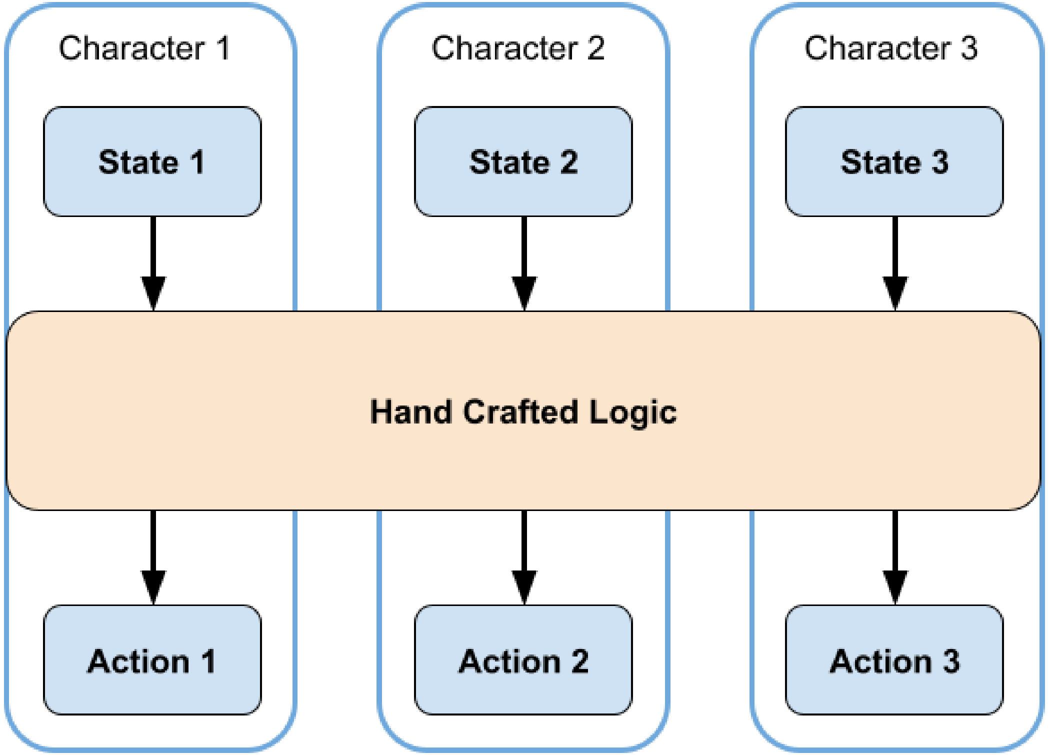
- Improve realism
- Create interesting gameplay
- Replace a player in an online game

Cooperative Behaviors: A Hard Problem

- The complexity of the behavior increases
  - With the number of characters
  - With the degree of cooperation between characters

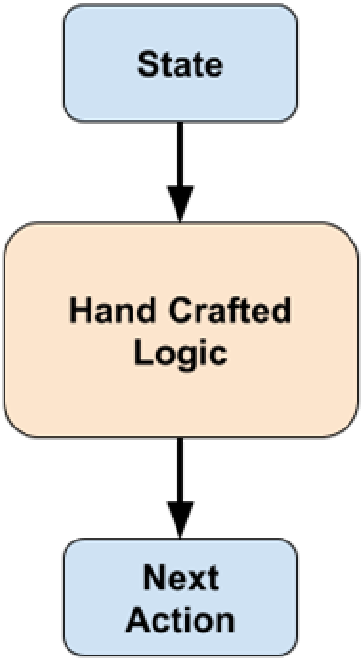
- 2 players - Co-op mode is same as single player mode	<	- 4 players - One player cannot finish the game alone

Cooperative Behaviors: A Hard Problem

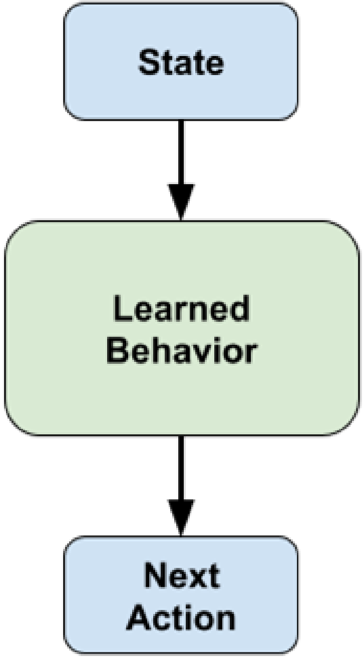


Deep RL as an Alternative to “Traditional” AI

“Traditional” AI



Deep Reinforcement Learning



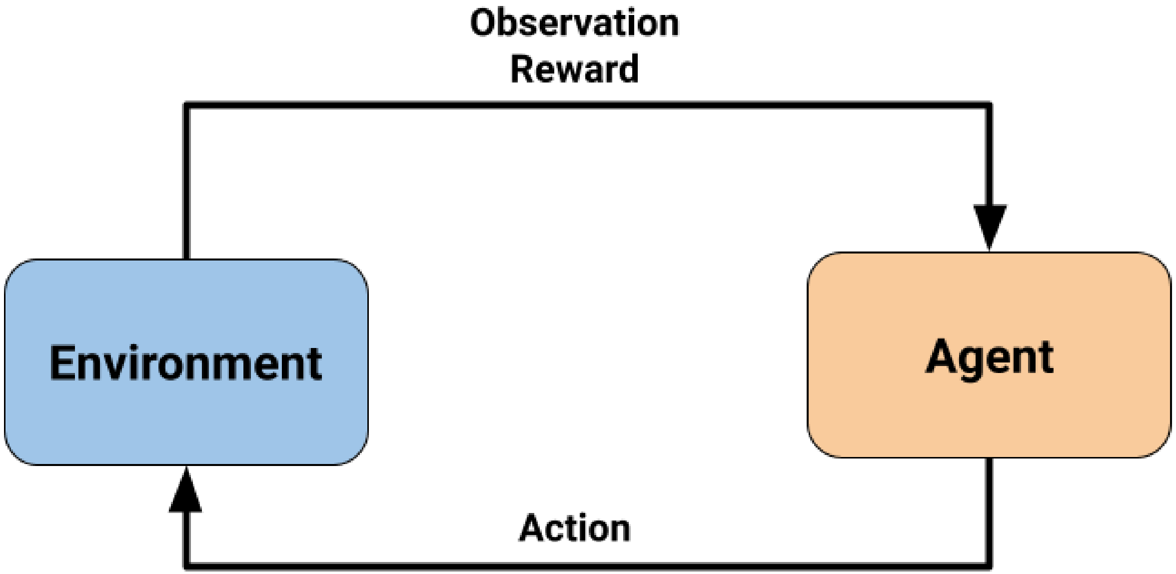
Deep Reinforcement Learning to Solve Games

A few examples

- Atari 2600 Games - DeepMind, OpenAI
- Doom - VizDoom from Poznan University
- Quake 3 - DeepMind
- Minecraft - Microsoft Project Malmo
- Starcraft 2 - DeepMind / Blizzard
- Dota 2 - OpenAI Five


unity

Deep Reinforcement Learning



Learn the mapping from observations to actions that will **maximize cumulative reward**

Advantages of Deep Reinforcement Learning

- Saves development time
- Not fragile, robust to most game design changes
- Updating agents after large game changes is easy
- Write less code
- Agent can learn complicated behaviors impossible to specify by hand
- Agent can learn to imitate a human

Types of Character Behaviors

- Single / Individual (Breakout)
  - Each character maximizes individual reward

- Competitive (Chess)
  - The character needs to beat another character
  - Self-Play allows the character to get better against past behaviors

- **Cooperative** (Overcooked)
  - A group of characters works together to accomplish a goal
  - Multi-Agent Reinforcement Learning (MARL) are algorithms used to solve cooperative behaviors

Reinforcement Learning for Cooperative Behaviors

- With Multi-Agent Reinforcement Learning (MARL)
  - Individual characters will learn how to optimize group behavior
  - The burden of complexity is on the algorithm, not the developer

Reinforcement Learning for Cooperative Behaviors

- Centralized learning, decentralized execution
- Asynchronous decision making
- Variable number of characters
- Optimizing for the greater good

Reinforcement Learning for Cooperative Behaviors

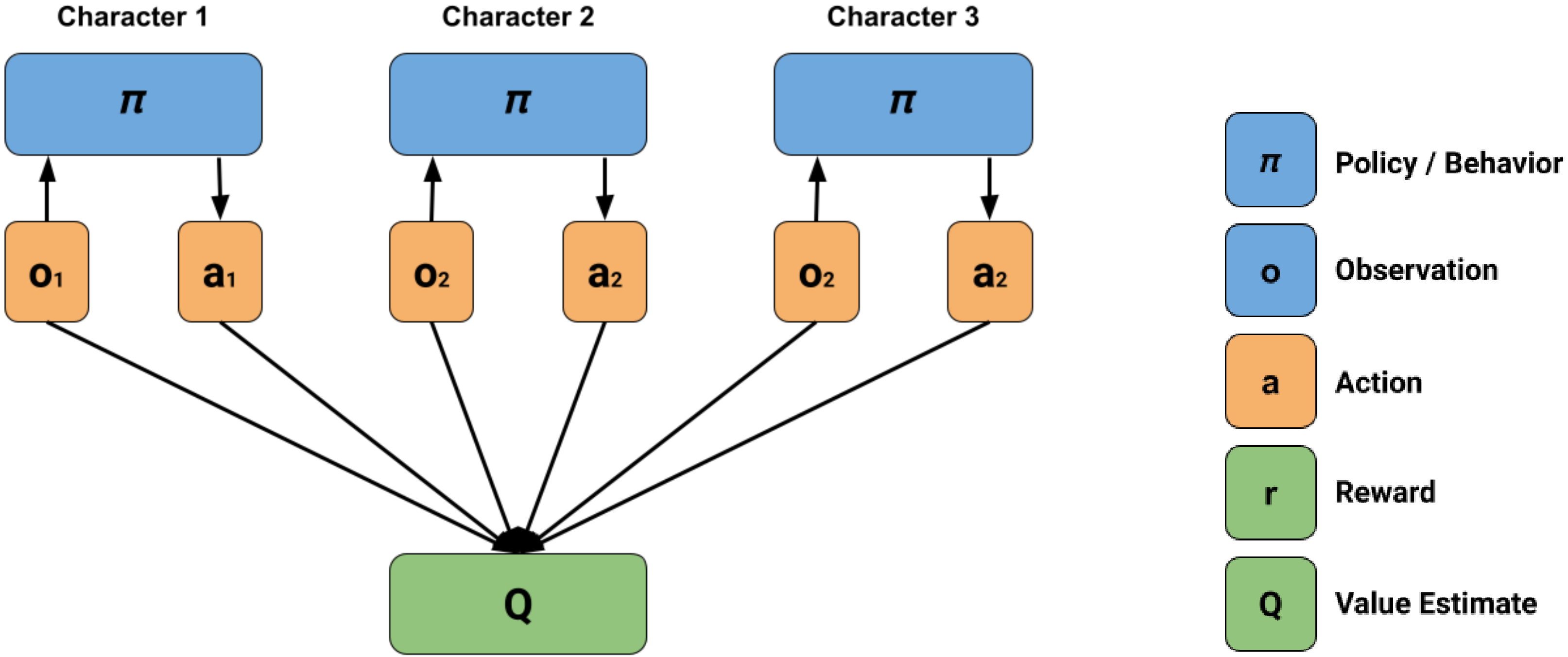
- **Centralized learning, decentralized execution**
- Asynchronous decision making
- Variable number of characters
- Optimizing for the greater good

Centralized Learning

- Centralized Critic allows characters to assign value to the states and actions of their teammates, not just themselves
- Example of algorithm using Centralized Critic : MADDPG

Without Centralized Critic:
This character is waiting for teammates to score because it will get a reward anyway

Centralized Learning



Example: Push the Blocks

Observations:

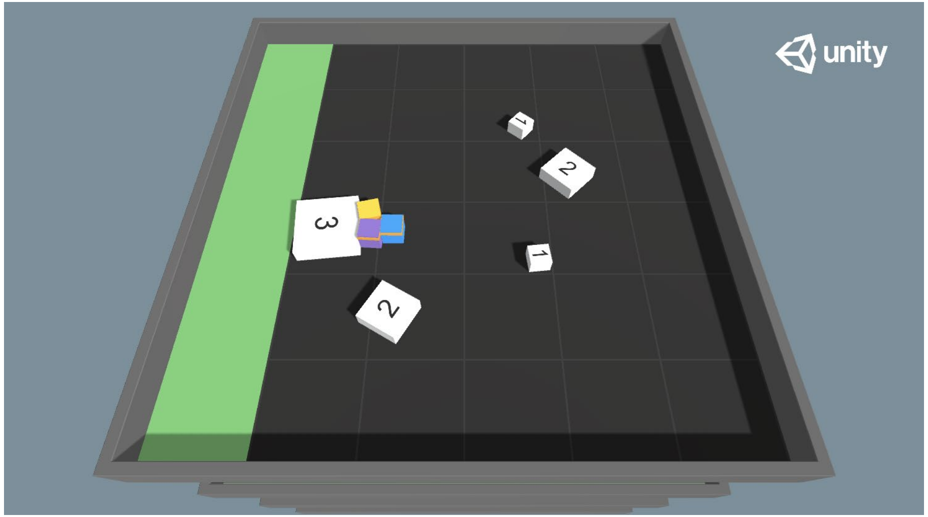
Raycasts

Actions:

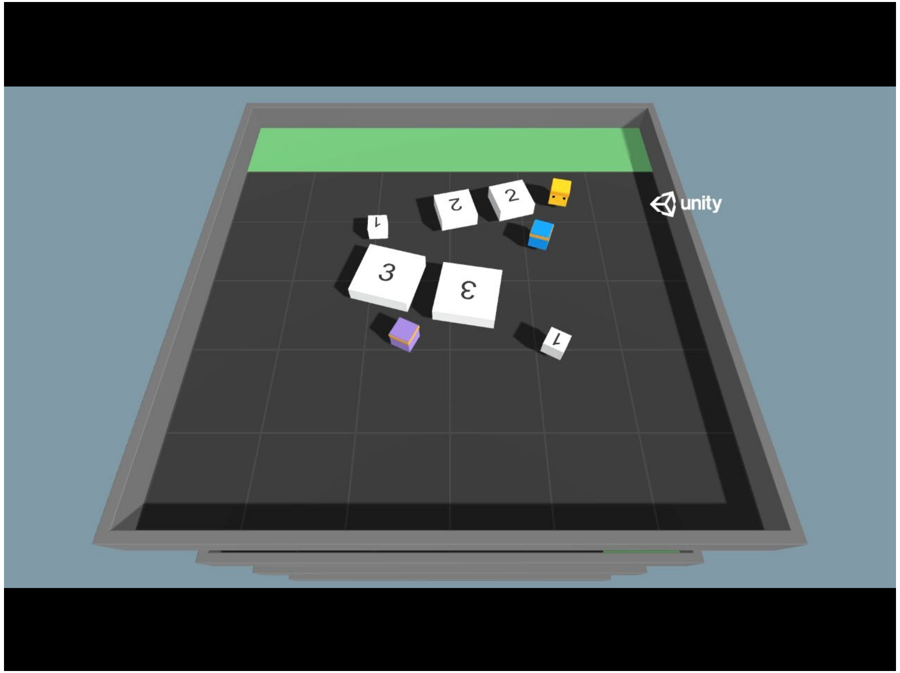
Move and rotate

Objective:

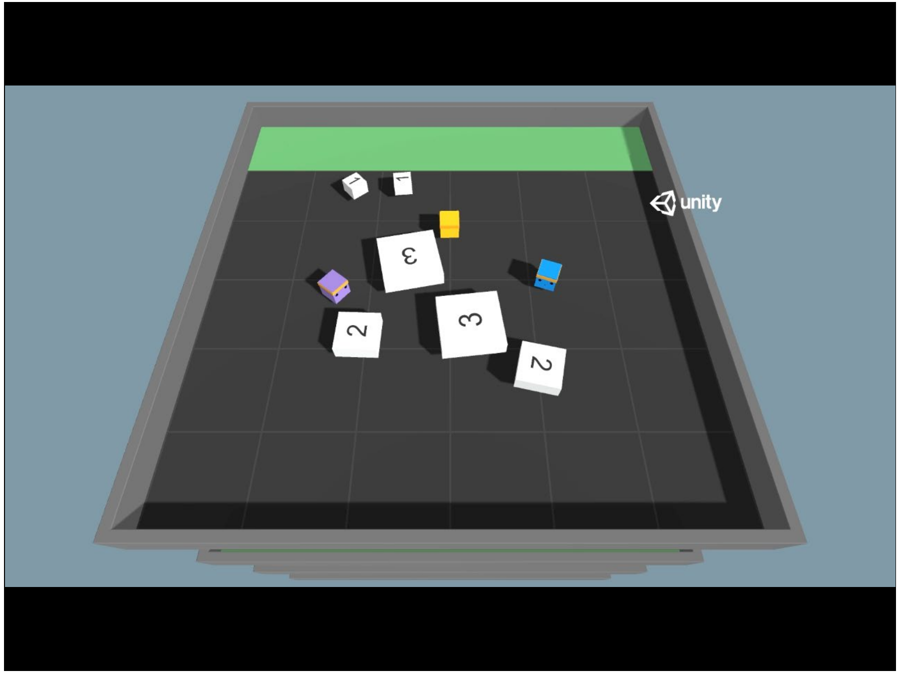
Push blocks into green zone



Centralized Learning



Greedy Solution



Centralized Critic

Using MARL in Real Games

- Most MARL algorithms are not applicable to games:
    - Characters must have their decision points at the same time
    - Limited to a constant number of characters
    - Characters are averse to “game over”

Reinforcement Learning for Cooperative Behaviors

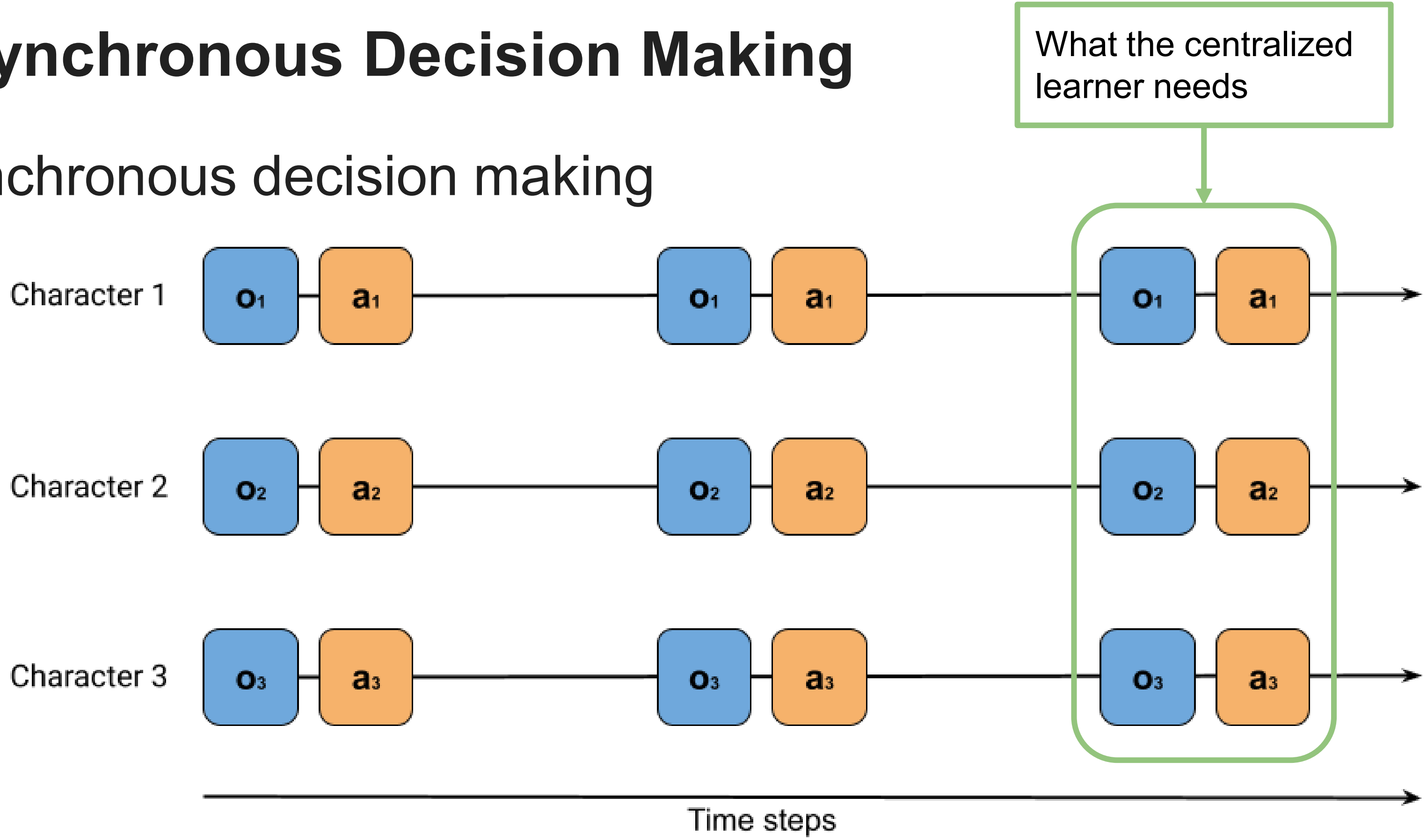
- Centralized learning, decentralized execution
- **Asynchronous decision making**
- Variable number of characters
- Optimizing for the greater good

Asynchronous Decision Making

- Problem: Most MARL algorithms are hard to use in games because characters must have their decision points at the same time
- Solution: Record latest state of each character to allow characters to make decisions anytime, even if other characters do not

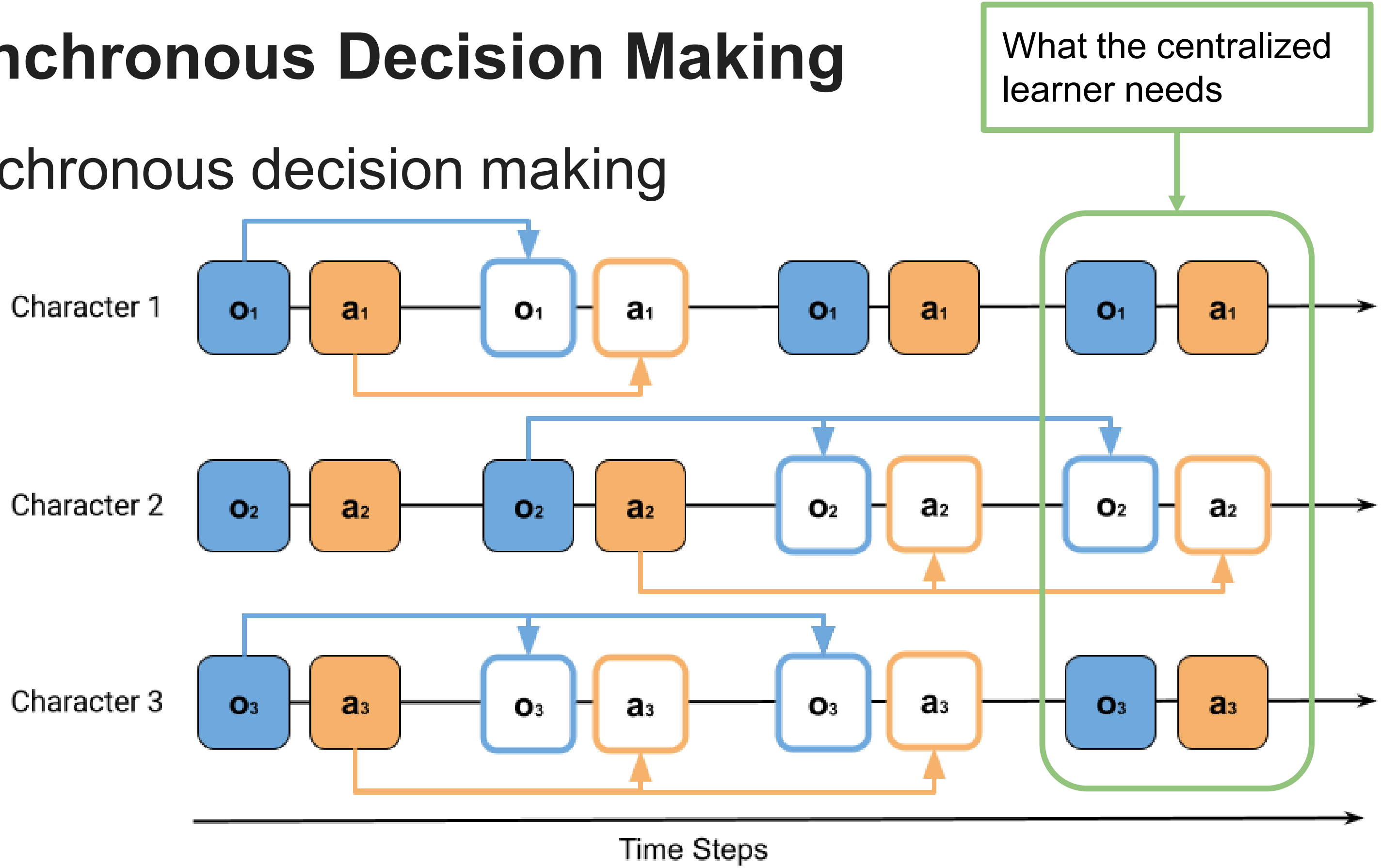
Asynchronous Decision Making

Synchronous decision making



Asynchronous Decision Making

Asynchronous decision making



Reinforcement Learning for Cooperative Behaviors

- Centralized learning, decentralized execution
- Asynchronous decision making
- **Variable number of characters**
- Optimizing for the greater good

Varying Number of Characters

- Problem: Most MARL algorithms are hard to use in games because limited to a constant number of characters
- Solution: Use attention modules to deal with varying number of characters

Varying Number of Characters

- Scaled Dot-Product Attention

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

- **Q**, **K** and **V** are encodings of observations or actions
- Attention can process a variable number of inputs

Example: DodgeBall

Observations:

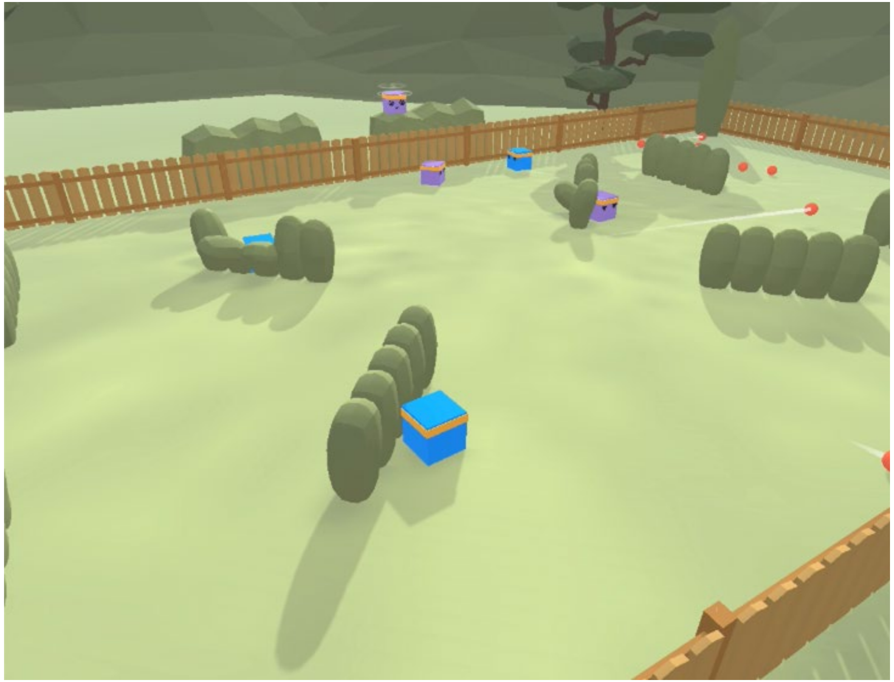
Raycasts

Actions:

Move, rotate and shoot

Objective:

Eliminate all opponents




y
x
z
Persp

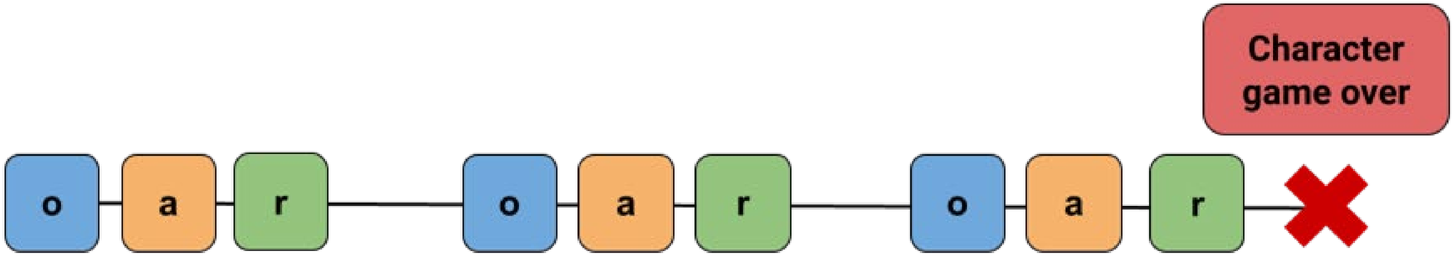
Reinforcement Learning for Cooperative Behaviors

- Centralized learning, decentralized execution
- Asynchronous decision making
- Variable number of characters
- **Optimizing for the greater good**

Optimizing for the Greater Good

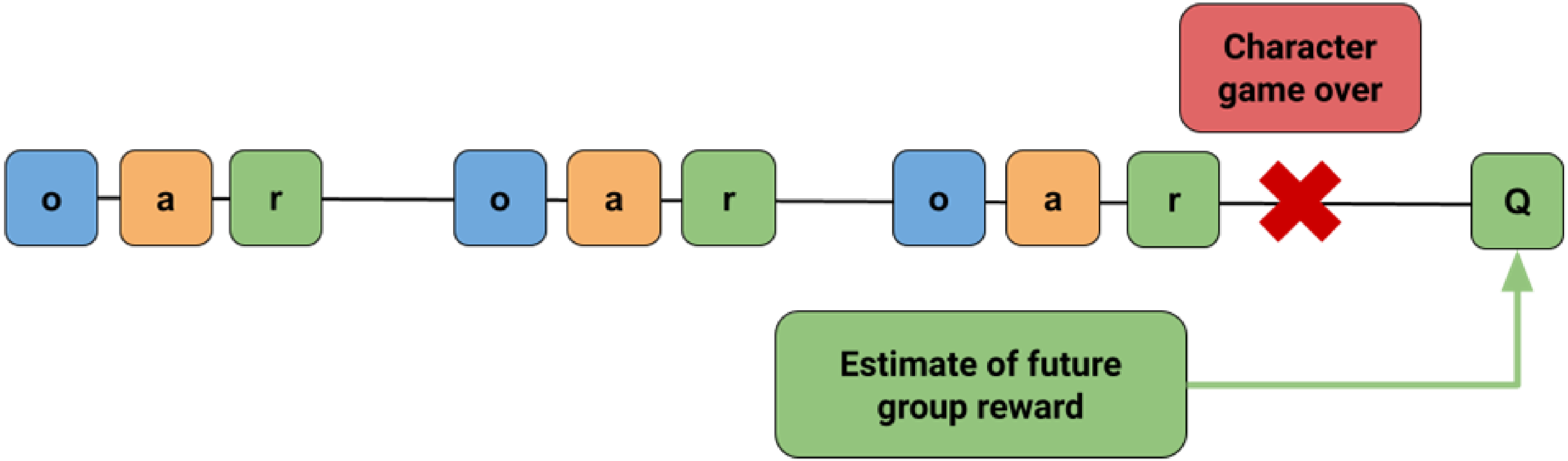
- Problem: Most MARL algorithms bias characters towards survival
- Solution: Each character needs to be rewarded depending on what its death accomplished for the group
- In real games, characters will need to sacrifice themselves for the good of the team

Optimizing for the Greater Good



Optimizing for the Greater Good

- Attention enables the same critic (Q function) to be shared even as agents are removed from the game (e.g. elimination, time out, etc.)



Example: Escape the Dungeon

Observations:

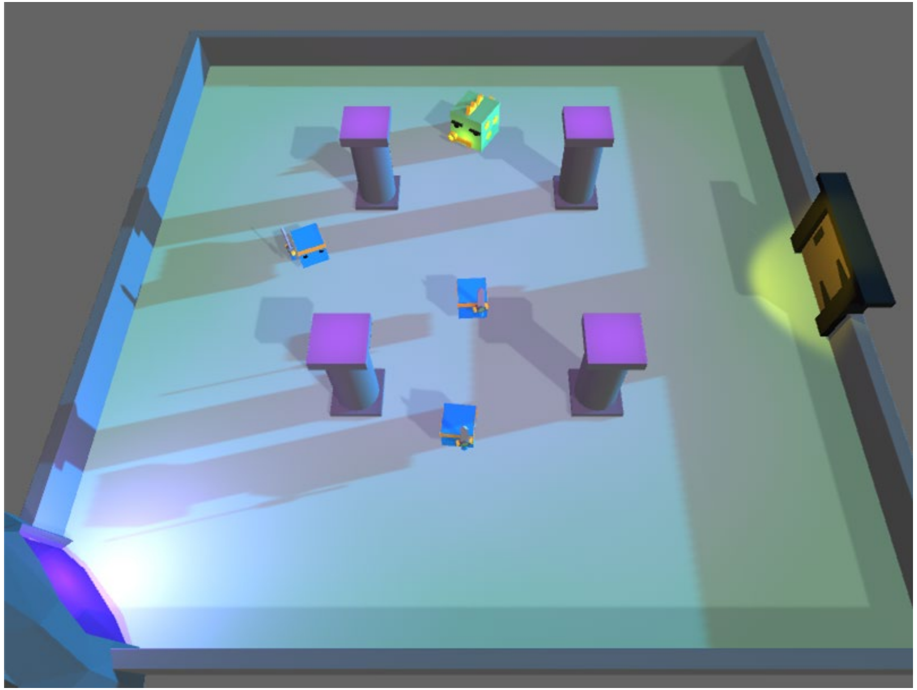
Raycasts

Actions:

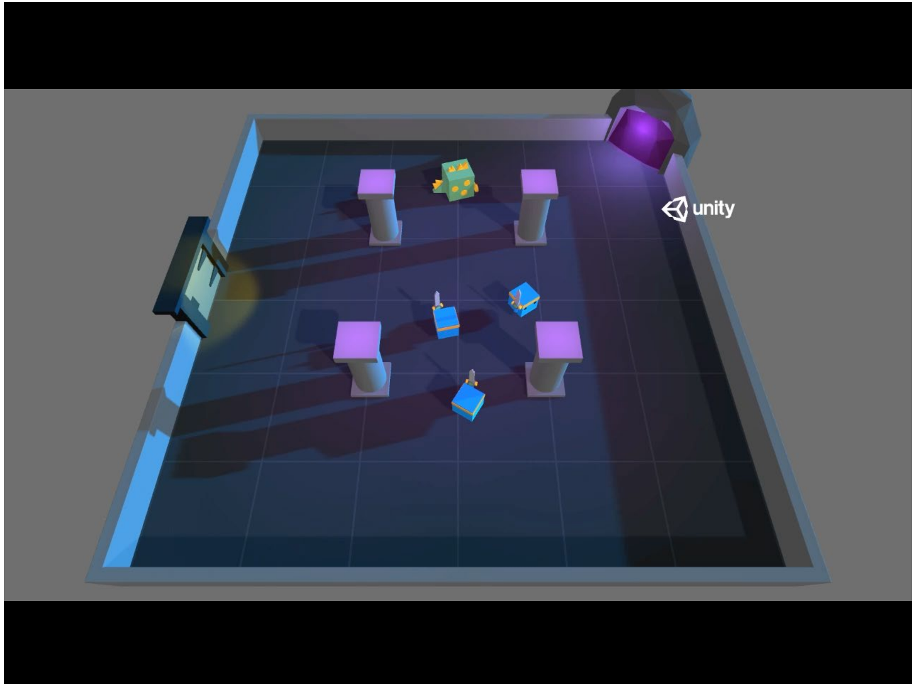
Move, rotate

Objective:

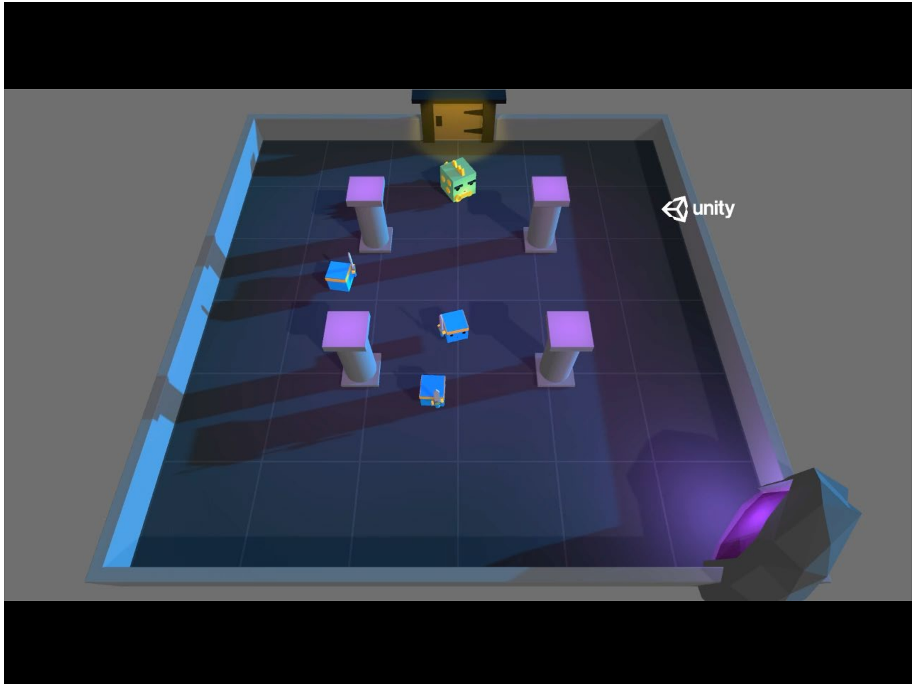
Kill the dragon to get the key and escape



Optimizing for the Greater Good



Greedy Solution



Optimizer for Greater Good

Summary

- Centralized learning, decentralized execution
- Asynchronous decision making
- Variable number of characters
- Optimizing for the greater good

MA-POCA

- Advantage calculation for agent *j*

$$G_{t:t+n} = (\bar{\mathbf{r}}_{\mathbf{t}} + \gamma \bar{\mathbf{r}}_{\mathbf{t+1}} + ... + \gamma^n V_\phi(atten(g(o^i_{t+n})_{1 \le i \le k}))), 0 \le t \le T - n$$

$$\hat{A}^j_t = (1-\lambda) \sum_{n=1}^{T-t-1} \lambda^{n-1} G_{t:t+n} + \lambda^{T-t-1} G_t - Q_\phi(atten(g(o^j_t), f(o^i_t, a^i_t)_{1 \le i \le k,\ i \ne j})$$

MA-POCA

- Advantage calculation for agent *j*

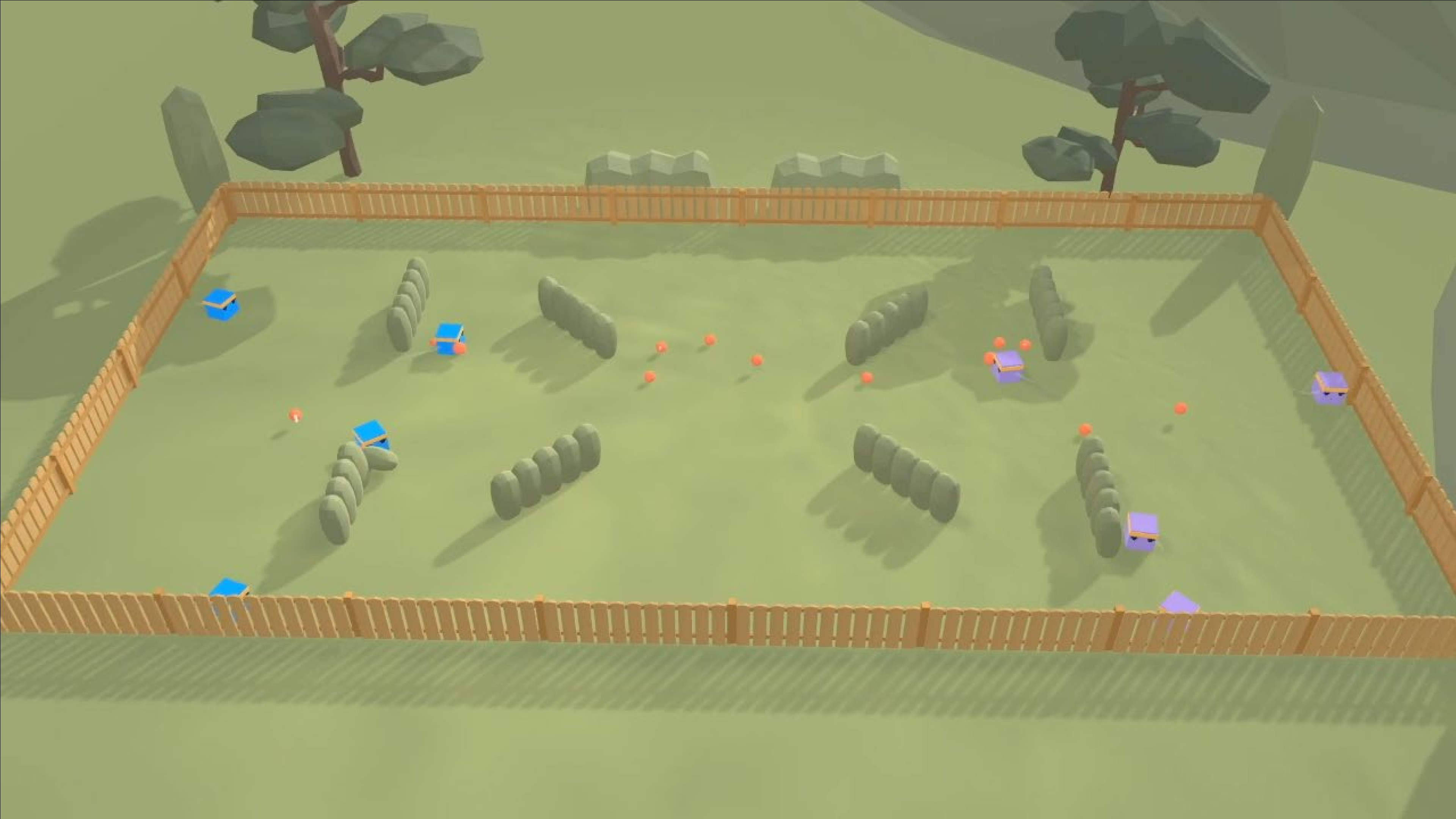
Group return at each time step

$$G_{t:t+n} = (\bar{\mathbf{r}}_{\mathbf{t}} + \gamma \bar{\mathbf{r}}_{\mathbf{t+1}} + \ldots + \gamma^n V_\phi(atten(g(o^i_{t+n})_{1 \leq i \leq k}))), 0 \leq t \leq T-n$$

$$\hat{A}^j_t = (1-\lambda) \sum_{n=1}^{T-t-1} \lambda^{n-1} \underline{G_{t:t+n}} + \lambda^{T-t-1} \underline{G_t} - Q_\phi(atten(g(o^j_t), f(o^i_t, a^i_t)_{1 \leq i \leq k,\ i \neq j})$$

Group return estimate (TD-lambda)

Baseline (Predict cumulative reward with action of agent *j* marginalized out)



References

- Centralized Critic : MADDPG
- Attention Mechanisms : Attention is all you need
- Counterfactual baseline : COMA

Contact

vincentpierre@unity3d.com

How to do Reinforcement Learning in NEON SHIFTER

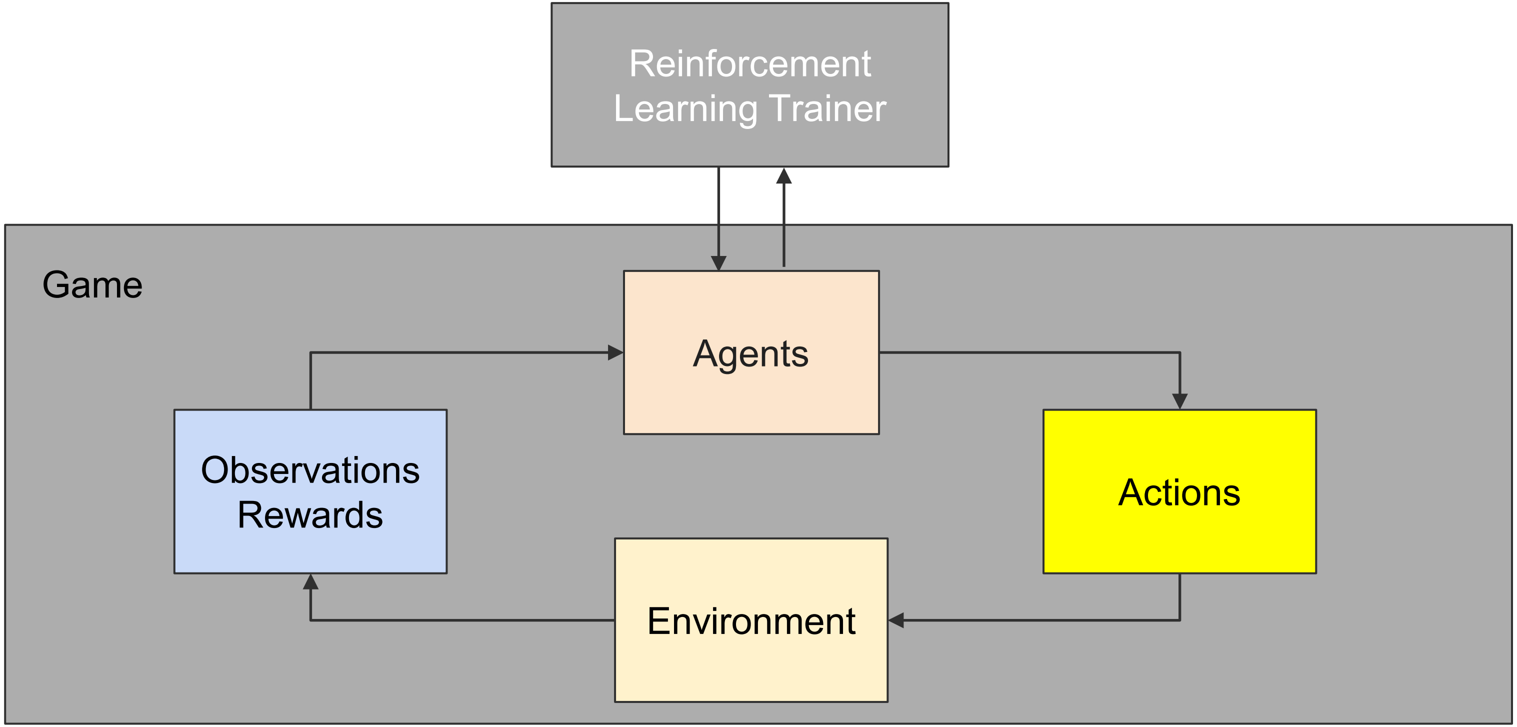
Couch in the Woods Interactive
Markus Weiß - Co-Founder

couchinthewoods.de

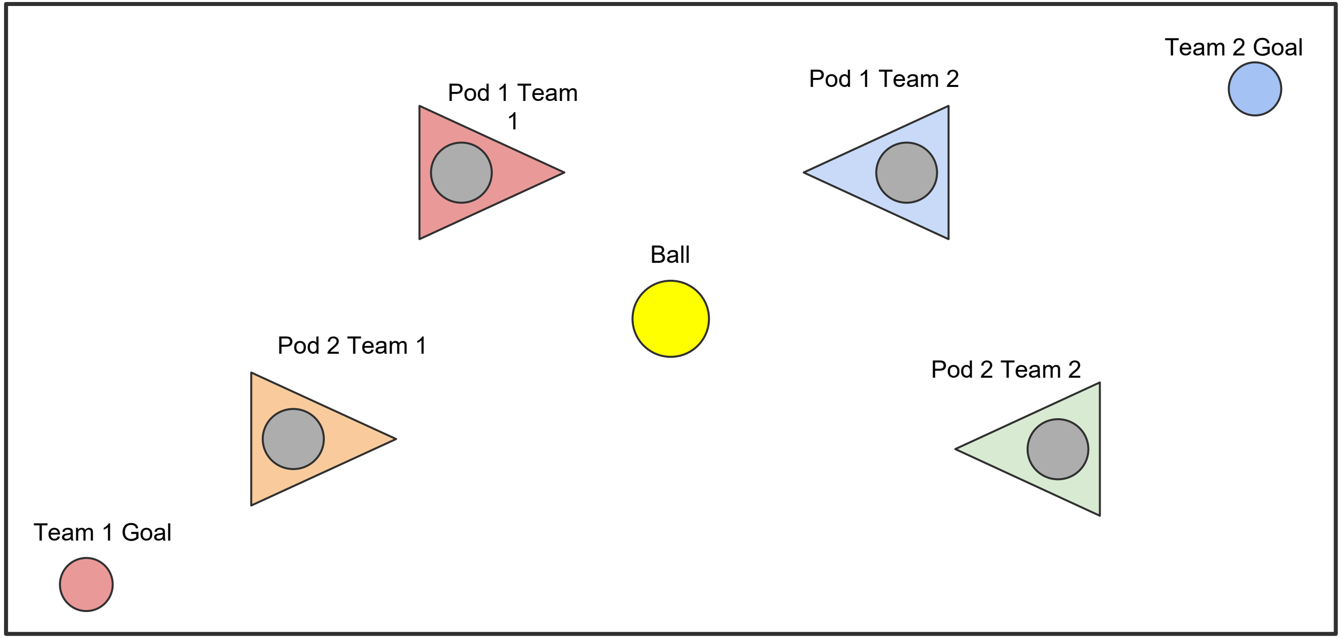
Overview

- Using Reinforcement Learning
  - Environment
  - Actions
  - Observations
  - Agent and Group Rewards
  - Reward Design
  - Behavior Design
- Results
- Summary and Outlook

Using Reinforcement Learning

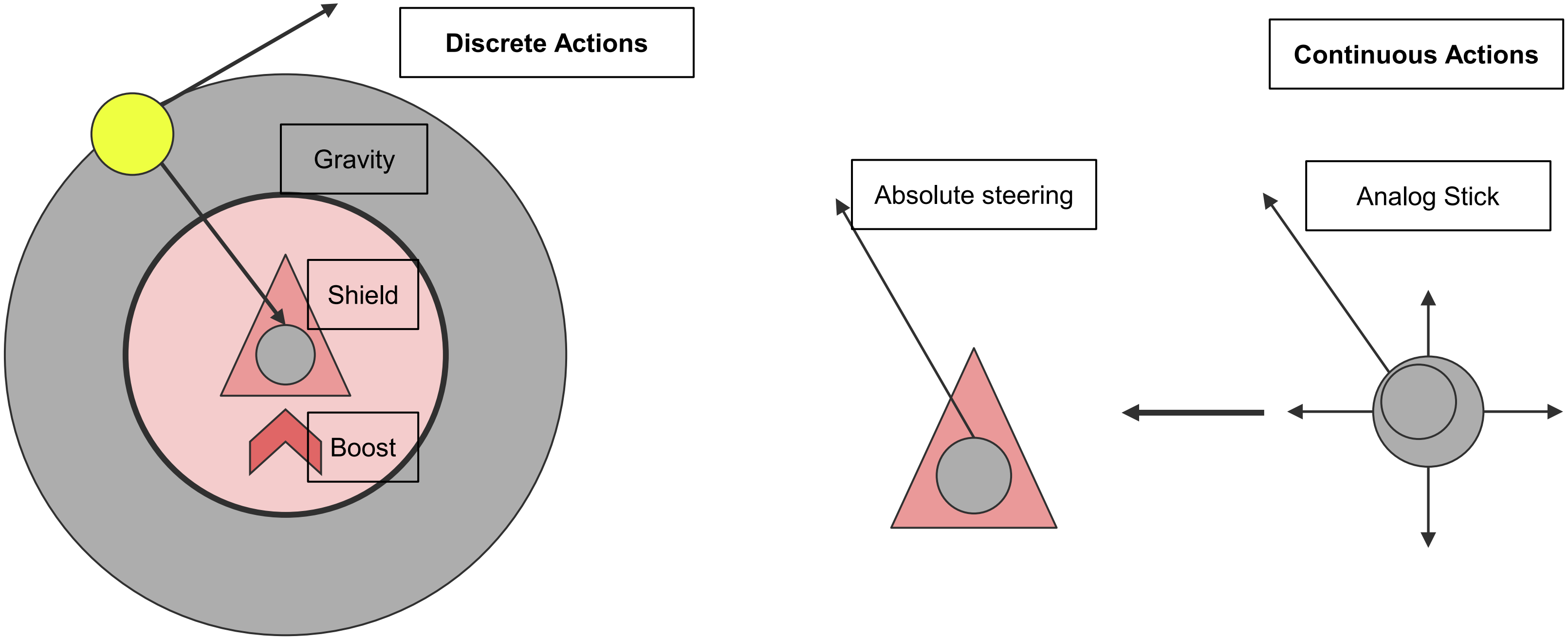


Environment

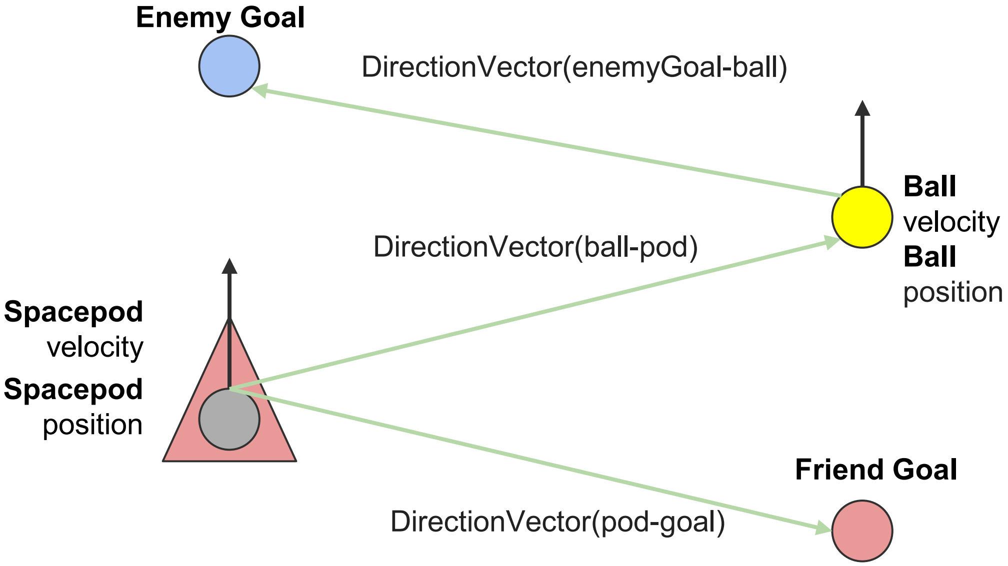


NEON SHIFTER - Gameplay

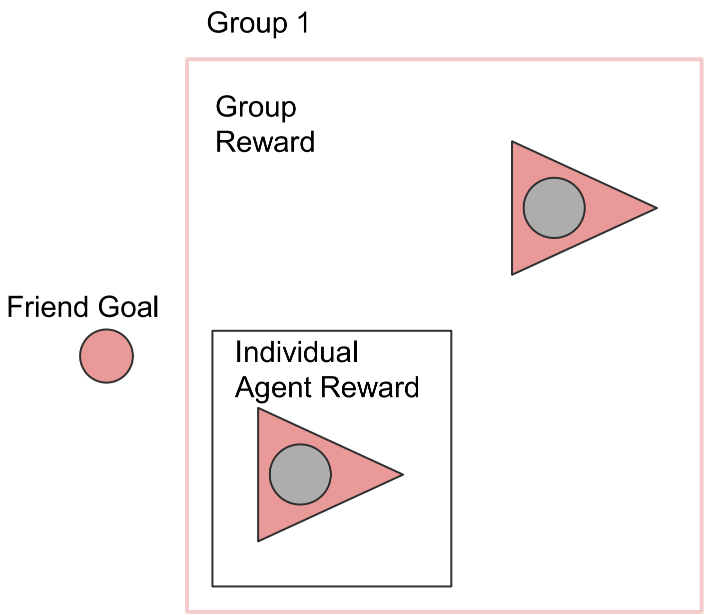
Actions


Discrete Actions
Gravity
Shield
Boost
Continuous Actions
Absolute steering
Analog Stick

Observations

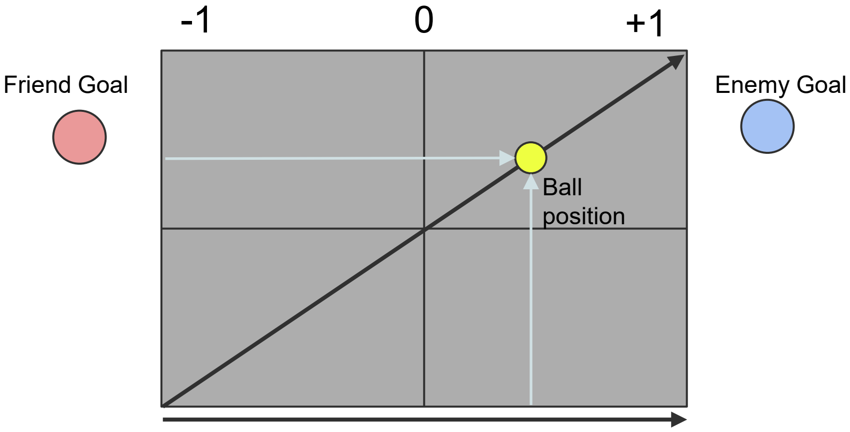


Agent and Group Reward



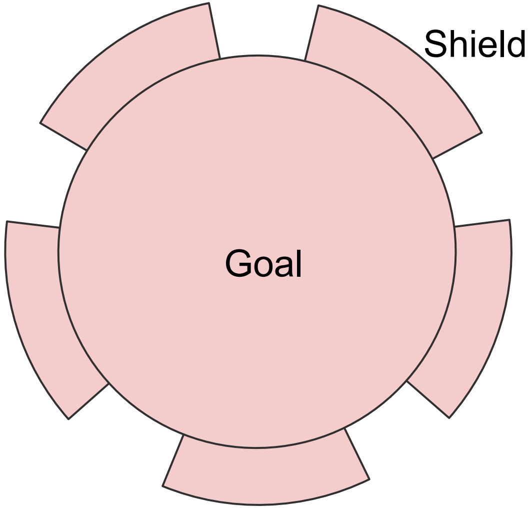
Ball Position Reward (GroupReward)

AddGroupReward(Ball Position Reward / maxStep)



Vector3.distance(friendGoal, enemyGoal)

Goal / Shield Reward (GroupReward)

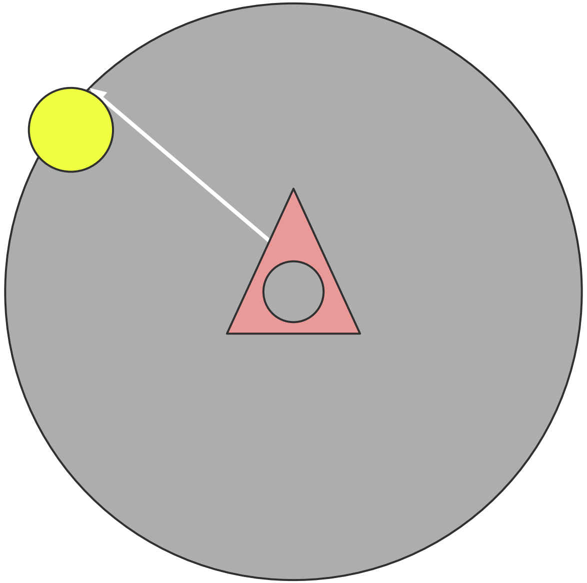


Shields Friend / Enemy
AddGroupReward +/- 0,1

Goal Friend / Enemy
AddGroupReward +/- 1
EndGroupEpisode()

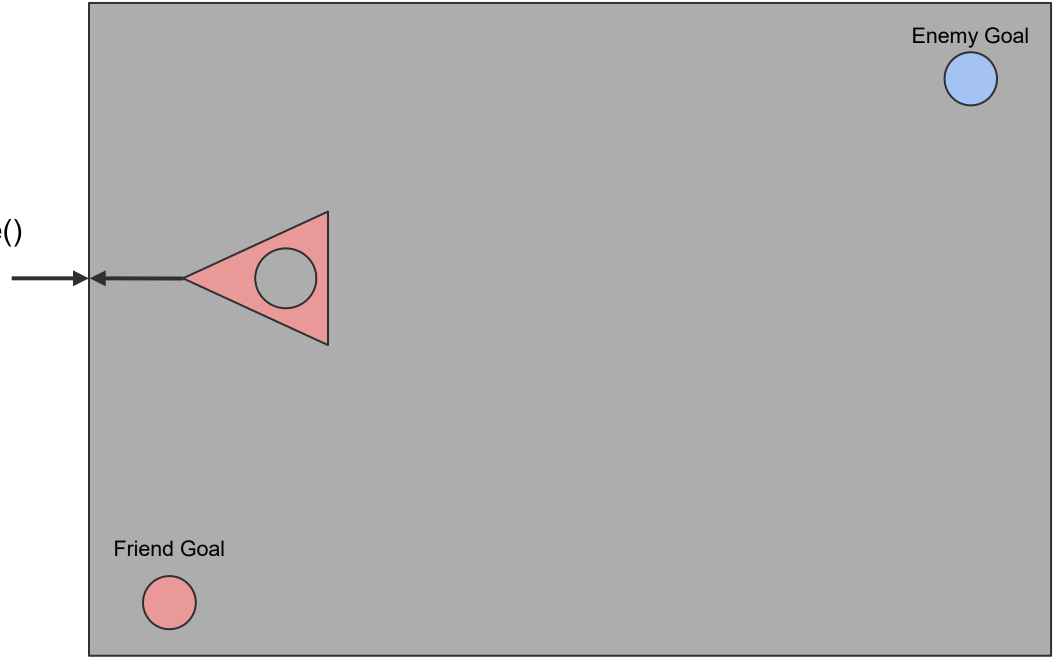
DistanceToBall Reward (Agent Reward)

if(BallIsInGravityRange)
AddReward(1 / MaxStep)

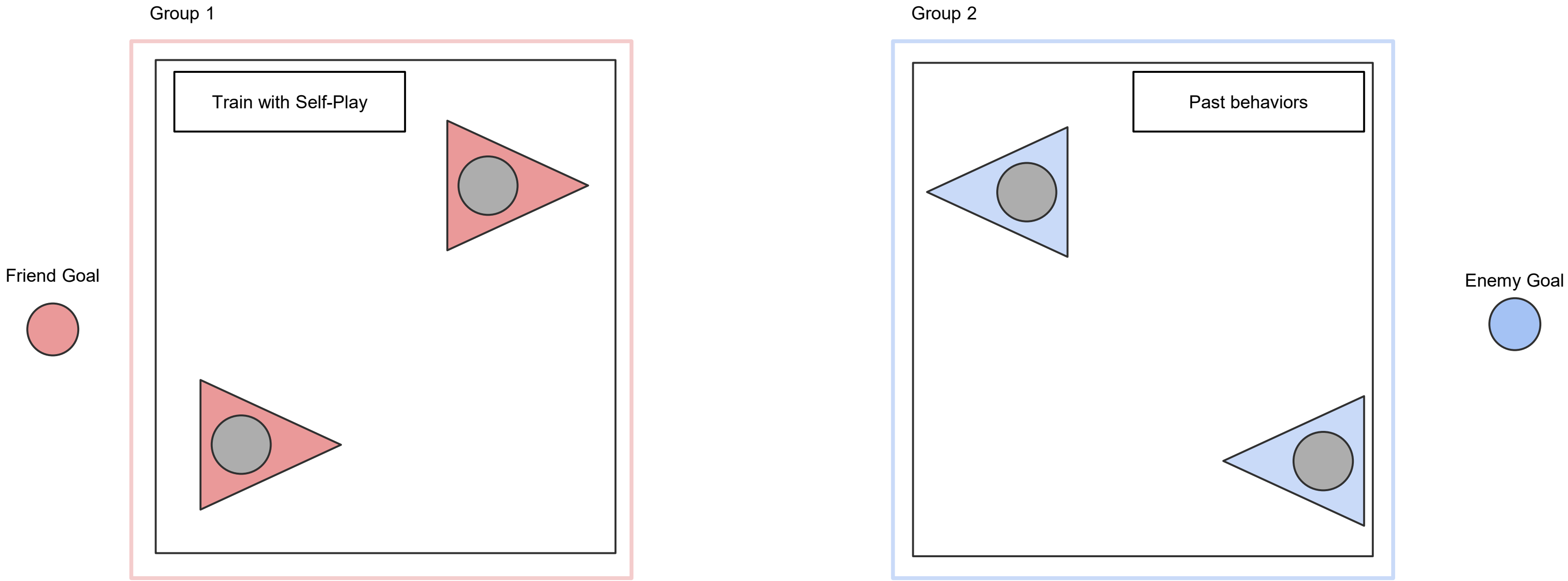


WallHit Reward (Agent Reward)

WallHitReward
if(count == 25)
addReward(-1)
EndGroupEpisode()



Behavior Design



Results

Summary

- Reinforcement Learning can be hard to get started with, but existing libraries make it easier
  - With existing libraries we don't need details about the complex Deep RL mathematics
- During training time, we are free to work on other parts of the game
- Configuring a training can sometimes be a little tricky
  - Starting off small or with a working example prevents mistakes
- We successfully reached our first target of creating an Agent our human play testers cannot beat

Outlook

- Our next goal is to adjust the difficulty of the Agent depending on the player's past performance

Thank you for your attention!

See funding partners of NEON SHIFTER below

„Das Projekt NEON SHIFTER der Firma Couch in the Woods Interactive wird im Rahmen des EXIST-Programms durch das Bundesministerium für Wirtschaft und Energie und den Europäischen Sozialfonds gefördert."

Ziel der Europäischen Union ist es, dass alle Menschen eine berufliche Perspektive erhalten. Der Europäische Sozialfonds (ESF) verbessert die Beschäftigungschancen, unterstützt die Menschen durch Ausbildung und Qualifizierung und trägt zum Abbau von Benachteiligungen auf dem Arbeitsmarkt bei. Mehr zum ESF unter: www.esf.de.

Bundesministerium für Wirtschaft und Energie

eXIST Existenzgründungen aus der Wissenschaft

ESF Europäischer Sozialfonds für Deutschland

Europäische Union

Zusammen. Zukunft. Gestalten.