RAPORT Z PRAC

13.04.2022

1. Przegląd podobnych rozwiązań – notatki

Baker, B., Kanitscheider, I., Markov, T., Wu, Y., Powell, G., McGrew, B., & Mordatch, I. (2019). Emergent tool use from multi-agent autocurricula. arXiv preprint arXiv:1909.07528.

Hide and seek

-nagrody: hiders -1 jeśli jakikolwiek znaleziony, +1 jeśli wszyscy ukryci, seekers -1 jeśli nikogo nie znaleźli, +1 jesli znaleźli chociaz 1 hidersa, -10 jeśli zbyt daleko pojda

-silnik: MUJOCO physics engine

-inputy: pozycja, velocity ~~rozmiar obiektow i innych agentow~~, jesli entity nie sa w linii widzenia albo w 135stopni z przodu to są maskowane, 30 laserow lidar

-agenty sa sferami

-możliwe akcje: ruch poprzez nadanie siły w osiach x y z, binarne podnoszenie obiektów, binarne zablokowanie obiektów w miejscu

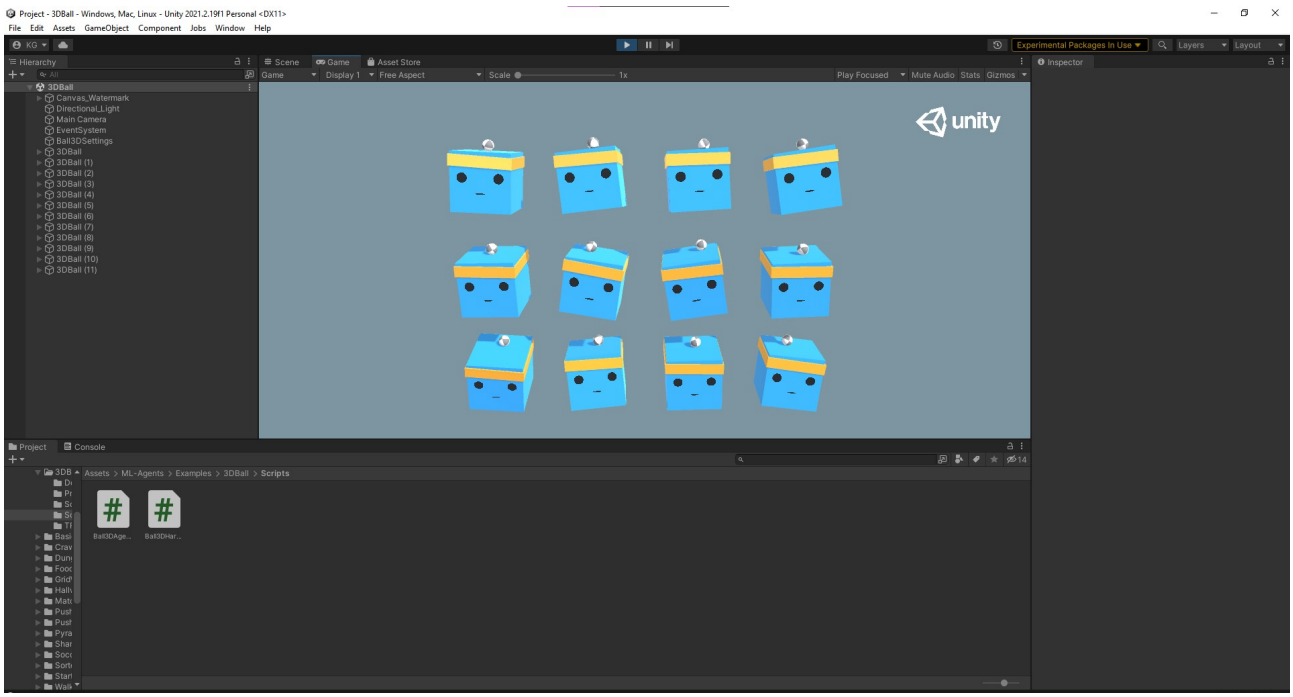
-Policies are optimized using Proximal Policy Optimization (PPO) (Schulman et al., 2017) and Generalized Advantage Estimation (GAE) (Schulman et al., 2015), and training is performed using rapid (OpenAI, 2018)

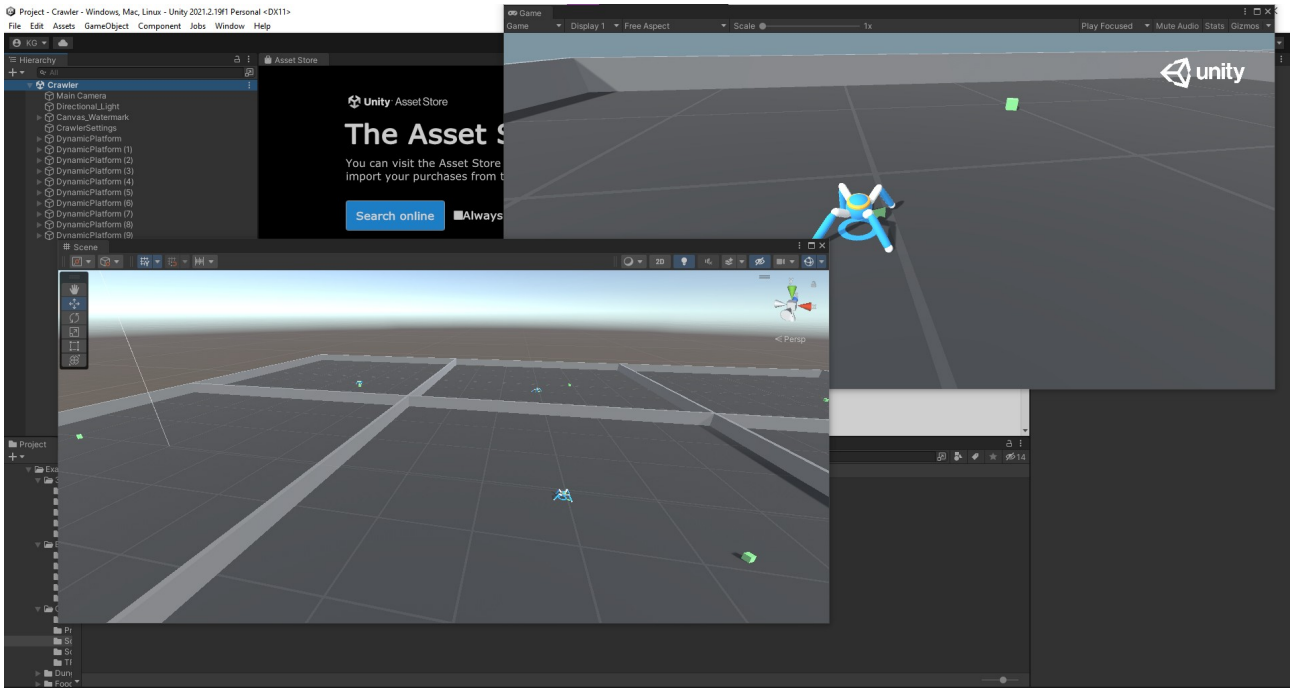
-If randomization is reduced, we find that fewer stages of the skill progression emerges, and at times less sophisticated strategies emerge instead (e.g. hiders can learn to run away and use boxes as moveable shields.);

- Dodatkowo soccer z Unity ML-Agents, wraz z kilkoma innymi przykładami. Mamy dostęp do ich pełnych implementacji więc jest idealne źródło wiedzy, na którym możemy się wzorować.

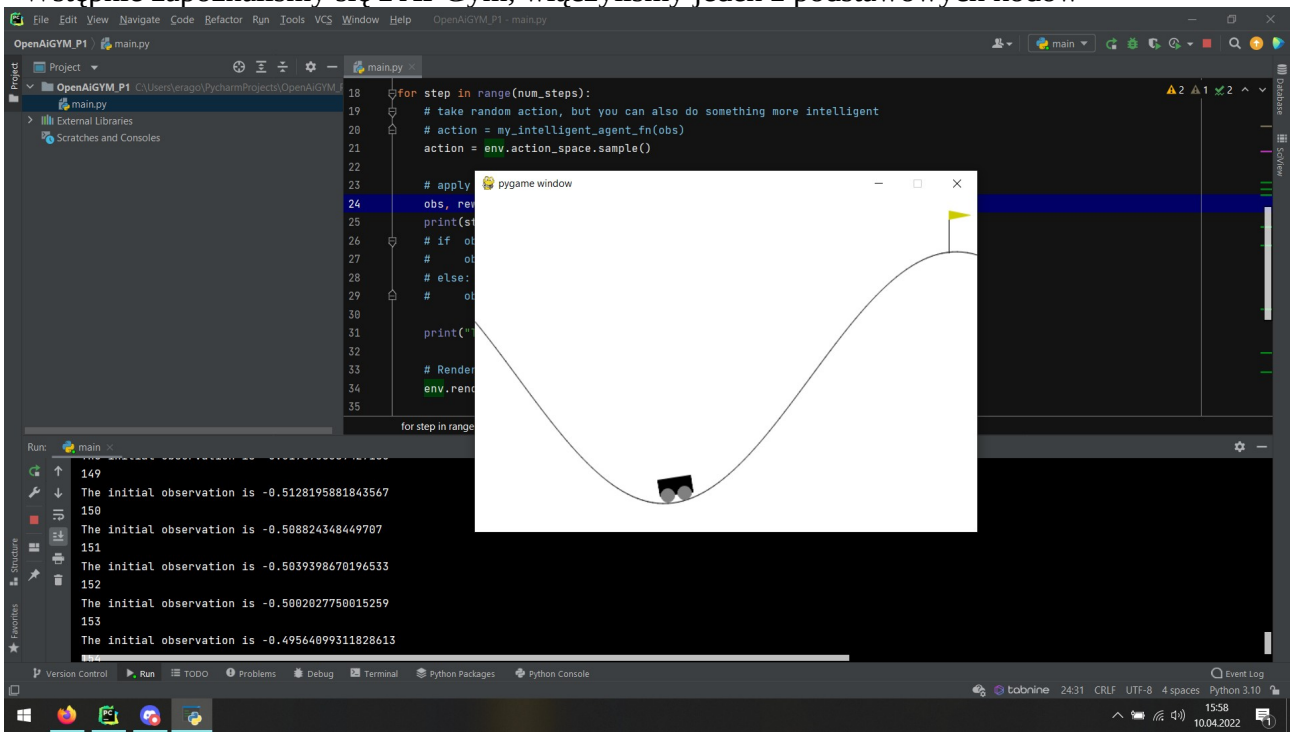
2. ML-Agents i AI-Gym

- Udało nam się włączyć przykładowe środowisko zapewnione przez ML-Agents





- Wstępnie zapoznaliśmy się z AI-Gym, włączyliśmy jeden z podstawowych kodów



- Znaleźliśmy Unity ML-Agents Gym Wrapper specjalnie do celu połączenia ML-Agents a AI-Gym, jednak jest on ograniczony, gdyż można go stosować tylko do środowisk z jednym agentem, co prawdopodobnie całkiem dyskwalifikuje go w naszym przypadku

- Czego w takim razie użyjemy do połączenia pytorcha z ML-Agents? Na ten moment znaleźliśmy: https://github.com/Unity-Technologies/ml-agents/blob/main/docs/Python-API.md

3. Dodatkowo

- zapoznaliśmy się z LATEX i stworzyliśmy sobie środowiska do tworzenia w nim plików na swoich komputerach

- zaczynamy zapoznawać się z PyTorchem wg tutoriala:
https://pytorch.org/tutorials/beginner/pytorch_with_examples.html

- Tworzymy wspólną notatkę ze znalezionymi rozwiązaniami typu ε-Greedy Exploration albo Curiosity driven model. To się przyda zapewne niedługo przy wyborze modelu, którego my użyjemy.

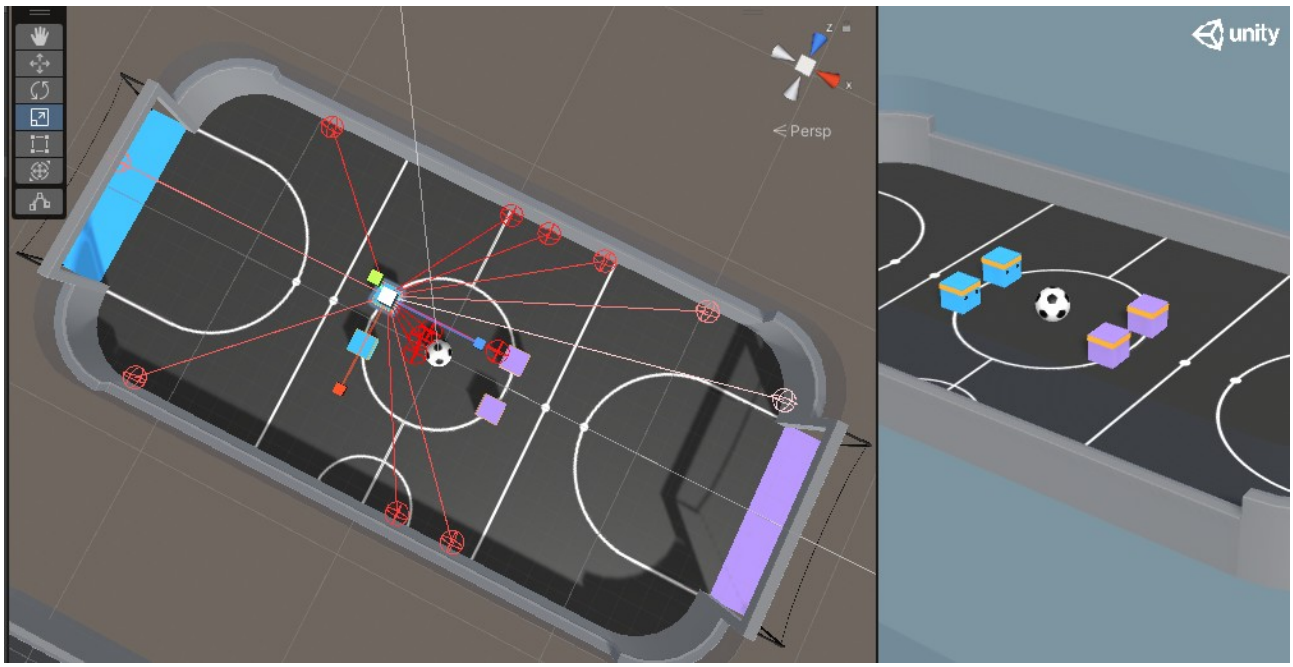
4. Logika gry

- Edytowaliśmy dokument z Zasadami Gry na dysku. Zawiera on teraz aktualną i pełną wersję logiki rozgrywki.

5. Pytania

- Czy w uczeniu wykorzystamy jakiś przyspieszacz czasu? Zmienną TIME? Żeby Uczenie bez interfejsu przebiegało z pełną szybkością obliczeniową. Co się robi w takiej sytuacji?

- Zakładamy, że mając pewne promienie widzenia agent potrafi odróżnić, tzn. zwraca czy dany promień widzenia zderza się z flagą, ścianą, czy jednym z biomów, czy z przeciwnikiem?
W innym wypadku w jaki sposób agenci z przykładu niżej nauczyli się odróżniać piłkę od ściany czy przeciwnika. Chyba trzeba przekazywać coś więcej np. Że jest to piłka (w przypadku przykładu niżej) niż tylko gdzie jest „coś” – jakiś obiekt.



- Czy w takim wypadku nasi agenci mają wiedzieć na początku gdzie są na przykład bazy albo flaga przeciwnika albo własna flaga?