UNIWERSYTET MARII CURIE-SKŁODOWSKIEJ
W LUBLINIE
Wydział Matematyki, Fizyki i Informatyki

Kierunek: **Informatyka**
Specjalność: -

Wiktor Wajszczuk
nr albumu: 296534

Zastosowanie uczenia przez wzmacnianie w środowisku agentowym

Application of reinforcement learning in an agent-based environment

Praca licencjacka
napisana w Katedrze Oprogramowania Systemów Informatycznych
pod kierunkiem dr Marcina Denkowskiego

Lublin, rok 2022

Spis treści

Wstęp

Jedną z większych dziedzin uczenia maszynowego jest uczenie przez wzmacnianie (ang. *reinforcement learning*) W odróżnieniu od zarówno uczenia nadzorowanego i nienadzorowanego nie potrzebujemy w tym przypadku żadnych gotowych danych wejściowych i wyjściowych. Zamiast tego, algorytm pozyskuje dane na bieżąco ze środowiska do, którego jest zastosowany. Dzięki temu, że algorytmy uczenia przez wzmacnianie nie mają tego ograniczenia możemy zastosować je do problemów takich jak gra na giełdzie[1], czy nauka grania w gry, w swojej pracy skupię się na tym drugim.

Celem pracy jest zaimplementowanie algorytmu uczenia przez wzmacnianie Q-Learningu oraz zoptymalizowaniu go tak by po zastosowaniu go do własnoręcznie zaimplementowanej gry był w stanie osiągnąć w niej możliwie najwyższy wynik.

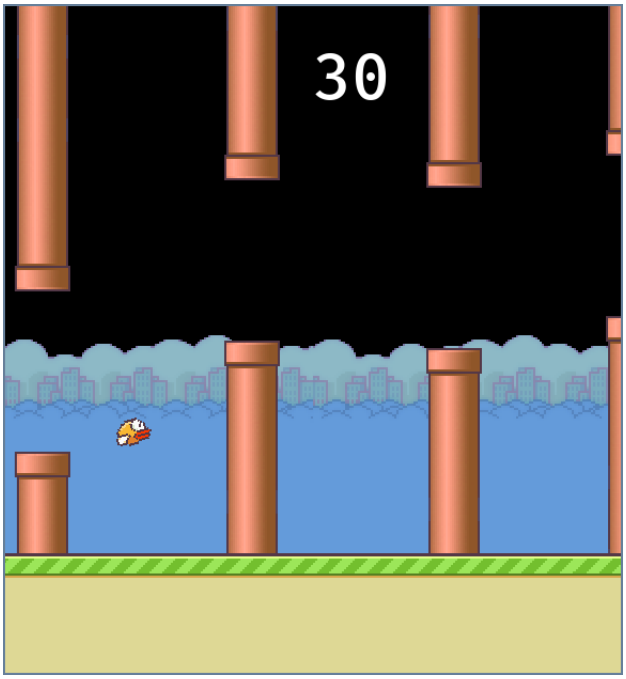
Na początku przedstawię jaką grę wybrałem, opiszę jej zasady oraz zaprezentuję szczegóły jej implementacji. Następnie przybliżę zagadnienie uczenia przez wzmacnianie oraz algorytmu Q-learning. Pod koniec pokażę jak zaimplementowałem wspomniany algorytm, oraz testy po optymalizacjach algorytmu, które zastosowałem.

Rozdział 1

Gra

Problem, który postawiłem przed Q-learningiem to gra z 2013 roku “Flappy Bird” autorstwa wietnamskiego developera Dong Nguyena[2]. Zdecydowałem się właśnie na tą grę, gdyż sterowanie w niej jest bardzo proste – jest tylko jeden przycisk do kontrolowania toteż jedna akcja do podjęcia przez algorytm.

1.1 Zasady gry



Rysunek 1.1: Zrzut ekranu z gry

Gracz kontroluje ptaka (na rysunku 1.1 kolor żółty), jego zadaniem jest tak nim sterować by omijać czerwone rury. Na ptaka działa jedynie siła grawitacji a to rury są

przesuwane w jego stronę ze stałą prędkością. Jak wspomniałem wcześniej gracz ma możliwość jedynie kontrolować czy ptak załopocze skrzydłami przeciwdziałając sile grawitacji i unosząc się czy tego nie wykona i opadnie. Gra kończy się gdy ptak spadnie na ziemię (część zielona na dole na rysunku 1.1) albo uderzy w jedną z dwóch nadchodzących do niego czerwonych rur – dolnej bądź górnej. Punkty przyznawane są jeżeli graczowi uda się ominąć nadchodzące przeszkody przelatując przez szczelinę między nimi. Za każde ominięcie przyznawany jest jeden punkt. Jak widać na rysunku 1.1 gracz ominął już trzydzieści par rur. Podsumowując – gra sprowadza się do kontrolowania wysokości ptaka tak by ten przelatywał między nadchodzącymi rurami.

1.2 Wykorzystane technologie

Do zaimplementowania tej gry zdecydowałem się na użycie skryptowego języka **python** w wersji 3.8. Wybrałem tą technologię ze względu na prostotę w pisaniu kodu, świetną dokumentację, fakt, że jest dostępny na większości współczesnych platform oraz to, że jest to język open-source. Dodatkowo wykorzystałem moduł **pygame**.

1.2.1 Pygame

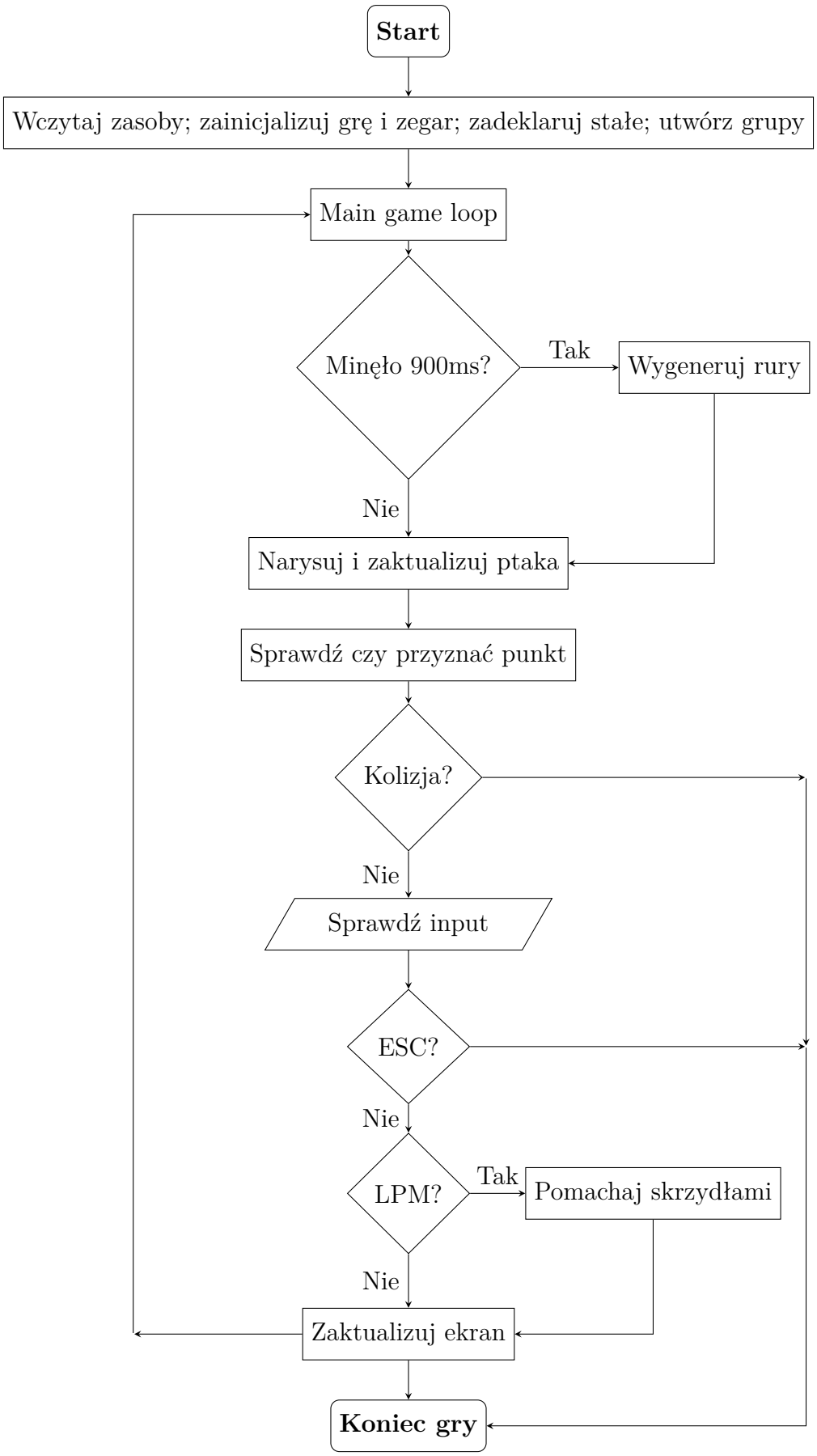
Pygame dodaje funkcjonalność do biblioteki SDL (więcej o SDLu w następnym podrozdziale). Pozwala tworzyć w pełni funkcjonalne gry oraz programy multimedialne w pythonie. Zaletami pygame są między innymi : podstawowe funkcje używają zoptymalizowanego kodu w C oraz Assembly co sprawia, że pygame jest naprawdę szybkie, podobnie jak python jest dostępny na większości współczesnych platform oraz prosty w użyciu co zaprezentuję w części implementacyjnej. Ponadto jest modularny co pozwala programiście używać tylko tych komponentów, których naprawdę potrzebuje.

1.2.2 SDL – Simple DirectMedia Layer

Jest to wieloplatformowa biblioteka zapewniająca nisko poziomowy dostęp do audio, urządzeń peryferyjnych takich jak mysz, klawiatura, joystick, sprzętu graficznego (za pomocą OpenGL i Direct3D). Technologia open source zaimplementowana w C. To za pomocą tej biblioteki pygame może odczytywać wejście czy rysować wyjście na ekranie.

1.3 Implementacja

W tym podrozdziale przedstawię w jaki sposób użyłem wyżej wymienionych technologii do zaimplementowania gry, zacznę od opisania diagramu przepływu.


Start
Wczytaj zasoby; zainicjalizuj grę i zegar; zadeklaruj stałe; utwórz grupy
Main game loop
Minęło 900ms?
Tak
Wygeneruj rury
Nie
Narysuj i zaktualizuj ptaka
Sprawdź czy przyznać punkt
Kolizja?
Nie
Sprawdź input
ESC?
Nie
LPM?
Tak
Pomachaj skrzydłami
Nie
Zaktualizuj ekran
Koniec gry

Tutaj będą opisane po kolei nody co robią na diagramie

Rozdział 2

Uczenie przez wzmacnianie

Tutaj do napisania o tym czym jest uczenie przez wzmacnianie pare slow jakie są algorytmy jak działa, schemat środowisko - agent - akcja - reward

2.1 Q-learning

Tutaj wszystko związane z qlearningiem, wzory, itp :)

Rozdział 3

Implementacja Q-learningu

Tutaj fragmenty kodu, jakie zmiany dokonałem względem książkowego algorytmu

3.1 Podejście pierwsze grid 10 na 10

Po kolei wyjaśnienie co to grid 10 na 10 jak to jest implementowane itp

3.2 Podejście drugie grid 10 na 10 i flaga za śmierć od wysokiej rury

Jak wyżej

3.3 Podejście trzecie grid 5 na 5 z flagą

Jak wyżej

Rozdział 4

Podsumowanie

Wyniki osiągane są w miarę ok ale mogło by być lepiej bo są na internecie lepsze podejścia do tego problemu – być może wynika to z tego ze sam implementowałem grę i nie jest ona tak dobrze przetestowana jak ta dostępna na GitHub?

4.1 Co można by ulepszyć

Tutaj będę pisał co mogłem zrobić gdybym poświęcił więcej czasu np:

- uczenie bez wizualizacji pygamowej
- zrównoleglenie by paru agentów na raz mogło się uczyć
- doszlifowanie algorytmu??

Bibliografia

[1] K. Dabérius, E. Granat, and P. Karlsson, “Deep execution - value and policy based reinforcement learning for trading and beating market benchmarks,” 2019. [Online]. Available: http://dx.doi.org/10.2139/ssrn.3374766

[2] “Strona internetowa dotgears.” [Online]. Available: https://dotgears.com/games/flappybird