# Sztuczna inteligencja - projekt

## 1. Cele projektu

- **Wybór najlepszego miejsca parkingowego** na podstawie parametrów:
  - Wymiary samochodu (szerokość, długość, promień skrętu).
  - Wymiary dostępnych miejsc parkingowych oraz ich współrzędne.
  - Położenie i wymiary innych pojazdów.
- **Parkowanie bezkolizyjne**, w tym unikanie kolizji z:
  - Liniami parkingowymi.
  - Innymi samochodami.
- **Efektywność manewru parkowania**:
  - Minimalizacja liczby korekt.
  - Zachowanie poprawnej pozycji końcowej samochodu.

---

## 2. Architektura projektu

1. **Symulacja i środowisko szkoleniowe**:
   - **Unity ML-Agents**:
     - Tworzenie środowiska 3D (symulacja parkingu, samochodów i miejsc parkingowych).
     - Definiowanie zachowania agenta i nagród (np. brak kolizji, poprawne parkowanie, wybór największego dostępnego miejsca).
     - Wizualizacja procesu szkolenia.
2. **Model AI**:
   - **PyTorch**:
     - Trenowanie modelu RL (Reinforcement Learning), np. PPO (Proximal Policy Optimization) lub DQN (Deep Q-Learning).
   - **OpenVINO**:
     - Optymalizacja wytrenowanego modelu AI dla szybszego działania w czasie rzeczywistym.
     - Użycie na platformach z ograniczonymi zasobami (np. systemy embedded).
3. **Integracja komponentów**:

- Symulacja (Unity) komunikuje się z modelem PyTorch w czasie szkolenia przez REST API lub sockety.
- Gotowy model zaimplementowany w symulacji działa na OpenVINO.

---

## 3. Kroki realizacji projektu

1. **Etap przygotowania środowiska**:
   - Stworzenie sceny parkingowej w Unity:
     - Wyznaczenie miejsc parkingowych, pozycji innych samochodów.
     - Dodanie parametrów samochodu (wymiary, kąt skrętu kół).
   - Przygotowanie ML-Agents:
     - Definicja agenta (obserwacje, działania, nagrody).
     - Obserwacje: wymiary, pozycje miejsc parkingowych, odległość od innych aut, sensory do wykrywania kolizji.
2. **Trenowanie modelu**:
   - Budowa i trenowanie modelu AI:
     - W Unity ML-Agents: trenowanie agenta z wykorzystaniem lokalnych nagród (np. brak kolizji, poprawne parkowanie).
     - Eksport wytrenowanego modelu do formatu kompatybilnego z PyTorch.
   - Optymalizacja modelu w OpenVINO.
3. **Testowanie i wdrażanie**:
   - Weryfikacja wyników symulacji w Unity.
   - Testowanie zoptymalizowanego modelu w scenariuszach czasu rzeczywistego.

---

## 4. Technologie i narzędzia

- **Unity + ML-Agents**:
  - Silnik symulacyjny do nauki i testowania modelu.
- **PyTorch**:
  - Framework do tworzenia i trenowania modeli AI.
- **OpenVINO**:
  - Narzędzie do optymalizacji modeli do zastosowań w czasie rzeczywistym.
- **Python**:
  - Obsługa komunikacji między Unity i PyTorch/OpenVINO.

- **REST API / Sockety**:
  - Wymiana danych między Unity a Pythonem (PyTorch).

---

## 5. Nagrody w procesie uczenia agenta

1. **Pozytywne nagrody**:
   - Poprawny wybór miejsca parkingowego.
   - Uniknięcie kolizji.
   - Parkowanie w całości wewnątrz wyznaczonego miejsca.
   - Zakończenie manewru w minimalnej liczbie kroków.
2. **Kary**:
   - Kolizje z innymi pojazdami lub liniami.
   - Przekroczenie obszaru miejsca parkingowego.
   - Zbyt duża liczba korekt.