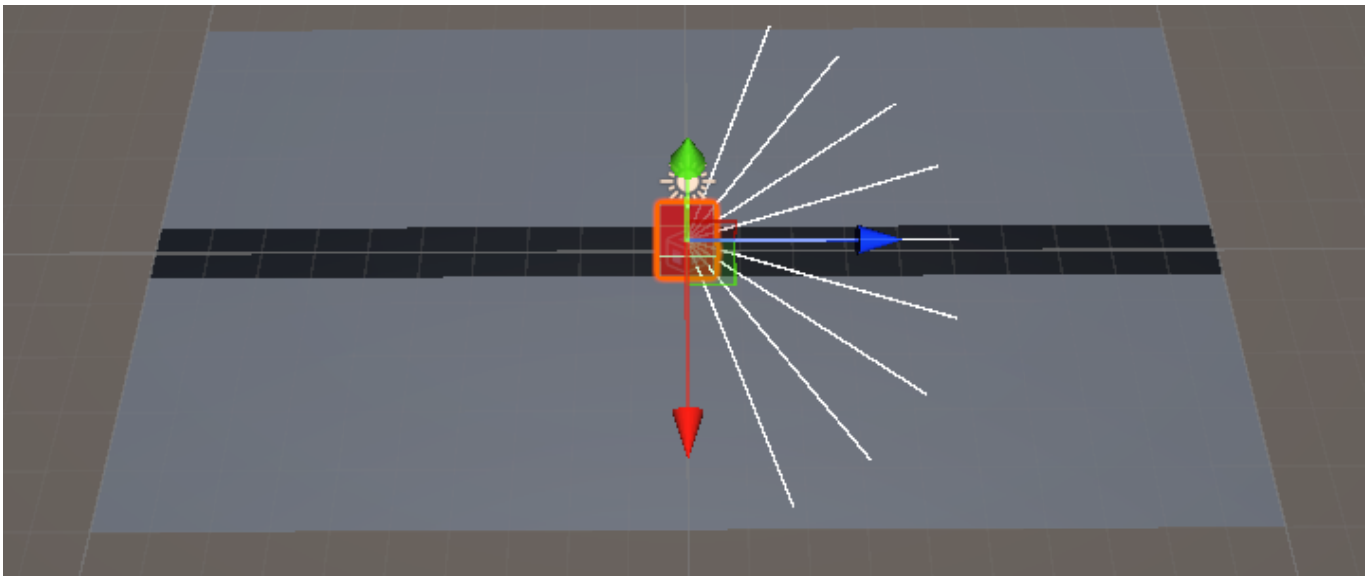
Rapport: AI Jumper exercise

Inleiding

Dit rapport beschrijft het proces en resultaat van een oefening waarbij een kunstmatige intelligentie (AI) getraind wordt om automatisch over een balk te springen. Het doel van dit rapport is om inzicht te geven in hoe een AI kan worden ontwikkeld om eenvoudige motorische taken uit te voeren in een gesimuleerde omgeving.

Deze oefening is bedoeld als leerervaring voor ons, om vertrouwd te raken met het ontwerpen, trainen en evalueren van AI-modellen in interactieve scenario's. Door dit probleem te onderzoeken, leren we meer over besluitvorming, beweging en obstakelvermijding binnen AI-systemen. Het rapport dient daarom zowel als verslag van het project als bewijs van ons opgedane kennis en vaardigheden.

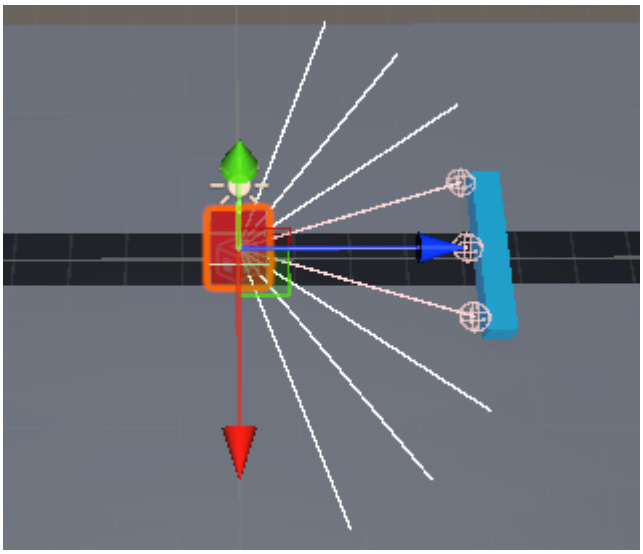
Structuur



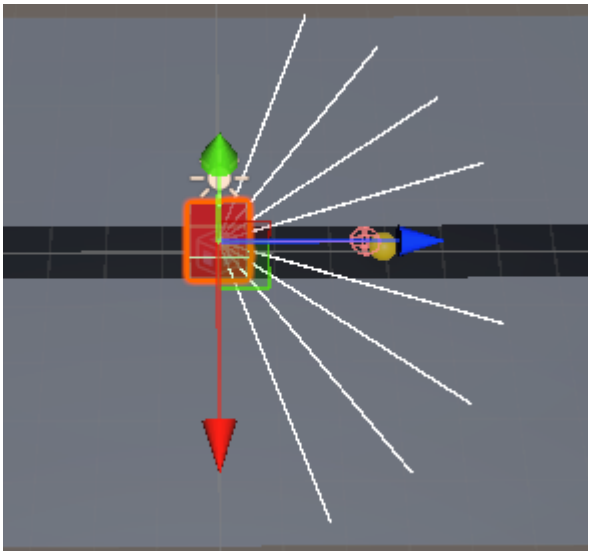
Een plane met een Cube(speler), een Road voor obstakels(Obstacles) of beloningen(Coins) die naar de speler 1 voor 1 bewegen en een Wall die de Coins en Obstacles opvangen.

De cube is de agent en heeft dus daarom deze componenten:

1. Behavior Parameters
2. Jump Agent (Controller script)
3. Ray Perception Sensor 3D
4. Decision Requester



De speler die een wall detecteerd. Waar het uiteindelijk over een tijd trainen, over gaat springen.



De speler die een coin detecteerd. Waar het uiteindelijk gaat tegen lopen om een beloning te krijgen.

Materialen

1. Unity
2. Tensorboard
3. ML agents
4. Anaconda

Methoden

1. JumpAgent.cs

Dit script hangt aan de Agent vast(Cube).

Methoden:

1. Start():
   Deze methode wordt één keer automatisch uitgevoerd bij het begin van het spel. Ze haalt de Rigidbody op van de agent zodat die later fysiek kan bewegen, zoals springen.

2. OnEpisodeBegin():
   Wordt automatisch gestart aan het begin van elke trainingsronde (episode). Het roept InitializeEnvironment() aan om de omgeving te resetten.
3. InitializeEnvironment():
   Zet de agent terug op de startpositie, verwijdert oude obstakels en plaatst willekeurig een nieuw obstakel of munt. Zo krijgt de AI telkens een nieuwe uitdaging.
4. Success():
   Wordt aangeroepen wanneer de agent succesvol het obstakel vermijdt. De agent krijgt dan een positieve beloning en de episode eindigt.
5. OnTriggerEnter(Collider other):
   Detecteert botsingen met objecten. Als de agent een obstakel raakt, krijgt hij strafpunten; raakt hij een munt, dan krijgt hij een beloning. Daarna stopt de episode.
6. CollectObservations(VectorSensor sensor):
   Verzamelt informatie over de positie van de agent. Deze observatie wordt gebruikt als input voor de AI om beslissingen te nemen.
7. OnActionReceived(ActionBuffers actionBuffers):
   Voert de actie uit die de AI kiest. Als de actie een sprong is, en de agent is op de grond, dan springt hij.
8. Heuristic(in ActionBuffers actionsOut):
   Geeft handmatige controle voor testen. Als de spatiebalk wordt ingedrukt, wordt een sprongactie gesimuleerd (voor manueel besturing).

1. ObstacleMovement.cs

Dit script hangt aan de objecten die naar de speler gaan dus Coins en Obstacles.

Methoden:

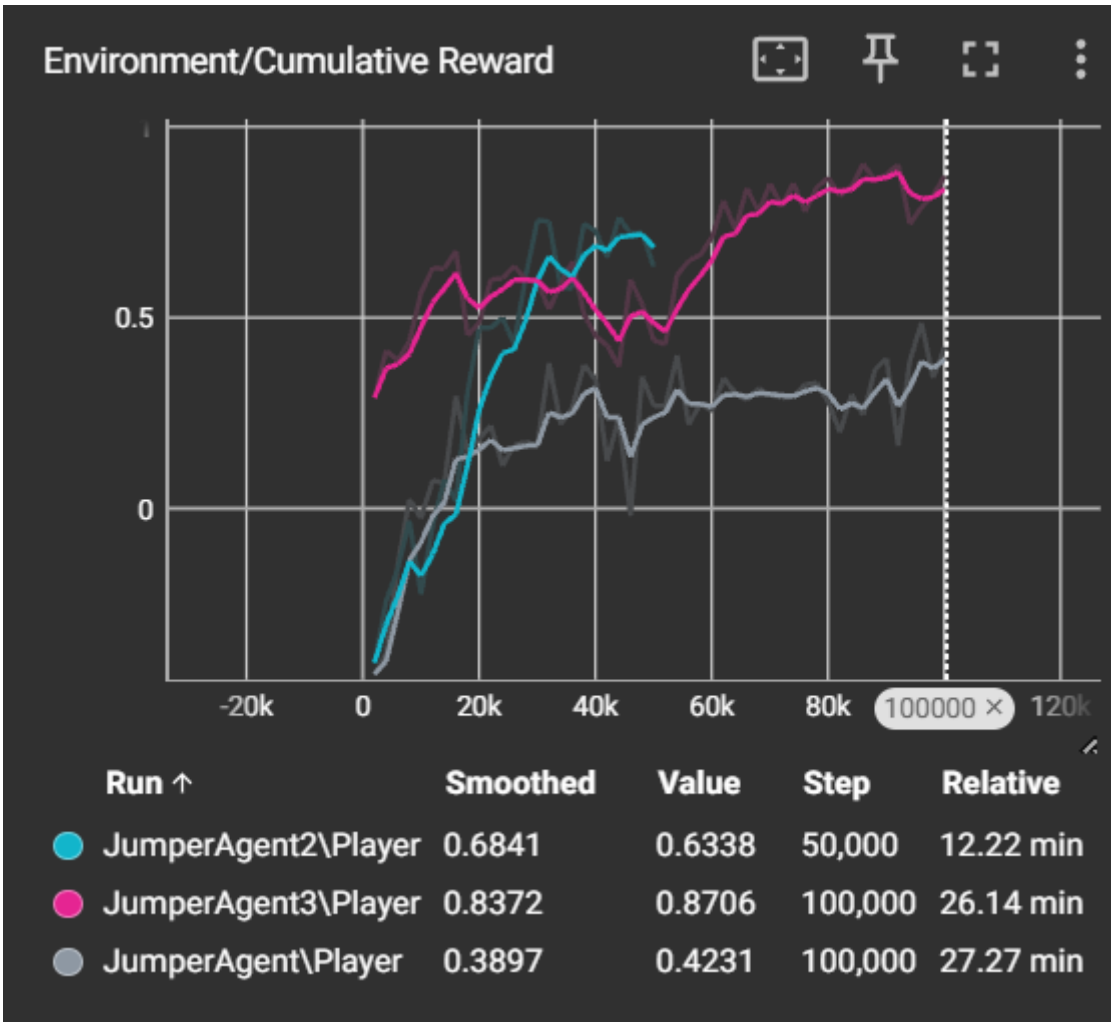
1. Update():
   Laat het obstakel naar voren bewegen. Als het voorbij een bepaald punt is, wordt de agent beloond via jumpAgent.Success(), dit betekent dat hij het obstakel succesvol heeft vermeden.
2. SetRandomSpeed(float minSpeed, float maxSpeed):
   Geeft het obstakel een willekeurige snelheid tussen een minimum en maximum waarde. Dit zorgt voor variatie in elke trainingsronde.

Variabelen

De belanrijkste variabelen zijn:

1. Behavior type:
   - Default: Voor het uitvoeren van de Agent.
     Of
   - Heuristic Only: Voor de manuele controle van de Cube.
2. Obstacle Prefab: Hier kan je een obstakel object meegeven.
3. Coin Prefab: Hier kan je een beloning object meegeven.
4. Jump Force: Hier kun je beslissen hoe hoog de Cube kan springen.

Resultaten



Run ↑	Smoothed	Value	Step	Relative
JumperAgent2\Player	0.6841	0.6338	50,000	12.22 min
JumperAgent3\Player	0.8372	0.8706	100,000	26.14 min
JumperAgent\Player	0.3897	0.4231	100,000	27.27 min

Run ↑	Smoothed	Value	Step	Relative
JumperAgent2\Player	13.1992	13.0845	50,000	12.22 min
JumperAgent3\Player	11.0635	10.7647	100,000	26.14 min
JumperAgent\Player	11.9804	11.8269	100,000	27.27 min

Conclusie

In dit project werd een AI-agent getraind om automatisch over obstakels te springen met behulp van Unity ML-Agents. De agent leert via trial-and-error op basis van beloningen en straffen. Na meerdere trainingsrondes toont de agent verbetering in zijn gedrag en springt hij vaker succesvol over obstakels.

Hoewel de training in een eenvoudige omgeving plaatsvond, toont dit experiment hoe reinforcement learning werkt in de praktijk. Mogelijke uitbreidingen zijn het toevoegen van meerdere obstakels, variabele springhoogtes, of een complexer landschap om de uitdaging te vergroten.