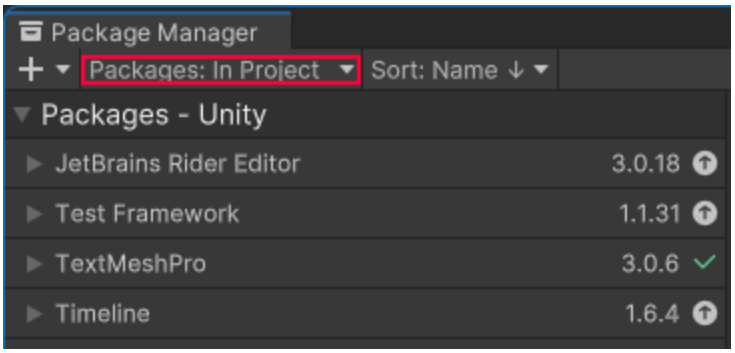
Laboratorium Zaawansowanych Technik Programowania Grafiki Komputerowej

Uczenie Maszynowe w Unity

Zadanie 1: Przygotowania

1. W ramach laboratorium wykorzystywane będą komponenty wykonane w języku Python. Środowisko Python powinno być zainstalowane, ale aby się upewnić czy tak jest, włącz konsolę i wpisz “py”. Jeśli odpowiedź zaczyna się od “Python 3.7.2” lub wyżej, Python jest zainstalowany, a Ty przejdź do następnego punktu.
    a. Jeśli Python nie jest zainstalowany, pobierz wersję co najmniej 3.8.13, ale nie późniejszą niż 3.9.2rc1 ze strony https://www.python.org/downloads/windows/
    b. Uruchom pobrany instalator, zaznacz “Add python.exe to PATH”, zainstaluj domyślnie.
    c. Wyłącz limit znaków i zamknij instalator po zakończonej instalacji.
    d. Zaktualizuj menadżer pakietów za pomocą “python.exe -m pip install --upgrade pip”
2. Zainstaluj pakiet mlagents. Wpisz w konsoli “python -m pip install mlagents==0.30.0”
3. Uruchom Unity Hub.
4. Stwórz nowy projekt (lub użyj już istniejącego, patrz niżej). Instrukcja napisana jest dla projektu 3D w potoku URP, w przypadku innych potoków mogą być drobne zmiany w interfejsie. Jeśli na wykorzystywanym komputerze nie tworzono wcześniej takiego projektu, wybierz [Download template] aby pobrać szablon.
5. Laboratorium wymaga stworzenia agenta sztucznej inteligencji. Jeśli masz jakąś scenę, do której chciałbyś dodać agenta - tym lepiej. Możesz wykorzystać scenę z modelem postaci, której animowałeś ciało i twarz na poprzednich laboratoriach, wraz z wcześniej wykonanym oświetleniem. Jeśli chcesz jakąś inną, która nie byłaby pusta, wybierz jedną z Asset Store [Window->Asset Store]. Jeśli wybrałeś środowisko z Asset Store, pobierz je (Download) i importuj (Import) do swojego projektu zgadzając się by importować całość.
6. Uruchom scenę, jeśli nie jest uruchomiona domyślnie
7. Jeśli scena jest pusta/jednolita, umieść na niej kilka kształtów, które pozwolą Ci się zorientować w którą stronę skierowana jest kamera.

8. Otwórz menadżer pakietów [Window->Package Manager].
9. W menadżerze pakietów, wybierz rejestr klikając na [Packages: In Project] i zmieniając wartość na [Packages: Unity Registry].
10. Z listy pakietów wybierz [ML Agents] i kliknij [Install]



Zadanie 2: Przykładowe środowisko uczenia

1. W ramach tego zadania stworzysz proste środowisko, którego użyjemy jako przykładu uczenia maszynowego. Celem jest to, by potwierdzić, że wszystko działa prawidłowo od strony Unity.
2. Umieść na scenie płaszczyznę, będzie reprezentowała powierzchnię, po której porusza się agent. Jeśli wykroczy poza płaszczyznę i spadnie - przegra.
3. Umieść na płaszczyźnie sześcian. Będzie reprezentował cel, do którego agent będzie podążał. Jeśli go osiągnie, wygra.
4. Umieść na płaszczyźnie kulę, będzie reprezentowała naszego agenta.
5. Umieść wszystkie elementy wewnątrz wspólnego obiektu hierarchii.
6. Do kuli dodaj komponent [RigidBody] w celu realizacji fizyki.
7. Do kuli dodaj skrypt [ML Agents\Decision Requester]
8. Do kuli dodaj skrypt [ML Agents\Behavior Parameters].
    a. Nadaj nazwę zachowaniu, które będziesz uczył. Na poczet instrukcji, będzie to “MojAgent”
    b. Ustal wielkość wektora obserwacji, [Vector Observation\Space Size]. To liczba prymitywnych zmiennych, które agent będzie rejestrował ze swojego otoczenia. Dla nas będzie to pozycja agenta w każdym wymiarze, pozycja celu w każdym wymiarze, oraz prędkość agenta na płaszczyźnie, po której będzie się poruszał. Ile zmiennych potrzebujemy?
    c. Ustal wielkość sygnału sterującego agentem [Actions\Continuous Actions]. Ile potrzebujemy zmiennych, by móc osiągnąć cel?
9. Do kuli dodaj nowy skrypt, nazwij go tak, jak nazwałeś zachowanie. Na poczet tej instrukcji, będzie to “MojAgent”.
10. Zmodyfikuj nowo powstały skrypt. Będziesz używać trzech przestrzeni nazw:
    using Unity.MLAgents;
    using Unity.MLAgents.Sensors;
    using Unity.MLAgents.Actuators;
11. Zmień klasę po której dziedziczy skrypt na “Agent” - to klasa reprezentująca agenta sztucznej inteligencji w ramach MLAgents.
12. Nie wahaj się modyfikować innych aspektów skryptu, niż opisane niżej. Opisane są tylko te, które wynikają bezpośrednio z zastosowania MLAgents.

13. Nadpisz metodę “public void OnEpisodeBegin()”. Wyzeruj stan fizycznej reprezentacji agenta, tak, żeby nie wpływały na niego żadne siły z poprzedniego epizodu. Wylosuj nową pozycję, w której pojawi się agent.
14. Nadpisz metodę “public void Heuristic(in ActionBuffers actionsOut)”. Służy ona do ręcznego sterowania agentem. Pobierz dane sterujące z klawiatury, lub innego urządzenia wejściowego i wypełnij nimi kolekcję “ContinuousActions” parametru “actionsOut”. Będą to zmienne ciągłe, za pomocą których będzie przeprowadzone sterowanie kulą.
15. Nadpisz metodę “public void OnActionReceived(ActionBuffers actionBuffers)”. W ramach metody pobierz wartości wpisane do parametru “actionBuffers” (patrz poprzednia metoda) i na ich podstawie poruszaj kulą na podstawie sił fizycznych. W ramach metody, określ czy spadłeś z płaszczyzny lub czy osiągnąłeś cel. W obu przypadkach, zakończ epizod uczenia poprzez wywołanie metody “EndEpisode()”, jednak w przypadku osiągnięcia celu, przyznaj najpierw nagrodę za pomocą metody “SetReward(float)”.
16. Uruchom grę i sprawdź, czy wszystkie dotychczasowe założenia odnośnie działania gry są spełnione, jeśli to Ty nią sterujesz.

Zadanie 3: Podstawowa nauka agenta

1. W ramach tego zadania przeprowadzisz proste uczenie. Celem jest to, by potwierdzić, że wszystko działa prawidłowo od strony biblioteki uczenia maszynowego.
2. Nadpisz metodę “public void CollectObservations(VectorSensor sensor)”. Dodaj do wektora obserwacji (parametr “sensor”) informacje o istotnych czynnikach wpływających na ruch agenta. Rozważ gdzie jest agent, gdzie potrzebuje dotrzeć, a także z jaką prędkością się aktualnie porusza.
3. Utwórz plik yaml zawierający konfigurację środowiska uczącego. Przykładowy plik yaml wyglądać będzie następująco:

```
behaviors:
  MojAgent:
    trainer_type: ppo
    hyperparameters:
      batch_size: 10
      buffer_size: 100
      learning_rate: 3.0e-4
      beta: 5.0e-4
      epsilon: 0.2
      lambd: 0.99
      num_epoch: 3
      learning_rate_schedule: linear
      beta_schedule: constant
      epsilon_schedule: linear
    network_settings:
      normalize: false
```

```
  hidden_units: 128
  num_layers: 2
reward_signals:
  extrinsic:
    gamma: 0.99
    strength: 1.0
max_steps: 500000
time_horizon: 64
summary_freq: 10000
```

4. Zwróć uwagę na nazwę zachowania. Musi być ona zgodna z nazwą w [Behavior Parameters]. W naszym przypadku będzie to “MojAgent”
5. Uruchom proces uczenia za pomocą komendy w konsoli:
   “mlagents-learn "ŚCIEŻKA_DO_PLIKU_YAML" --run-id=MojAgent”
6. Jeśli otrzymujesz błąd typu “TypeError: Descriptors cannot not be created directly.”, przejdź na niższą wersję pakietu protobuf poprzez komendę w konsoli:
   “pip install protobuf==3.20.3”
7. Jeśli otrzymujesz błąd wskazujący na brak PyTorcha, możesz go zainstalować za pomocą komendy w konsoli:
   “pip3 install torch~=1.7.1 -f https://download.pytorch.org/whl/torch_stable.html”
8. Pierwsze uruchomienie może trwać istotnie długo. Jeśli kursor miga, ale nic się nie zmienia - daj chwilę programowi.
9. Po wyświetleniu się komunikatu “[INFO] Listening on port 5004. Start training by pressing the Play button in the Unity Editor.”, program się zainicjalizował, a Ty możesz nacisnąć przycisk [Play] w Unity.
10. Po uruchomieniu gry, konsola powinna poinformować Cię o prawidłowym połączeniu “Connected to Unity environment with package version 2.0.1 and communication version 1.5.0”
11. Powinieneś zauważyć, że teraz kula porusza się bez Twojej pomocy. Obserwuj jak rodzi sobie agent sztucznej inteligencji. Czy spada coraz rzadziej i coraz szybciej dociera do celu?
12. Po każdych 10 000 kroków, na konsoli zobaczysz informację o średniej nagrodzie i odchyleniu standardowym nagrody. Pierwsze powinno rosnąć, drugie spadać. Agent się uczy.

Zadanie 4: Stworzenie własnego agenta

1. Wiesz już, że wszystko skonfigurowane jest prawidłowo. Możesz przystąpić do tego, by stworzyć własnego agenta sztucznej inteligencji.
2. Wykorzystaj wcześniej zrobione, lub stwórz nowe środowisko, w którym chciałbyś wyuczyć agenta. Upewnij się, że Twój pomysł na to, co agent powinien robić, jest inny, niż pomysł innych grup. Porozmawiaj z prowadzącym na temat swojego pomysłu, żeby upewnić się, że nie jest on za prosty, ani nie jest też za trudny. W przypadku

trudniejszych pomysłów, będziesz oceniany lżej, ale czasem może nie udać Ci się osiągnąć zadowalających efektów.

3. Wykorzystaj dokumentację Unity ML-Agents Toolkit dostępną pod adresami:
   https://unity-technologies.github.io/ml-agents/
   https://github.com/Unity-Technologies/ml-agents
   W większości problemów nie będą potrzebne dodatkowe funkcjonalności, jednak jeśli coś jest niejasne, powinno to być pierwsze miejsce, w które zajrzysz. W razie wątpliwości zapytaj prowadzącego.
4. Myśl kreatywnie o tym, co właściwie jest agentem. Wcale nie musi nim być chodząca postać gracza - równie dobrze może nią być płaszczyzna po której postać się porusza, lub przeszkody, które utrudniają graczowi grę.
5. Oprogramuj agenta wykorzystując poznane wcześniej fragmenty środowiska ML-Agents.
6. Rozpocznij uczenie swojego agenta. Obserwuj czy radzi sobie coraz lepiej. Czy ma szansę nauczyć się tego, czego potrzebuje?
7. Rozważ system nagród. Nie wymagaj od swojego agenta, by nauczył się biegać bezpośrednio po raczkowaniu. Nagradzaj już za chodzenie. Jeśli Twój agent ma przed sobą skomplikowane zadania, upewnij się, że otrzymuje nagrody, za wykonywanie zadań cząstkowych, które mogą doprowadzić go do celu, a nie tylko za coś, czego nie jest w stanie osiągnąć poprzez losową eksplorację możliwości.