



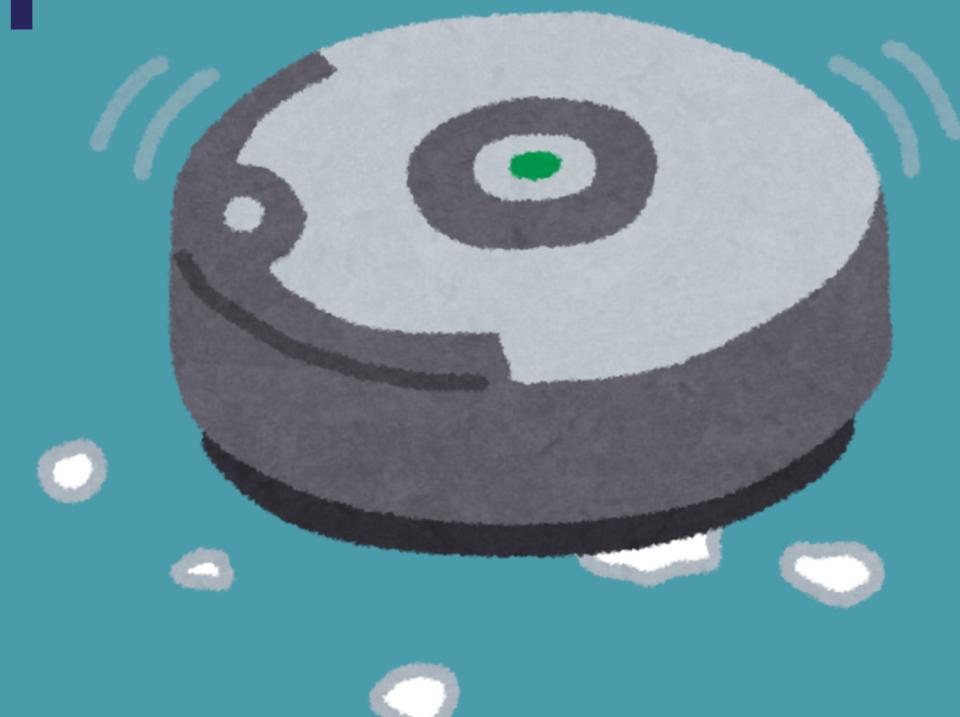
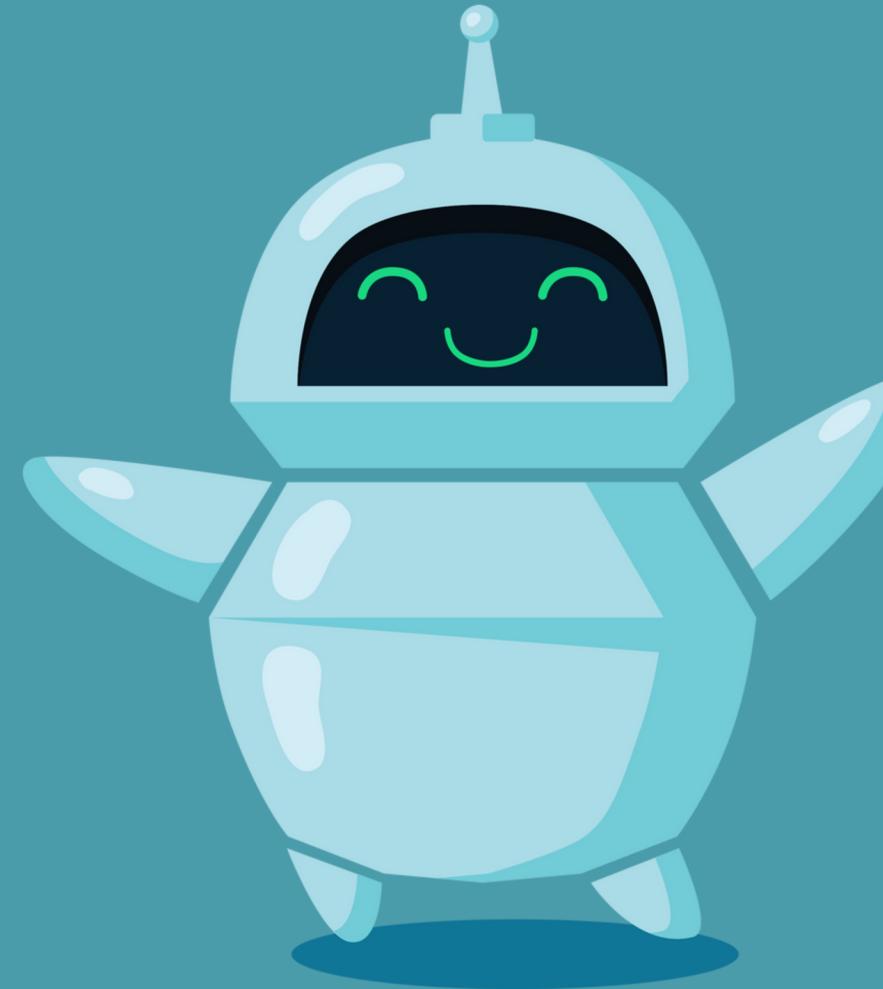
POLITECHNIKA
RZESZOWSKA
im. IGNACEGO ŁUKASIEWICZA

PROJEKT INŻYNIERSKI

SYMULATORY ROBOTÓW MOBILNYCH W JĘZYKU PYTHON

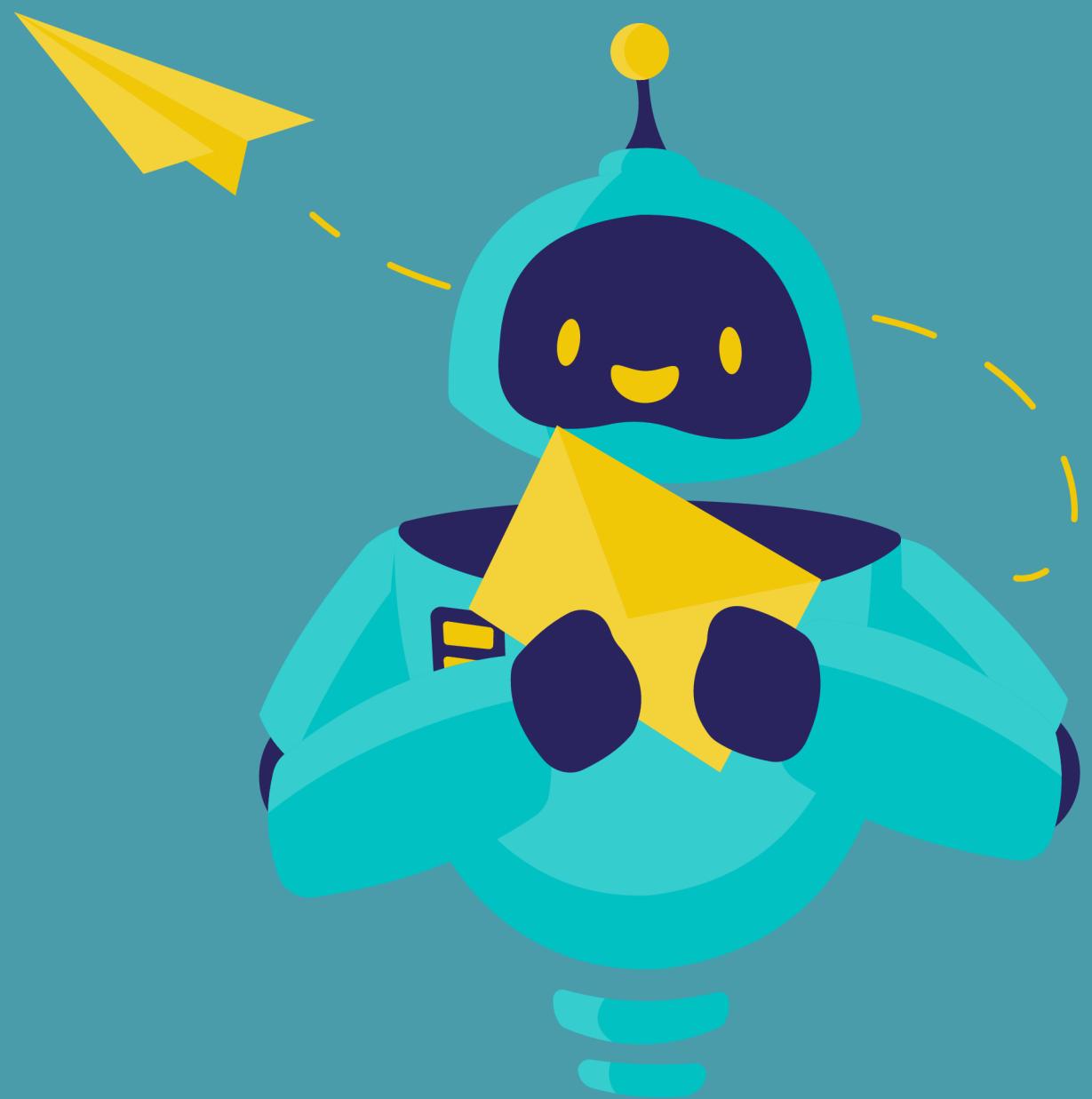
AUTOR PROJEKTU:
MAKSYMILIAN CELTNER

OPIEKUN:
DR HAB. INŻ. ROMAN ZAJDEL, PROF. PRZ

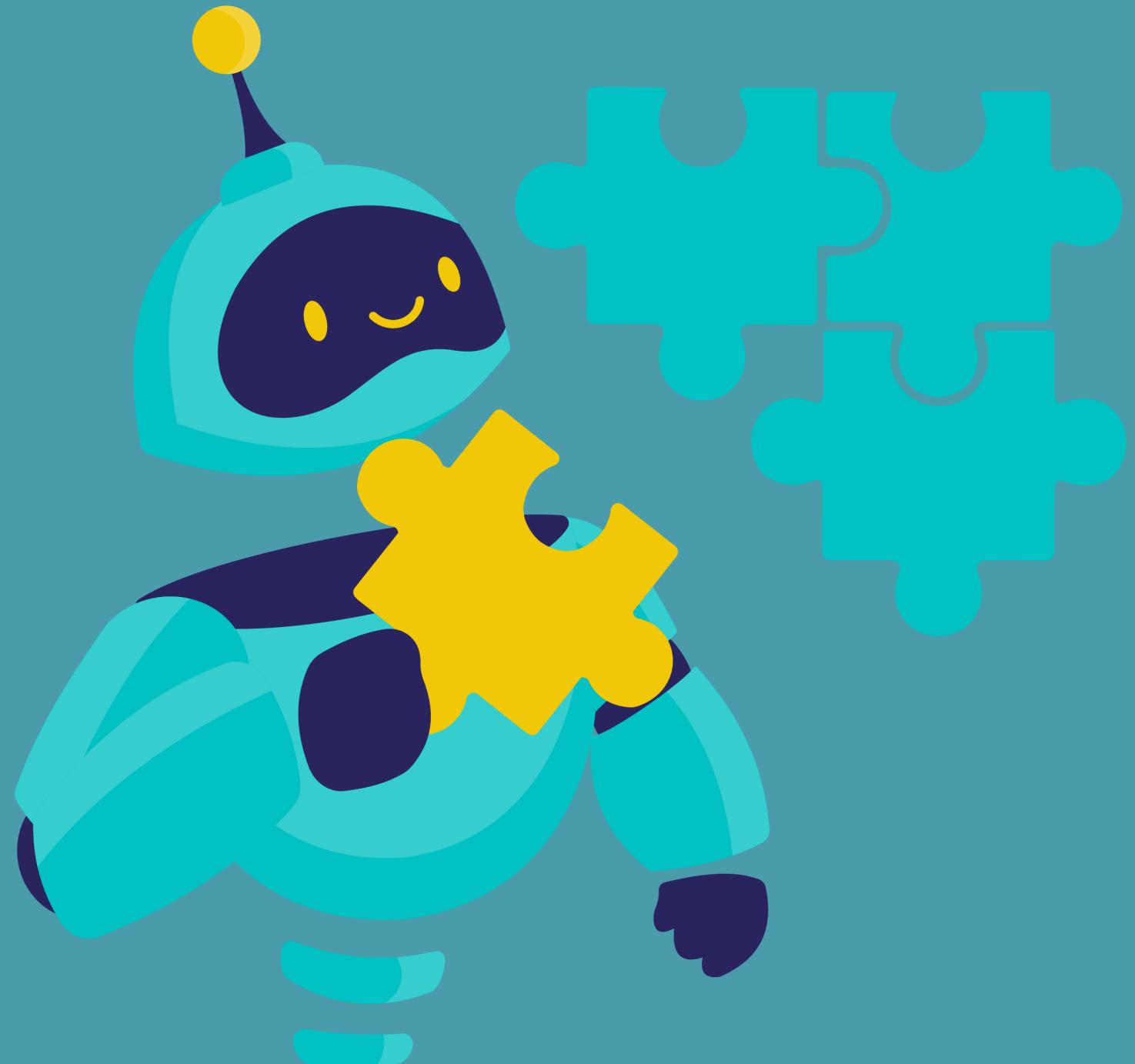


ZAWARTOŚĆ PREZENTACJI

- Wprowadzenie
- Roboty mobilne i ich symulacja
- Język Python w robotyce i symulacji
- Eksperymenty symulacyjne
- Perspektywa dalszych prac
- Napotkane problemy
- Porównanie symulatorów
- Wnioski i podsumowanie
- Wkład własny autora



WPROWADZENIE



Cel i zakres pracy:

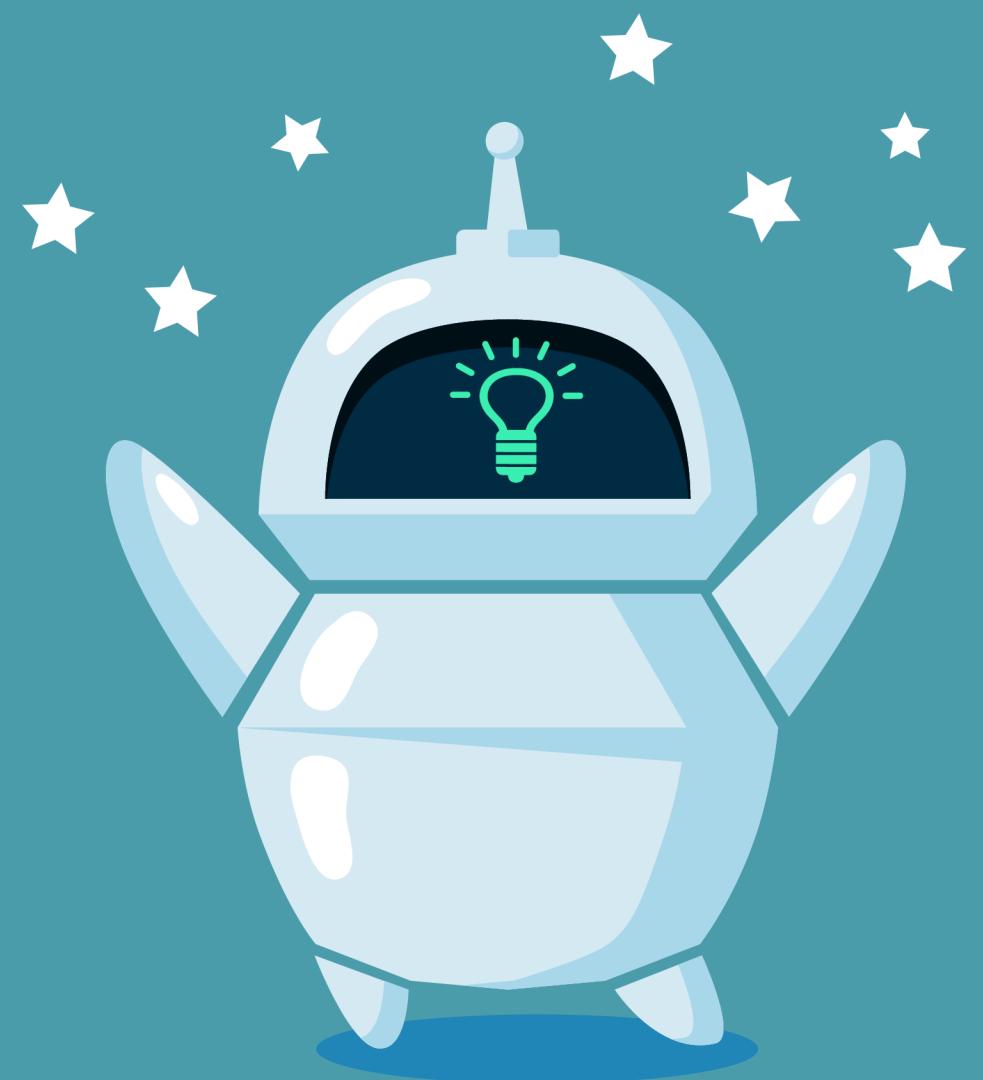
- Przegląd dostępnych symulatorów robotów mobilnych
- Wybór symulatorów, w których operacyjnym językiem jest Python
- Przeprowadzenie autorskich eksperymentów symulacyjnych
- Ocena symulatorów pod kątem funkcjonalności i integracji z Pythonem

ROBOTY MOBILNE I ICH SYMULACJA



Czym są roboty mobilne?

Roboty mobilne to maszyny zdolne do samodzielnego poruszania się w otoczeniu. Wykorzystywane są w przemyśle, logistyce, eksploracji, medycynie i badaniach naukowych. Ich zadania obejmują transport, inspekcję, nawigację i interakcję ze środowiskiem.

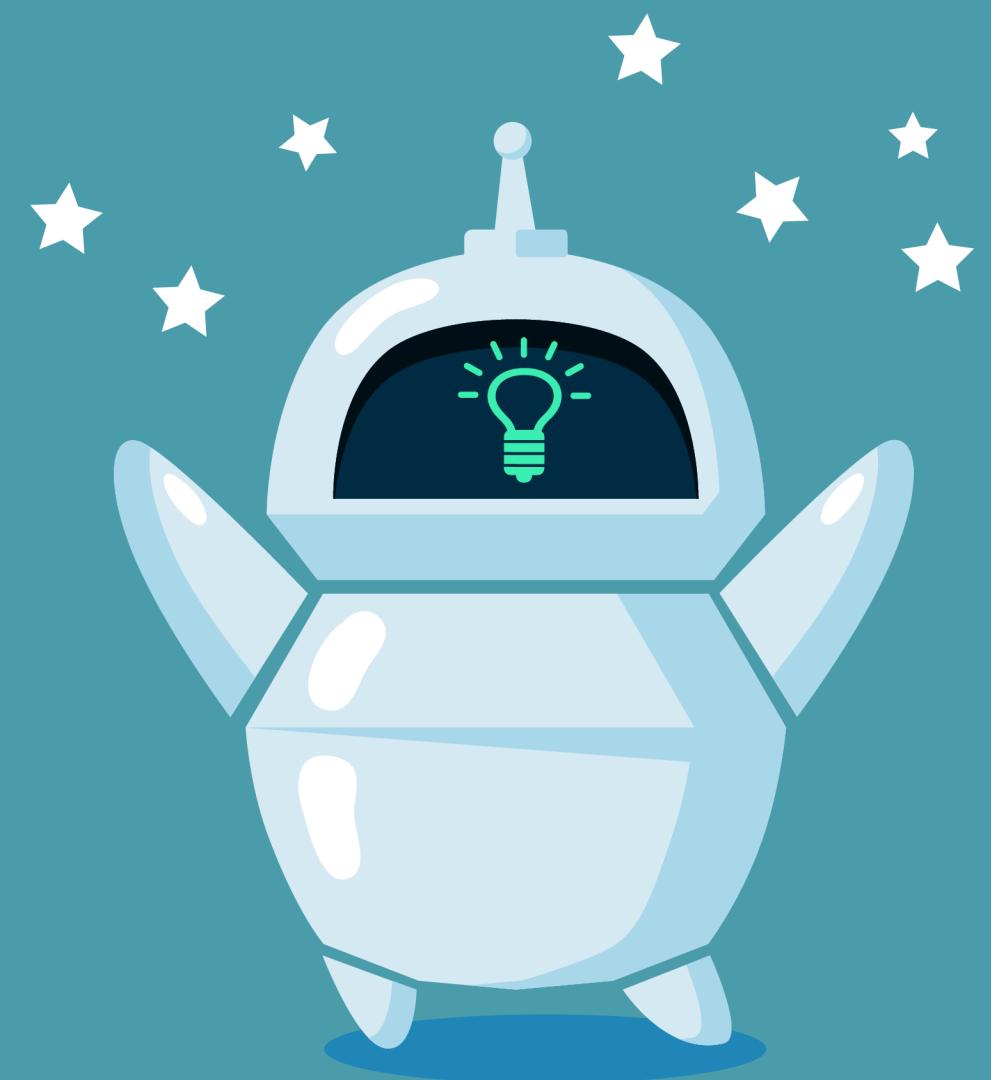


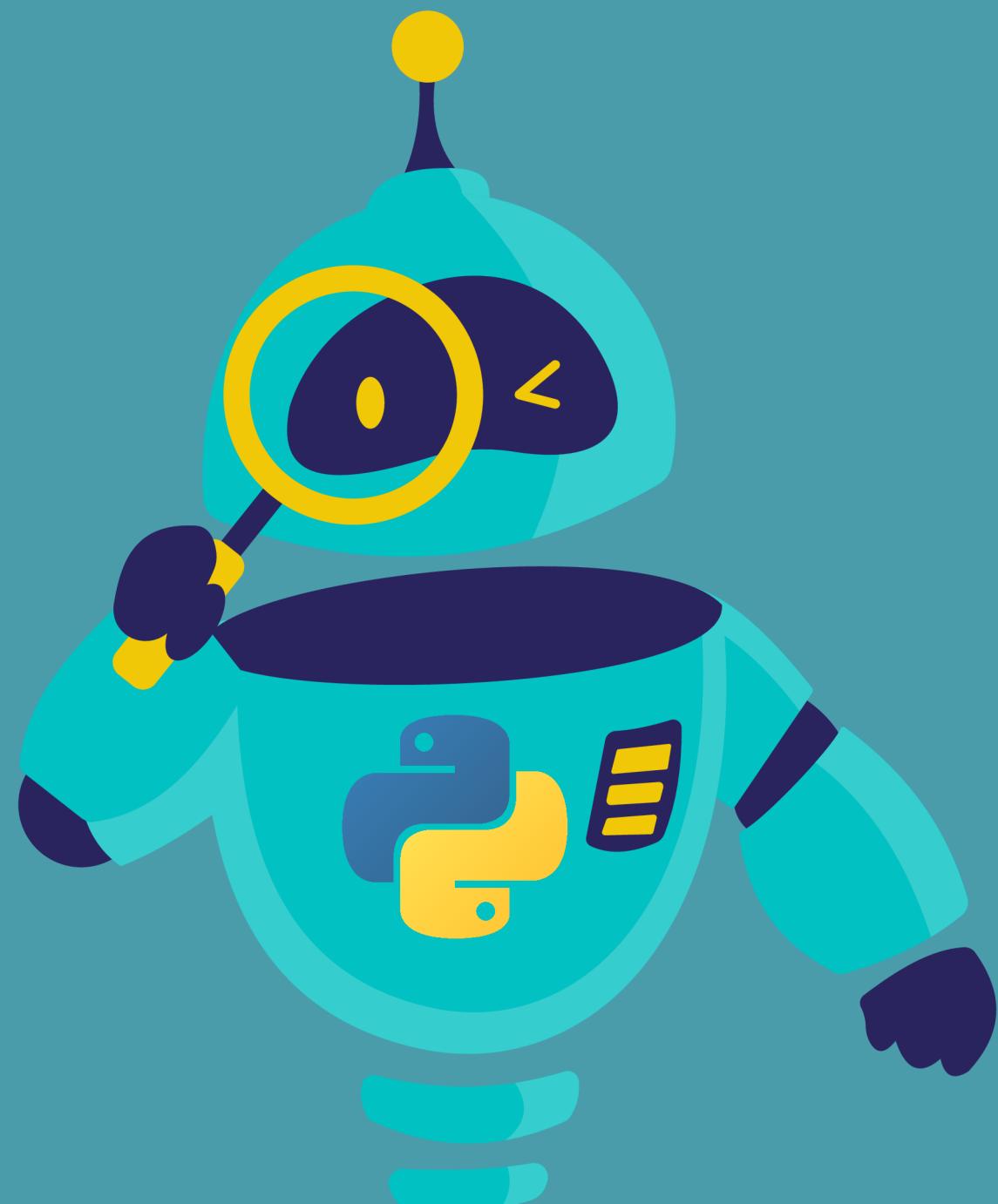
ROBOTY MOBILNE I ICH SYMULACJA



Znaczenie symulacji w robotyce mobilnej:

- Szybkie prototypowanie i testowanie różnych scenariuszy
- Eliminacja kosztów i ryzyka uszkodzenia sprzętu
- Eksperymentowanie w powtarzalnych warunkach
- Testowanie algorytmów nawigacji i sterowania
- Brak ograniczeń fizycznych



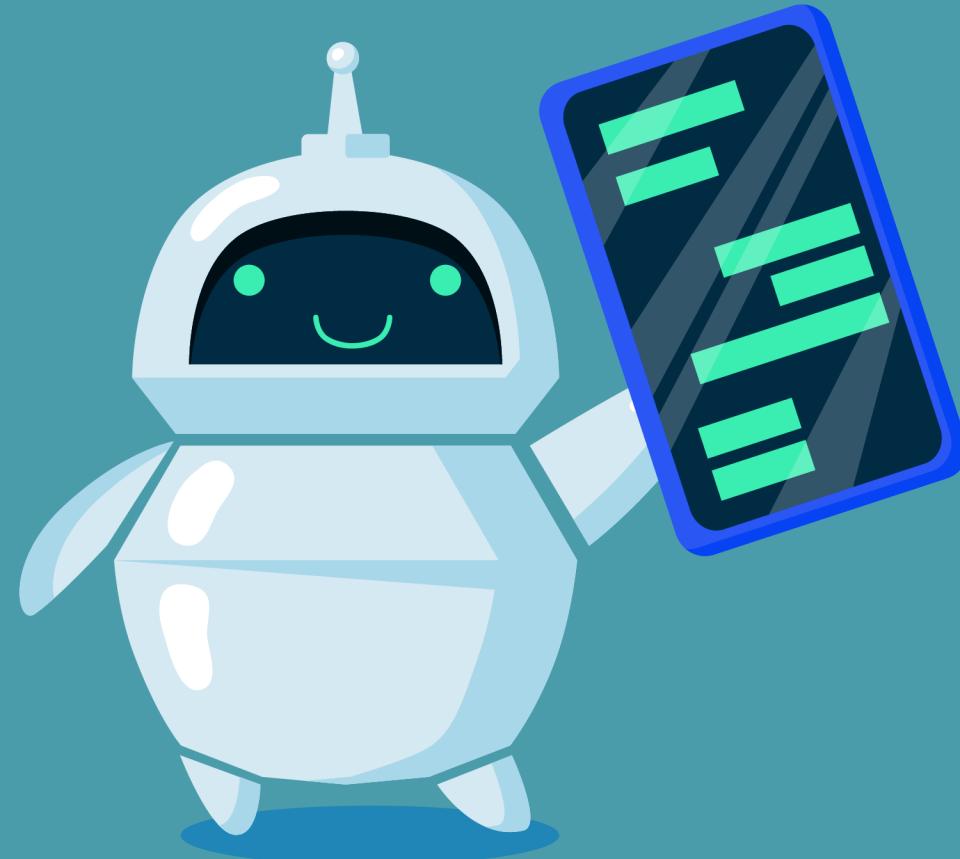
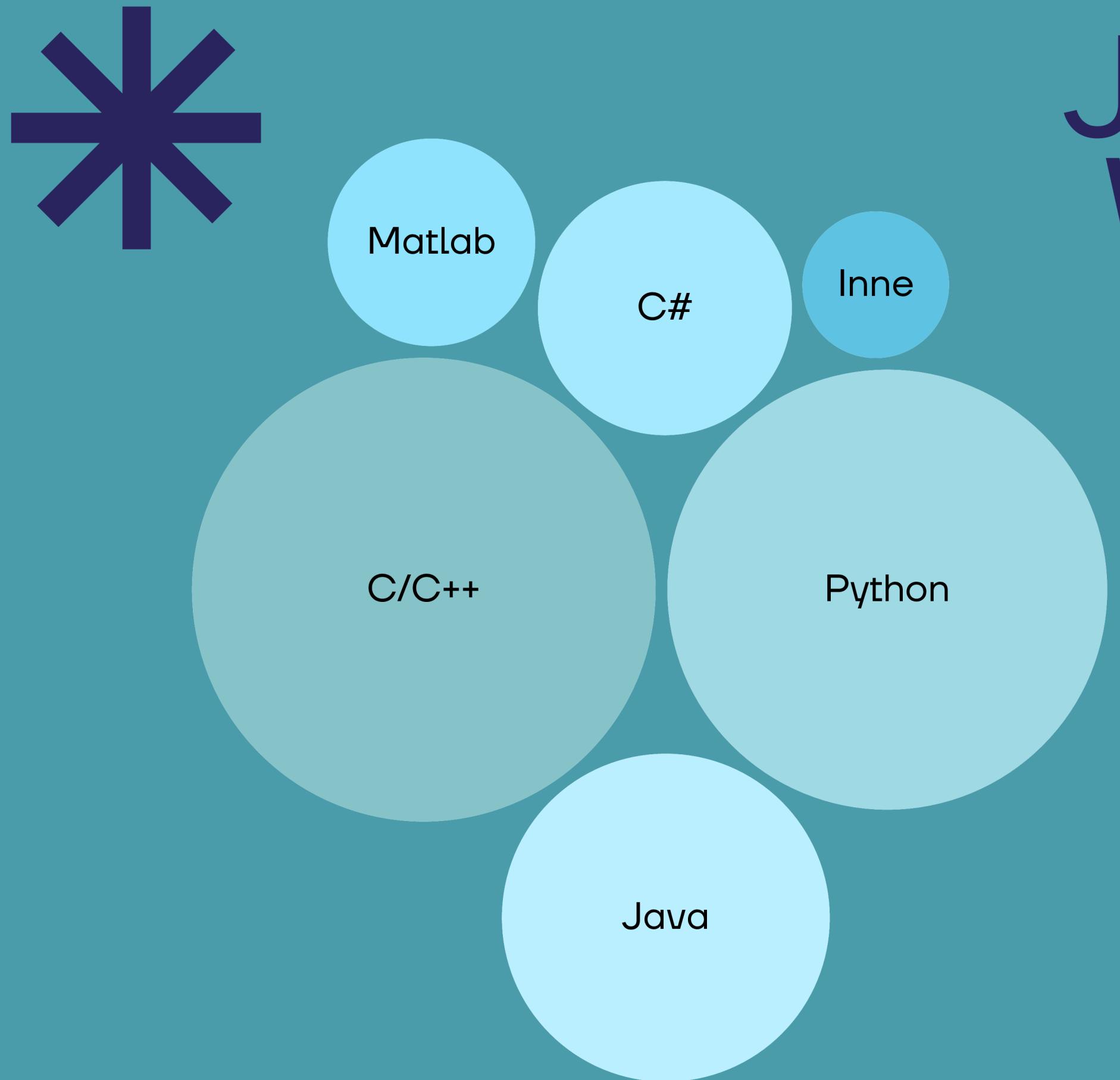


JEZYK PYTHON W ROBOTYCE I SYMULACJI

Dlaczego Python?

- Popularność w robotyce i sztucznej inteligencji
- Łatwość nauki i czytelność kodu
- Bogaty ekosystem bibliotek
- Doskonała integracja z symulatorami

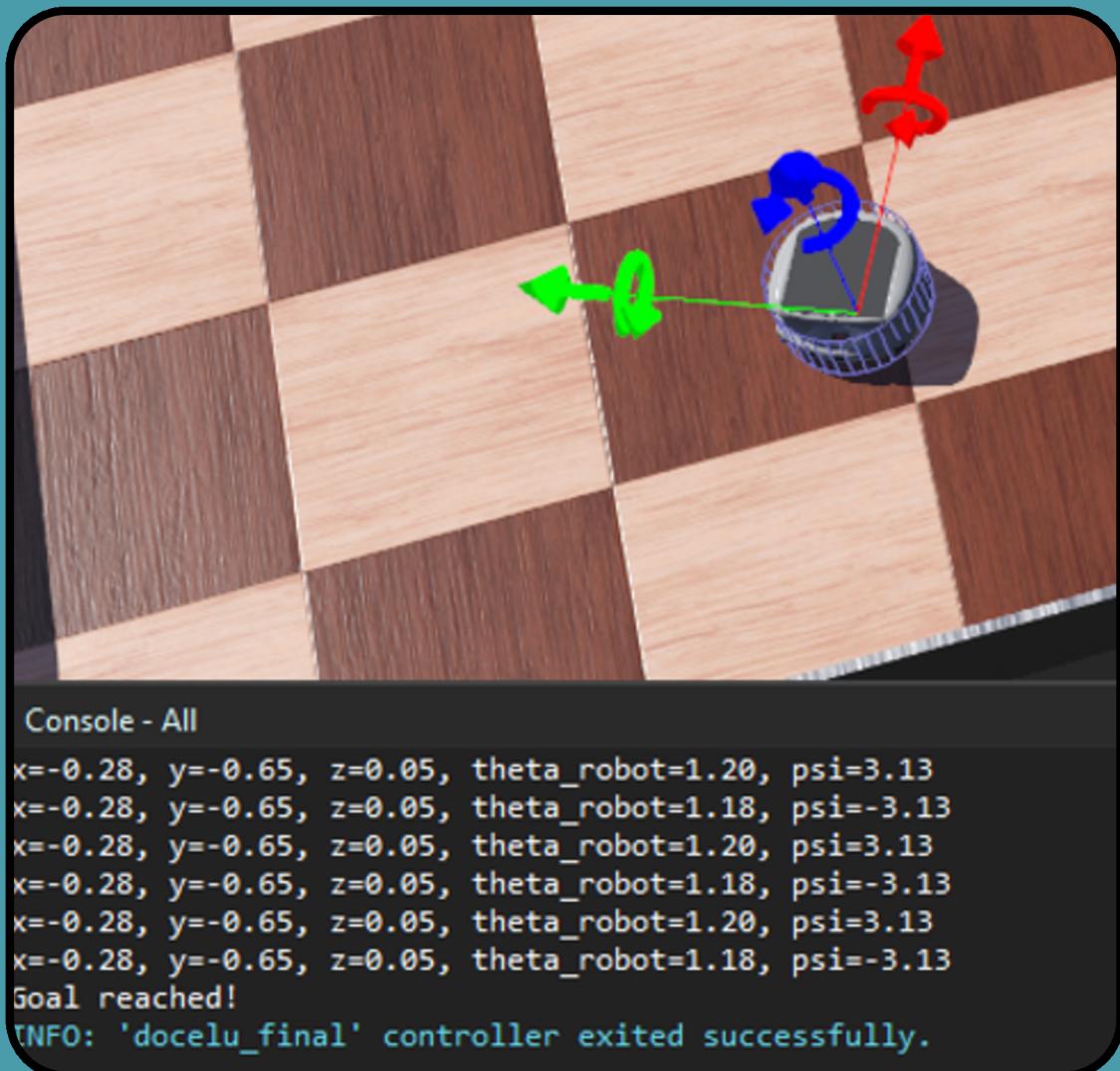
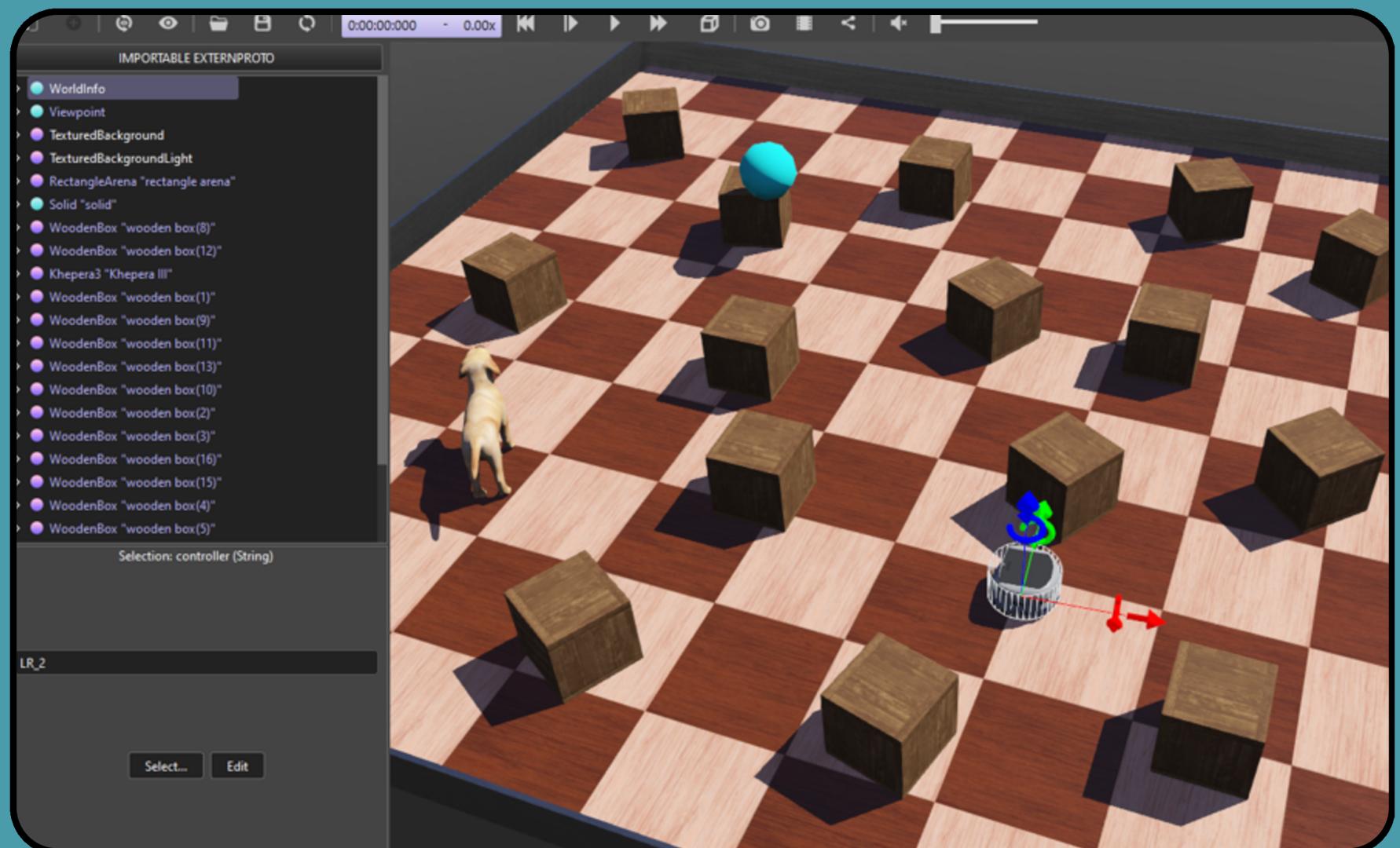
JEZYK PYTHON W ROBOTYCE I SYMULACJI



EKSPERYMENTY SYMULACYJNE

Webots - Omijanie przeszkód i podążanie do celu (Khepera III)

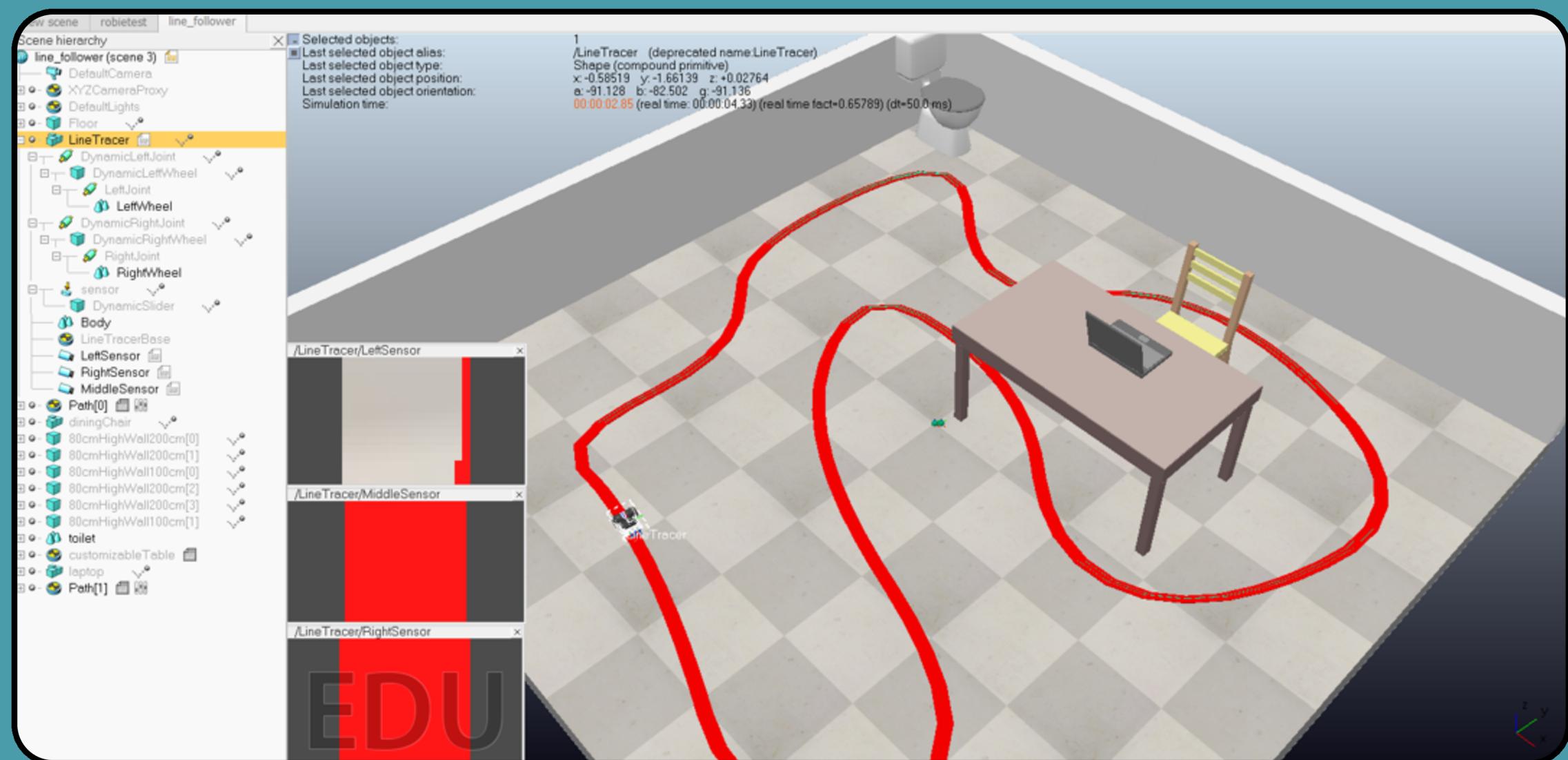
Robot Khepera III wykorzystuje algorytm logiki rozmytej do unikania przeszkód oraz nawigacji do wyznaczonego celu, bazując na danych z czujników odległości.



EKSPERYMENTY SYMULACYJNE

CoppeliaSim – Robot typu Line-Follower

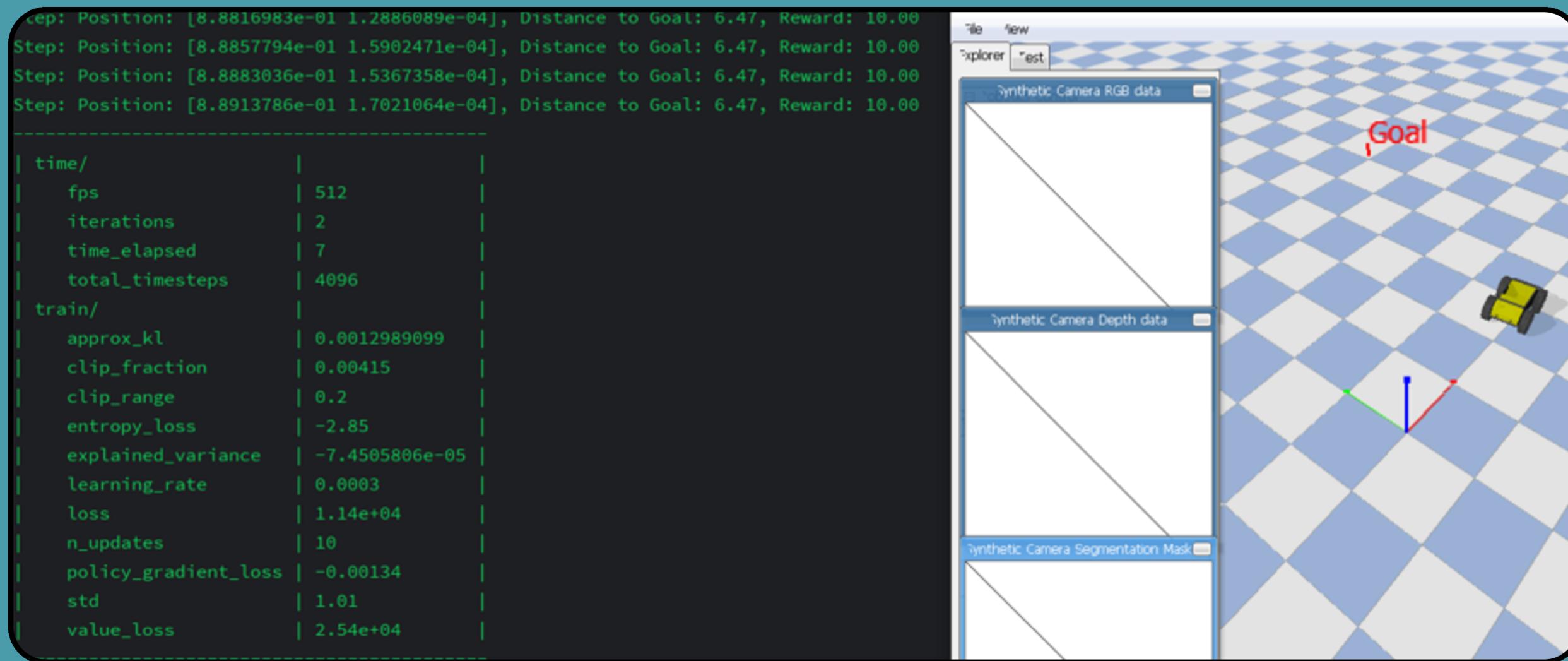
Symulacja robota mobilnego podążającego za linią, który analizuje dane z trzech czujników i dostosowuje prędkości kół do trajektorii linii.



EKSPERYMENTY SYMULACYJNE

PyBullet – Trenowanie robota do podążania do celu

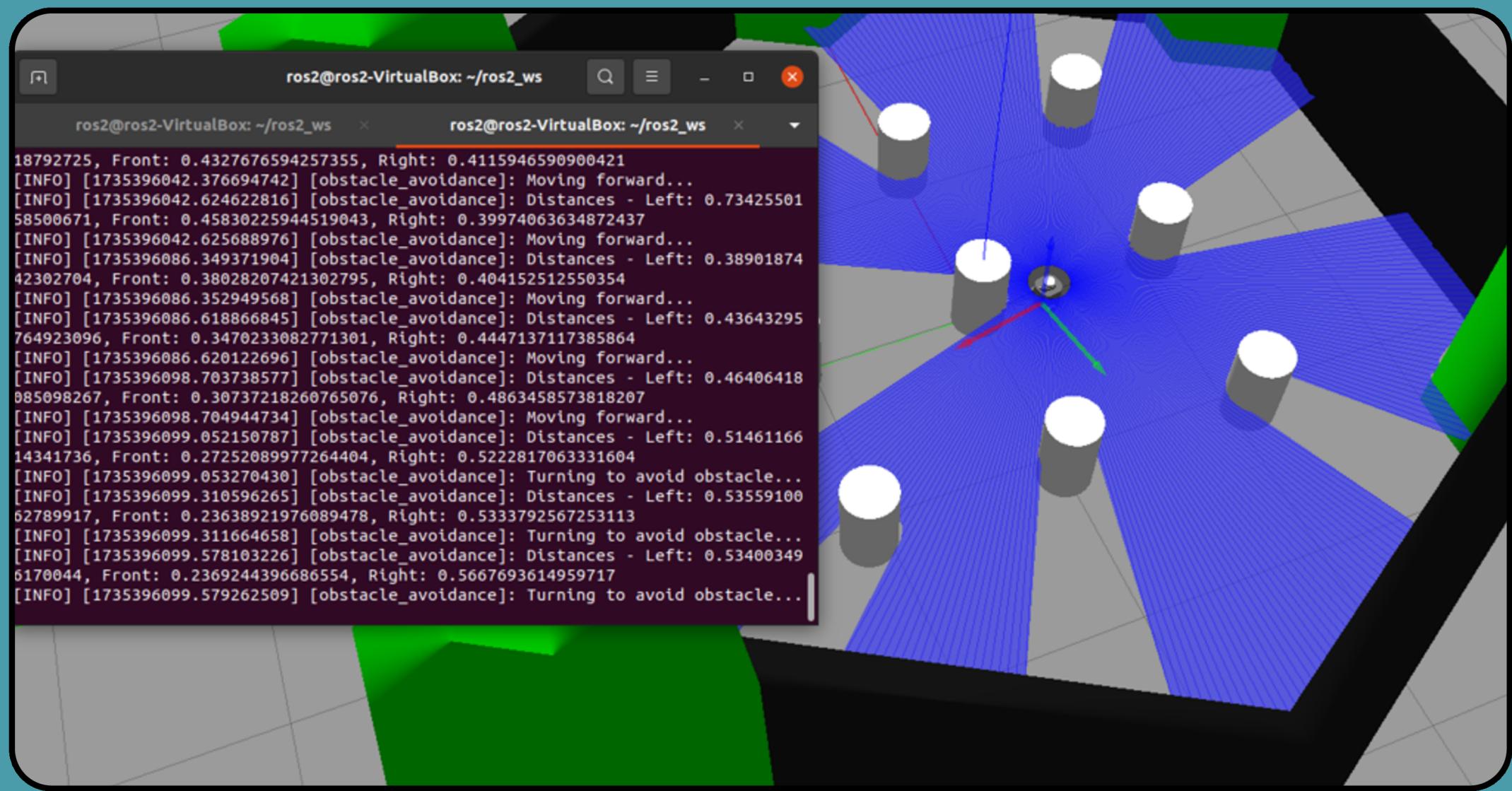
Autonomiczny robot Husky wykorzystuje uczenie ze wzmacnieniem (PPO), aby zoptymalizować sterowanie i efektywnie docierać do celu



EKSPERYMENTY SYMULACYJNE

Gazebo – Omijanie przeszkód z użyciem ROS

Robot TurtleBot3, korzystając z systemu ROS i danych sensorycznych, porusza się po planszy, analizuje otoczenie i omija przeszkody



PERSPEKTYWA DALSZYCH PRAC

- Rozwijanie utworzonych dotychczas symulacji
- Testowanie innych modeli robotów mobilnych wykonujących odmienne zadania
- Przetestowanie symulatorów dotychczas niebędących obiektem badań



NAPOTKANE PROBLEMY

- Nieintuicyjna konfiguracja interpretera Python w Webots i CoppeliaSim
- Liczne błędy w kodach sterujących
- Brak wiedzy i umiejętności obsługi symulatorów
- Nierozwiążane problemy dotyczące działania symulacji

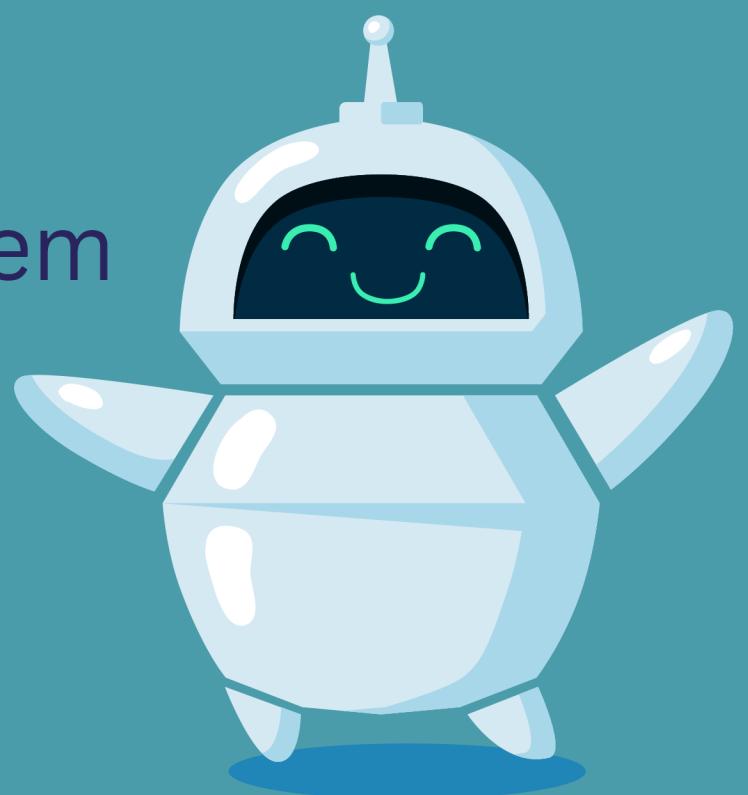


PORÓWNANIE SYMULATORÓW

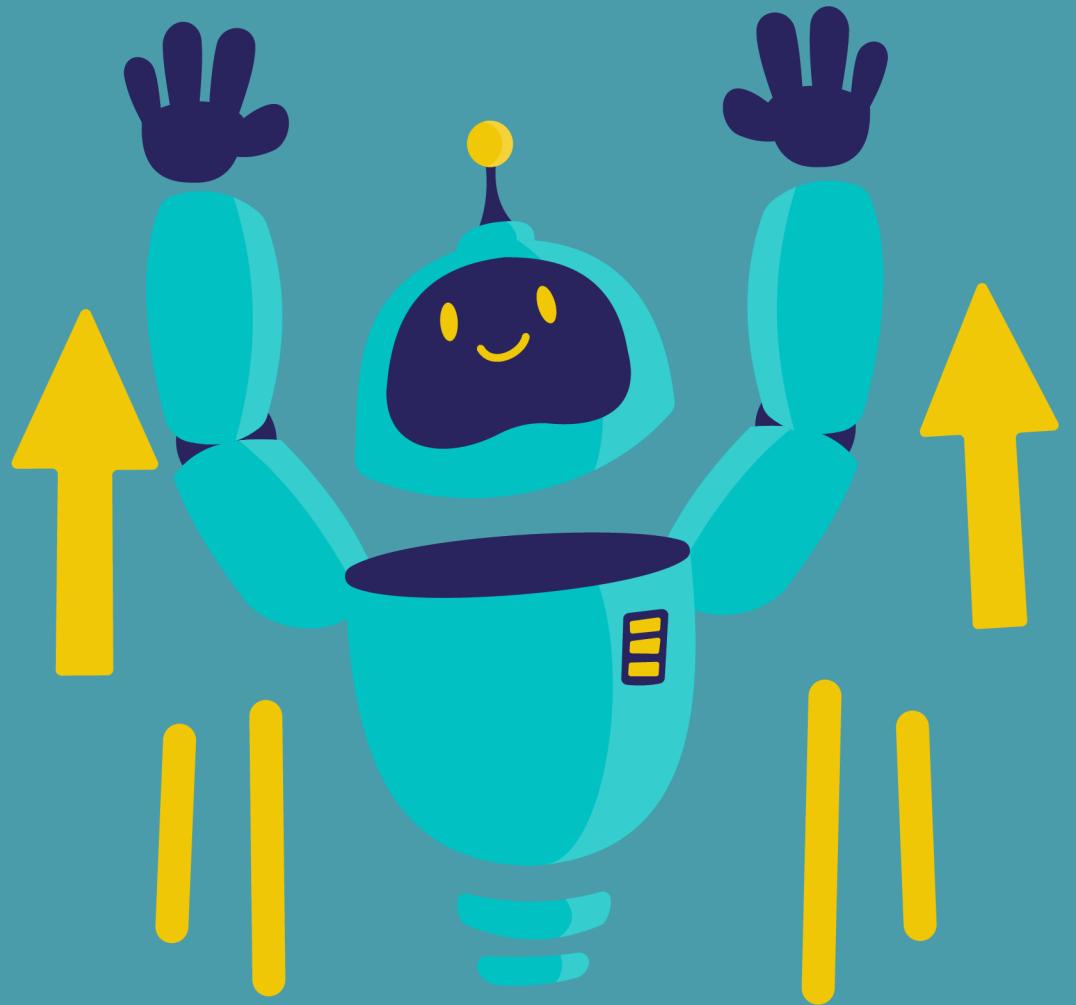
Kryterium	Webots	CoppeliaSim	PyBullet	Gazebo
Realizm fizyki	Wysoki	Średni	Średni	Bardzo wysoki
Interfejs użytkownika	Intuicyjny	Intuicyjny	Brak wbudowanego GUI, prosta wizualizacja	Techniczny, bardziej zaawansowany
Integracja z Pythonem	Wymaga konfiguracji interpretera	Wymaga konfiguracji interpretera	W pełni natywne – działa jako biblioteka Pythona	Poprzez ROS
Biblioteka modeli	Duża	Duża	Ograniczona	Obszerna
Tworzenie scenerii symulacyjnych	Łatwe	Łatwe	Wymaga programowania w Pythonie	Trudne – wymaga znajomości ROS i plików konfiguracyjnych
Wymagania sprzętowe	Wysokie - wymaga dobrej karty graficznej	Średnie	Niskie	Wysokie - wymaga dobrego procesora i RAM
System operacyjny	Windows, Linux, macOS	Windows, Linux, macOS	Windows, Linux, macOS	Głównie Linux

WKŁAD WŁASNY AUTORA

- Przegląd i selekcja dostępnych na rynku symulatorów robotów mobilnych w języku Python
- Konfiguracja wybranych symulatorów pod kątem działania z Pythonem
- Projektowanie scenerii symulacyjnych
- Pisanie i modyfikowanie kodów sterujących
- Analiza, porównanie i ocena symulatorów będących obiektem testów
- Stworzenie instrukcji do pracy z symulatorem Webots i uruchomienia symulacji omijania przeszkód na potrzeby przyszłych zajęć laboratoryjnych



WNIOSKI I PODSUMOWANIE



Który symulator wybrać?

Dla początkujących i edukacji → Webots, CoppeliaSim

Dla realistycznej fizyki i badań → Gazebo + ROS, Webots

Dla AI i uczenia maszynowego → PyBullet, Gazebo

Dla przemysłu i wdrożeń → Gazebo, CoppeliaSim



Nie ma jednego najlepszego simulatora – wybór zależy od projektu!

DZIĘKUJE ZA UWAGĘ

MAKSYMILIAN CELTNER

