howest
/ we develop people

Research Project & Bachelor Proef

Zelfrijdende AI met routebeschrijving

Gebruikershandleiding

Team Carlier Alex

Module Research Project

Richting Multimedia and Creative Technology (MCT)

Schooljaar 2021-2022

1 – Projectorganisatie

Dit project werd ontwikkeld als een simulatie omgeving voor het trainen en draaien van AI-modellen voor zelfrijdende auto's. Het maakt gebruik van diverse technologieën die hieronder staan opgesomd. Voor de installatie van deze technologieën verwijs ik u naar de gebruikershandleiding.

- Unity Hub en Unity (versie 2020.3.25f1)
- Python
- Pytorch en Tensorboard
- Unity ML-Agents Toolkit

1.1 – Projectorganisatie – Folders

De broncode en assets van het project zijn gestructureerd in verschillende folders

Assets	Bevat Unity assets voor de omgeving zoals prefabs, models, scripts en andere assets
builds	Directory om gecompileerde builds van Unity scenes op te slaan
config	Bevat configuratie files (in YAML formaat) voor de Reinforcement Learning algoritmes van de toolkit
results	Resulltaten van de trainingsessies
training	Python code voor externe Reinforcement Learning algoritmes te trainen (WIP)

De Assets folder is ook verder opgesplitst.

ML-Agents	Package van de Unity ML-Agents Toolkit
Models	Getrainde modellen om te gebruiken in de scenes
Prefabs	Opgeslagen gameobjecten om te hergebruiken en voor makkelijk onderhoud
Scenes	Opgeslagen scenes
Scripts	Script files met classes en code

2 – Scene & Omgeving

Unity scenes zijn virtuele scenario's die allerlei objecten bevatten waaraan scripts kunnen worden gekoppeld. Een scene creëren is vrij simpel en kan onmiddellijk draaien binnen de editor of gecompileerd naar een build met een uitvoerbaar bestand.

De traingingsomgevingen voor dit project bestaan uit meerdere objecten die gekoppeld zijn aan elkaar en samen één geheel vormen. De belangrijkste objecten in de scene staan hieronder opgesomd.

Environment	Virtuele omgeving voor de agent dat bestaat uit meerdere tiles die het wegdek en obstakels representeren
GPS	Staat in voor de A*pathfinding te genereren in de scene waarbij de positie van het GPS object als startpunt dient voor het pathfinding grid
Goal	Eindbestemming van de agent
Start	Bevat startposities voor de agent
CarAgent	De agent die acties zal nemen in de omgeving

Deze traingingsomgevingen zijn opgeslagen als prefabs. Een object als prefab opslaan heeft meerdere voordelen:

- Aanpassingen aan de originele prefab worden overgedragen naar alle instanties (zowel voor scenes als binnen andere prefabs)
- Herbruikbaar over meerdere scenes
- Gemakkelijk te dupliceren

Wees wel zeer voorzichtig bij het omgaan met de prefabs. Het verwijderen of aanpassen van de originele prefab heeft een effect op alle instanties en varianten.

2.1 – Scene & Omgeving – Agent object

De agent is een object dat bestaat uit meerdere kleinere objecten waaraan scripts gekoppeld zijn. Het is ook opgeslagen als prefab om gebruik te versimpelen.

Body	Het 3D-model van de agent (afkomstig van de geïmporteerde asset packages)
WheelColliders	De colliders voor de wielen met scripts die instaan voor de besturing van elk wiel en parameters zoals frictie en suspensie. De wielen zorgen voor de beweging van de agent.
RaySensorCar	Zorgt voor de Raycast sensor observaties

Meerdere scripts en componenten zijn ook rechtstreeks gekoppeld aan het agent object. Deze componenten zijn afkomstig van Unity, de ML-Agents toolkit of zijn zelf-gedefinieerd.

Box Collider	Staat in voor collisie tussen de agent en zijn omgeving
Rigidbody	Component dat voor fysica simulaties op de agent
Decision Requester	Toolkit script dat instaat voor het nemen van steps tijdens de training
Behavior Parameters	Toolkit script dat het aantal observaties, type acties en bepaalde configuratie parameters definieert zoals de Behavior Name
Car Agent (script)	Script dat overerft van de Toolkit's Agent class en alle code bevat omtrent de definitie van de agent
Car Agent Wheel	Staat in voor de besturing van de auto en heeft controle over de Wheelcolliders

2.2 – Scene & Omgeving – Agent script

Het CarAgent script definieert de agent. De code is verdeeld over meerdere methodes en classes om alles beheersbaar en uitbreidbaar te houden.

De belangrijkste methodes in deze class zijn overgeërfd van de Toolkit's Agent class en staan in voor interacties van de agent met zijn omgeving.

Initialize	Definieert wat er moet gebeuren bij de start van de trainingssimulatie
OnEpisodeBegin	Definieert wat er moet gebeuren bij de start van elke episode
CollectObservations	Toevoegen van observaties voor elke step in de training.
Heuristic	Manuele besturing van de agent. Dit wordt voornamelijk gebruikt om de besturing van de agent te controleren.
OnActionReceived	Acties om uit te voeren op basis van de outputs van het model

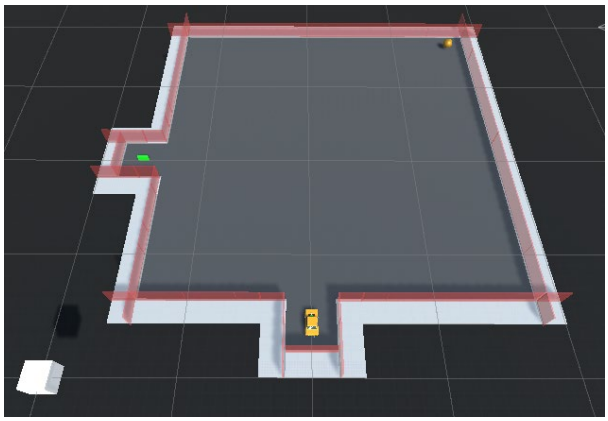
Daarnaast zijn er andere classes die code onderhouden voor de agent en de omgeving.

ConfigAgent	Configuratie parameters voor de agent. Worden ook gebruikt bij berekenen van rewards
ConfigReward	Configuratie voor de waardes van rewards
Config	Files die overervende classes bevatten van de ConfigAgent en ConfigReward voor bepaalde versies van de agent configuratie
EnvController	Om de omgeving de initialiseren en resetten
GoalController	Beheerst de opstelling van de eindbestemming voor elke episode
StartController	Beheerst de opstelling van de startposities van de agent voor elke episode
GPS	Extern script dat instaat als GPS voor de agent en een route berekent naar de eindbestemming
Rewards	Functies voor het berekenen van de rewards van de omgeving voor de agent

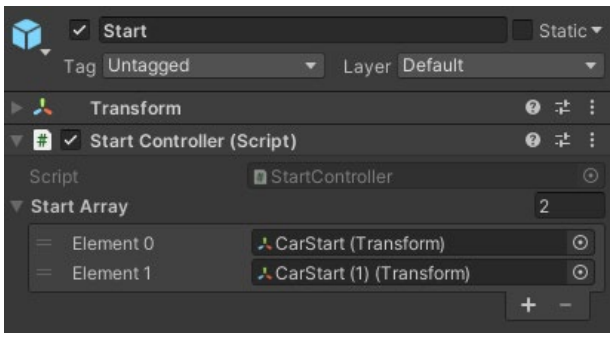
2.3 – Scene & Omgeving – Gebruik

Voor het opstellen van een trainingsomgeving moet eerst een object voor de omgeving zelf aangemaakt worden. U kunt gebruik maken van een bestaande prefab of zelf een object opstellen via de volgende stappen:

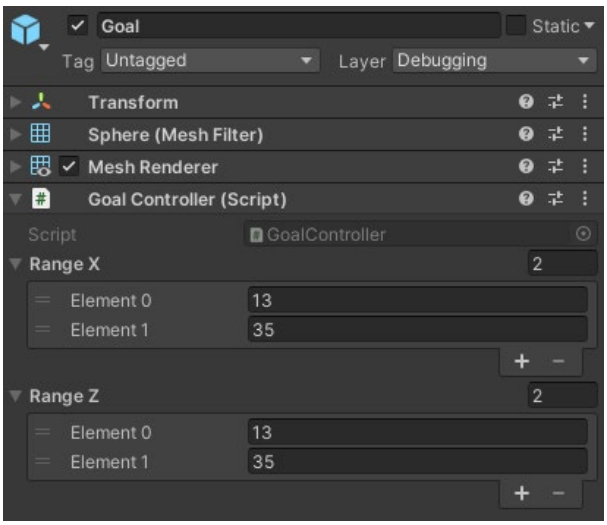
- Creëer een nieuwe omgeving of kies een bestaande omgeving.



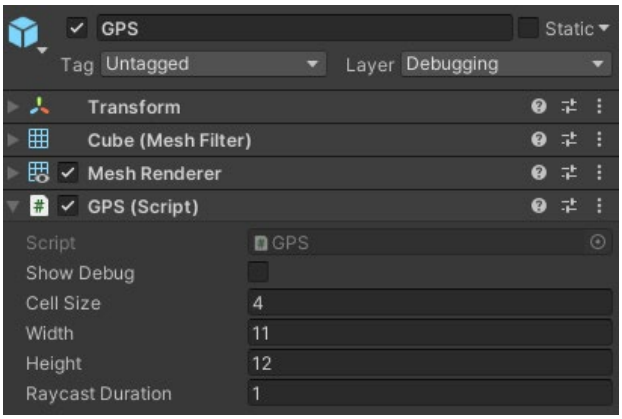
- Geef de mogelijke startpositie objecten mee aan het Start Controller component van het Start object. Deze worden gebruikt om de positie en rotatie van de agent te initialeren bij het begin van elke episode.



- Definieer de limieten waar de eindbestemming zich kan bevinden in het Goal Controller component van het Target object (minimale en maximale lokale X en Z positie in de scene)



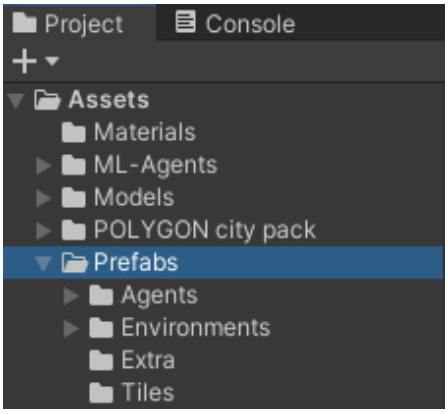
- Plaats de GPS op de rand van de omgeving als startpunt voor het pathfinding grid en definieer een bepaald aantal rijen (height), kolommen (width) en breedte van de grid cellen in het GPS component.



- Zorg dat deze gezamenlijk worden gebonden aan een gameobject als child objecten.

- Geef alle voorgaande controllers mee aan het Env Controller component van het parent object.

- Sla dit object op als een nieuwe prefab (door simpelweg het object te slepen in de *Assets/Prefabs/Environments* directory).



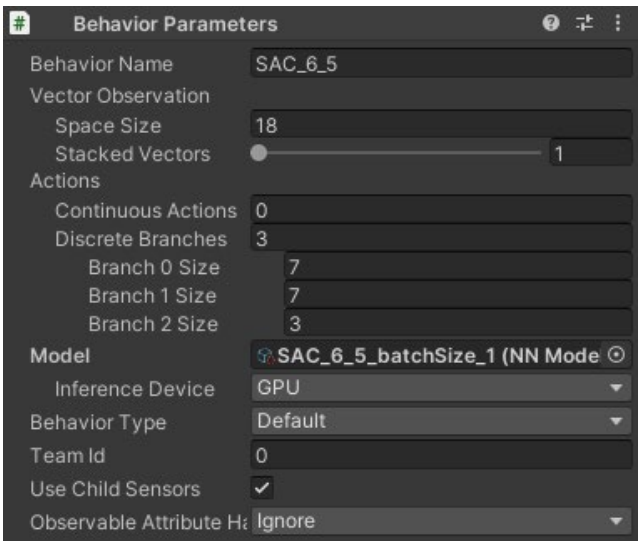
Daarna kunt u het agent object toevoegen aan de scene

- Voeg de agent prefab toe aan de scene (laatste versie van de prefab is CarAgent6)

- Ga naar het Behavior Parameters component van de agent en geef een Behavior Name mee.

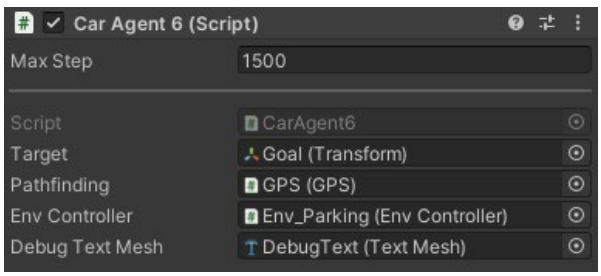
- Stel het aantal vector observaties in voor de agent. Deze observaties worden gedefinieerd in de CollectObservations methode. Enkel vector en scalar waarden kunnen meegegeven worden als observaties in deze methode. Raycast en Camera sensor observaties van componenten (zowel van de parent als child objecten) worden automatisch toegevoegd buiten het script.

- Stel de Behavior Type in op Default. Deze parameter beheerst of de agent staat ingeschakeld op manuele controle (Heuristic) of op het trainen/draaien van het model (Inference). Bij Default heeft u manuele controle zolang dat er geen training gebeurt en er geen getraind model is toegekend aan de agent.



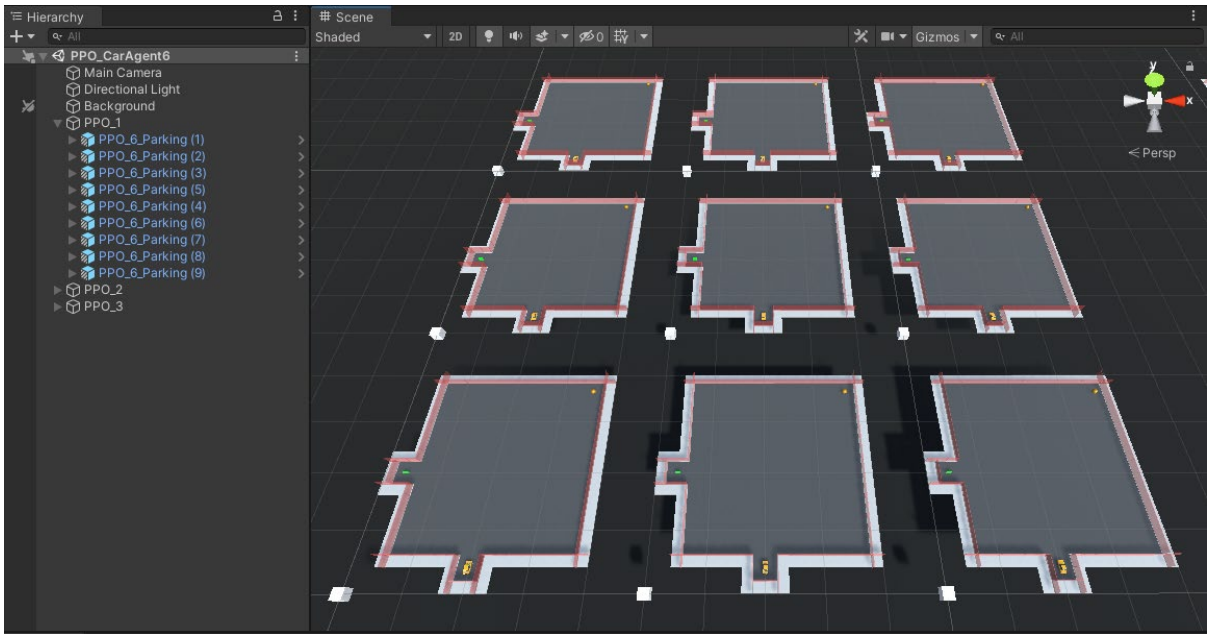
- Ga naar het CarAgent component en het maximaal aantal stappen in per episode voor de agent (Max Step)

- Geef de volgende objecten mee aan het component
    - Goal (het eindbestemming object)
    - GPS (voor de pathfinding)
    - EnvController



Nadat de environment en agent klaar zijn, kunt u deze toevoegen aan een lege gameobject en het object terug opslaan als een nieuwe prefab

Op deze manier kan de volledige environment makkelijk gebruikt worden over meerdere scenes en kan u meerdere instanties van de environment in dezelfde scene hebben (wat de duratie van de trainingssessie drastisch kan verlagen).



3 – Configuraties

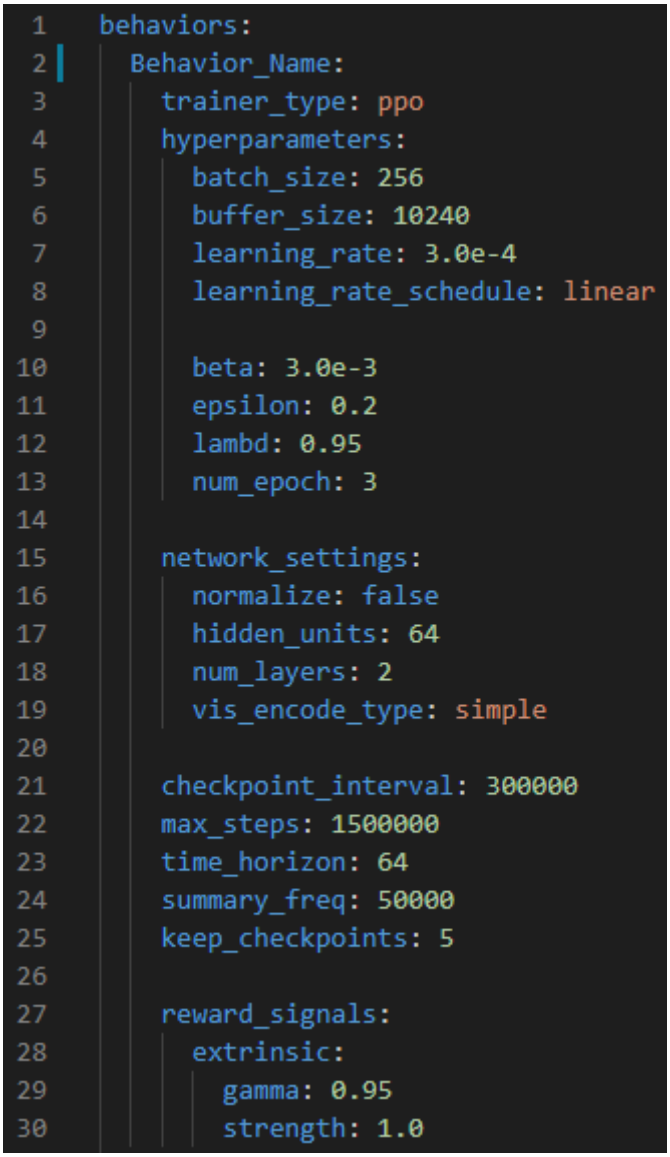
Voor de opmaak van de Reinforcement Learning algoritmes binnen de Unity ML-Agents Toolkit wordt gebruik gemaakt van configuratie files in YAML formaat. Deze bevatten hyperparameters en andere configuratie instellingen die bepalen welk algoritme toe te passen en hebben invloed op de training van het model.

De diverse configuraties zijn terug te vinden in de *config* directory.

3.1 – Configuraties – Parameters

Hier vindt u een lijst van een aantal belangrijke parameters. Indien een parameter niet is ingevuld, zal de training de standaard ingestelde waarde toepassen.

behavior_name	Naam van de gebruikte configuratie en het getrainde model
trainer_type	Type van trainer (ppo, sac of poca)
hyperparameters	Hyperparameters voor het model
network_settings	Instelling voor het neurale netwerk van het model
max_steps	Maximaal aantal training steps dat de agents mogen nemen voor de volledige training
summary_freq	Aantal stappen tussen de statistische samenvattingen die worden weergegeven op de command prompt tijdens de training
time_horizon	Aantal experiences (combinatie van observatie, actie en reward per step) dat moet verzamelt worden per agent voordat ze aan de buffer worden toegevoegd
checkpoint_interval	Aantal steps tussen elke checkpoint om een huidige versie van het model op te slaan
keep_checkpoints	Maximaal aantal checkpoint modellen om bij te houden op het einde van de training
reward_signals	Parameters voor extrinsic en intrinsic reward signals/beloning signalen



```yaml
behaviors:
  Behavior_Name:
    trainer_type: ppo
    hyperparameters:
      batch_size: 256
      buffer_size: 10240
      learning_rate: 3.0e-4
      learning_rate_schedule: linear

      beta: 3.0e-3
      epsilon: 0.2
      lambd: 0.95
      num_epoch: 3

    network_settings:
      normalize: false
      hidden_units: 64
      num_layers: 2
      vis_encode_type: simple

    checkpoint_interval: 300000
    max_steps: 1500000
    time_horizon: 64
    summary_freq: 50000
    keep_checkpoints: 5

    reward_signals:
      extrinsic:
        gamma: 0.95
        strength: 1.0
```

Indien u meer details wil over de configuratie kunt u deze terugvinden in de documentatie op de Toolkit's officiële GitHub pagina.

Link: https://github.com/Unity-Technologies/ml-agents/blob/main/docs/Training-Configuration-File.md

Het is belangrijk is dat de naam van de behavior in uw configuratie overeenkomt met de Behavior name parameter in het Behavior Parameters component van minstens één agent in de Unity scene. Deze opstelling geeft aan welke configuratie de agent zal gebruiken tijdens de training en deze waardes zijn hoofdlettergevoelig. Indien een configuratie file geen configuratie bevat voor een agent in de scene, krijgt u een foutmelding.

Het is mogelijk om meerdere behaviors te definiëren in één configuratie file en ze te gebruiken in éénzelfde scene door meerdere environment objecten in de scene te zetten. Op deze manier kunnen meerdere modellen tegelijkertijd trainen.

Hou wel rekening dat de training zal blijven draaien tot alle behaviors/modellen zijn getraind. Indien er geen agents in de scene zijn die gebruik maken van één behavior configuratie in de file, zal de training oneindig blijven doordraaien.

4 – Training

Als de trainingsomgeving is opgezet in een scene en we een configuratie file hebben, kunnen we beginnen met het trainen van modellen. Er zijn 2 manieren om de ML-Agents toolkit te gebruiken voor training:

- Binnen de Editor
- Via een build

4.1 – Training – Editor

Training via de Editor is de simpelste methode als u bezig bent met de ontwikkeling van bepaalde scenes en agents en de Unity Editor hebt openstaan.

Om een nieuwe trainingssessie te starten voert u het onderstaande commando uit in een command prompt (via de virtuele omgeving) vanuit de root directory van het project.

mlagents-learn <path_to_config> --run-id=<run_id>

Bij dit commando geeft u aantal parameters mee:

- *path_to_config* pad naar de configuratie file vanuit de root
- *run_id* naam van de training sessie/run

Hou rekening dat elke run een unieke id heeft. Het is niet mogelijk om meerdere sessies te draaien met dezelfde id. Zorg dus dat u voor elke nieuwe run een andere naam gebruikt.

Daarnaast zijn er extra parameters om het commando uit te breiden

- *-- resume* voortzetten van een sessie indien deze werd onderbroken
- *-- force* de bestaande sessie overschrijven met een nieuwe sessie

Indien alles correct werkt krijgt u de onderstaande scherm te zien.

Nu draaien de Python scripts van de training en moet enkel nog de scene in de editor starten om met de training te beginnen. U start de scene door op de Play knop te drukken die zich bevindt boven de viewport op de scene.

Terwijl dat de scene draait kunt u de agents te werk zien gaan in de scene. Op regelmatige intervallen krijgt u ook statistische samenvattingen te zien in de command prompt en zal een checkpoint van elk model worden opgeslagen (afhankelijk van de configuratie parameters).

```
[INFO] SAC_6_5_2. Step: 10000. Time Elapsed: 117.623 s. Mean Reward: 40.932. Std of Reward: 10.247. Training.
[INFO] SAC_6_5_3. Step: 10000. Time Elapsed: 119.629 s. Mean Reward: 56.421. Std of Reward: 22.308. Training.
[INFO] SAC_6_5. Step: 10000. Time Elapsed: 121.620 s. Mean Reward: 77.593. Std of Reward: 52.710. Training.
[INFO] SAC_6_5. Step: 20000. Time Elapsed: 247.322 s. Mean Reward: 344.210. Std of Reward: 250.043. Training.
[INFO] SAC_6_5_2. Step: 20000. Time Elapsed: 247.850 s. Mean Reward: 132.943. Std of Reward: 48.401. Training.
[INFO] SAC_6_5_3. Step: 20000. Time Elapsed: 249.255 s. Mean Reward: 215.643. Std of Reward: 63.429. Training.
[INFO] SAC_6_5. Step: 30000. Time Elapsed: 378.366 s. Mean Reward: 312.943. Std of Reward: 289.078. Training.
[INFO] SAC_6_5_2. Step: 30000. Time Elapsed: 379.170 s. Mean Reward: 359.793. Std of Reward: 115.984. Training.
[INFO] SAC_6_5_3. Step: 30000. Time Elapsed: 381.465 s. Mean Reward: 308.333. Std of Reward: 169.677. Training.
[INFO] SAC_6_5_2. Step: 40000. Time Elapsed: 508.026 s. Mean Reward: 709.924. Std of Reward: 48.782. Training.
[INFO] SAC_6_5_3. Step: 40000. Time Elapsed: 509.943 s. Mean Reward: 294.716. Std of Reward: 241.561. Training.
[INFO] SAC_6_5. Step: 40000. Time Elapsed: 512.922 s. Mean Reward: 395.807. Std of Reward: 362.047. Training.
[INFO] SAC_6_5_2. Step: 50000. Time Elapsed: 634.062 s. Mean Reward: 735.931. Std of Reward: 126.219. Training.
[INFO] Exported results\SAC_04_entcoef2\SAC_6_5_2\SAC_6_5_2-49980.onnx
[INFO] SAC_6_5_3. Step: 50000. Time Elapsed: 635.486 s. Mean Reward: 410.267. Std of Reward: 303.659. Training.
[INFO] Exported results\SAC_04_entcoef2\SAC_6_5_3\SAC_6_5_3-49969.onnx
[INFO] SAC_6_5. Step: 50000. Time Elapsed: 638.226 s. Mean Reward: 371.184. Std of Reward: 363.733. Training.
[INFO] Exported results\SAC_04_entcoef2\SAC_6_5\SAC_6_5-49989.onnx
[INFO] SAC_6_5. Step: 60000. Time Elapsed: 757.755 s. Mean Reward: 248.663. Std of Reward: 486.328. Training.
```

Op het einde van de training zullen de finale getrainde modellen worden opgeslagen in ONNX formaat.

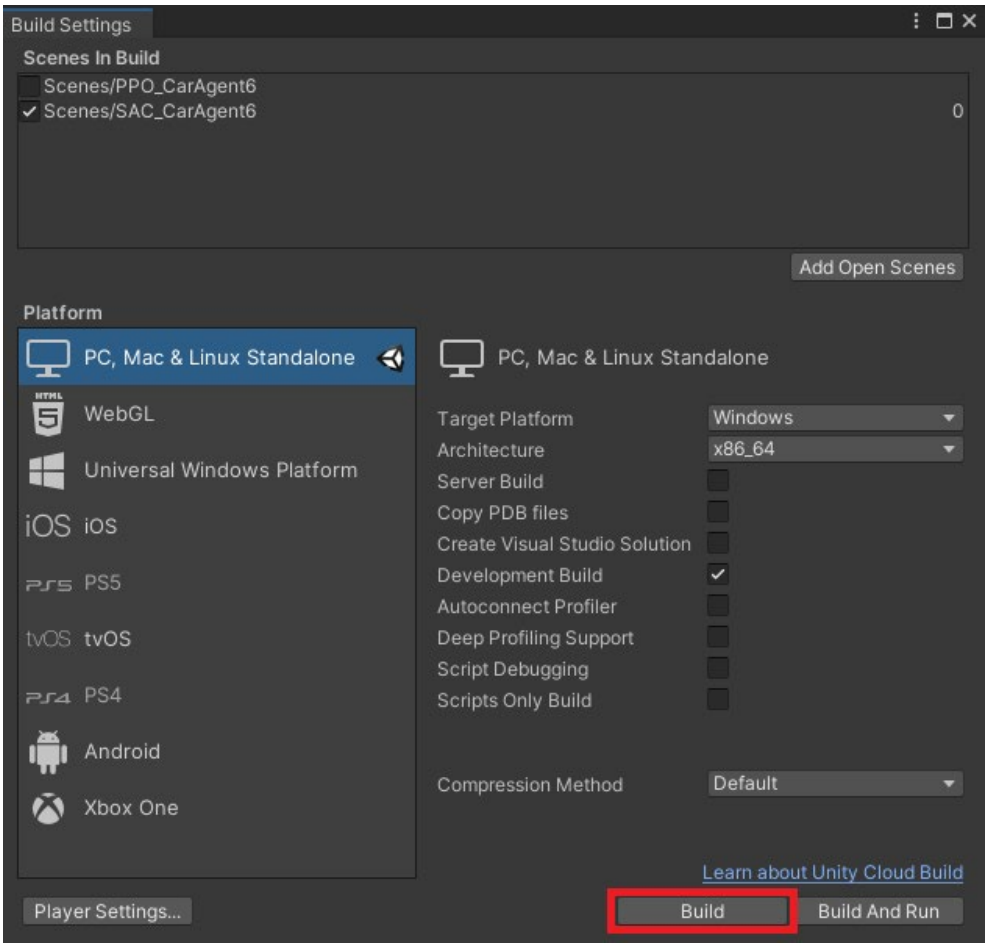
```
[INFO] Exported results\SAC_04_entcoef2\SAC_6_5\SAC_6_5-499986.onnx
[INFO] Exported results\SAC_04_entcoef2\SAC_6_5_2\SAC_6_5_2-500064.onnx
[INFO] Copied results\SAC_04_entcoef2\SAC_6_5_2\SAC_6_5_2-500064.onnx to results\SAC_04_entcoef2\SAC_6_5_2.onnx.
[INFO] Exported results\SAC_04_entcoef2\SAC_6_5_3\SAC_6_5_3-500142.onnx
[INFO] Copied results\SAC_04_entcoef2\SAC_6_5_3\SAC_6_5_3-500142.onnx to results\SAC_04_entcoef2\SAC_6_5_3.onnx.
[INFO] Exported results\SAC_04_entcoef2\SAC_6_5\SAC_6_5-500050.onnx
[INFO] Copied results\SAC_04_entcoef2\SAC_6_5\SAC_6_5-500050.onnx to results\SAC_04_entcoef2\SAC_6_5.onnx.

(mlagent_release_18) C:\Users\alexc\Documents\Unity\Projects\ResearchProject>
```

4.2 – Training - Build

Trainen via een build verschilt niet veel van de editor methode. Hierbij kan u in plaats van de editor gebruik maken van een vooraf gecompileerde build van de scene.

- Ga naar File > Build Settings
- Voeg de geopende scene toe en zorg dat enkel de te gebruiken scene is aangevinkt
- Selecteer PC, Mac & Linux Standalone als platform
- Vink Development Build aan
- Ga naar Player Settings > Player > Resolution en vink Run in Background aan
- Klik Build en selecteer de folder waar u build zal terechtkomen (aangeraden om alle builds voor de scenes onder te brengen in de *builds* directory)



Na de compilatie, kan het pad naar de build folder van deze scene worden meegegeven met het commando om de training te starten. De build zal automatisch opstarten voor de trainingssessie en sluit zichzelf na de training.

mlagents-learn <path_to_config> --run-id=<run_id> --env=<path_to_build>

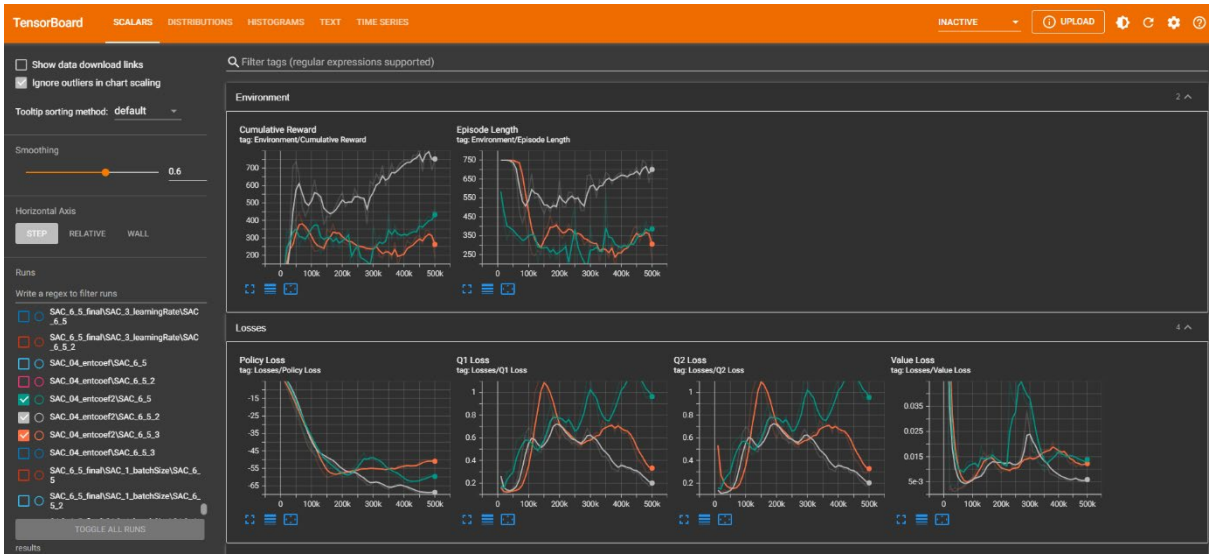
Het is niet mogelijk om meerdere training commando's tegelijkertijd uit te voeren doordat de communicatie tussen Unity en Pytorch verloopt over een lokaal netwerk over dezelfde poort (poort 5004).

5 – Resultaten

Nadat de training is voltooid, bevinden de resultaten van de sessie (finale modellen, checkpoints en statistieken) zich in de *results* folder van de project root directory. We kunnen het verloop van de sessies inspecteren via Tensorboard en de getrainde modellen importeren in Unity om ze te zien draaien in de scene.

5.1 – Resultaten – TensorBoard

Via TensorBoard kunt u de resultaten en het verloop van de training bekijken. Hier kunt u meerdere statistieken terugvinden per model zoals de cumulatieve rewards, lengte van de episodes, policy loss, value loss, …



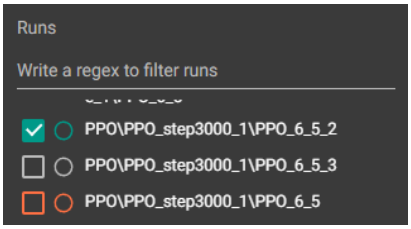
Om TensorBoard te draaien gebruikt u het volgende commando vanuit de command prompt in de project root directory.

tensorboard --logdir=results

Vanaf dat Tensorboard draait, kunt u in uw browser surfen naar http://localhost:6006 en wordt u gebracht naar de hoofdpagina van Tensorboard.

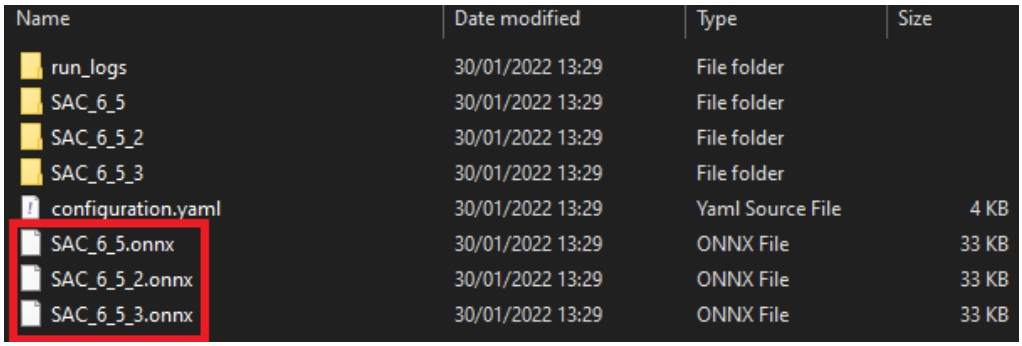
```
(mlagent_release_18) C:\Users\alexc\Documents\Unity\Projects\ResearchProject>tensorboard --logdir=results
TensorFlow installation not found - running with reduced feature set.
Serving TensorBoard on localhost; to expose to the network, use a proxy or pass --bind_all
TensorBoard 2.7.0 at http://localhost:6006/ (Press CTRL+C to quit)
```

In de zijbalk kan u dan selecteren welke training runs u wilt bekijken.



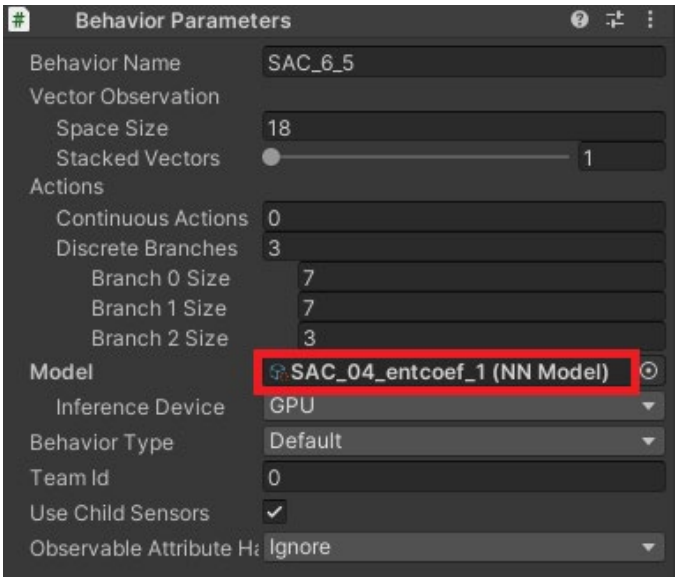
5.2 – Resultaten – Unity

In de resultaat folder van uw sessie (met de opgegeven run id) vindt u de getrainde modellen voor de geconfigureerde behaviors in ONNX formaat.



Door deze model files toe te voegen aan de Assets/Models directory binnen het Unity project kunt u deze gebruiken binnen de editor. Het model moet nu enkel nog toegekend worden aan een agent in de scene. Meerdere agents kunnen ook tegelijkertijd hetzelfde model toepassen.

- Open een scene in Unity
- Ga naar de Behavior Parameters component van de agent gameobject
- Selecteer het getrainde model in de Model parameter (door de file in te slepen vanuit de assets of op het bolletje te klikken en te zoeken op naam)



Als een getraind model is toegekend aan een agent, zal de agent bij het starten van de scene in de editor automatisch reageren op observaties van de omgeving. U kunt ook de scene opnieuw compileren naar build samen met een camera in de scene. Bij het opstarten van het build programma (via het .exe bestand) zal een nieuw venster openen waar u de agent zal zien bewegen vanuit het perspectief van de camera.