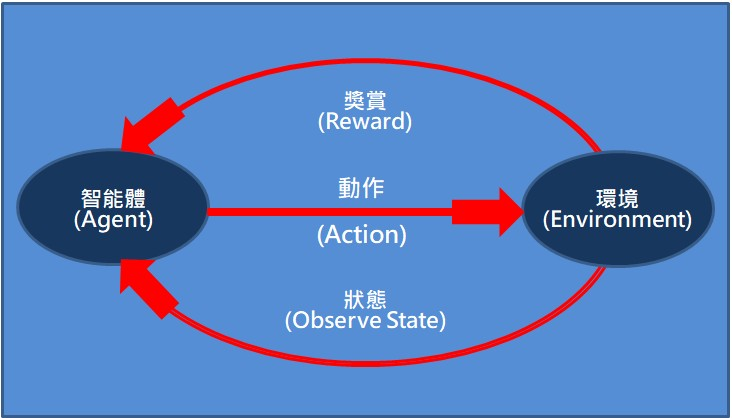
深度學習期末專題 1092B0024周翰文 1092B0030 李沐風

摘要(專題簡介):

我們的期末專題是利用UnityEngine內建的深度學習函式庫-ML Agent來達成自走車以及自動停車。其中，我們是用conda prompt來進行訓練的，在conda創造一個虛擬環境後，再往裡面安裝python、pytroch，與ml-agents套件。

我們在這次專案使用的是強化學習。如圖，我們的Agent會隨機的控制我們的車輛，也就是Action。然後環境就會回傳在這一連串的動作裡面得到了多少獎賞與現在的狀態。然後我們使用的算法是ppo(近便政策最佳化)，他是一種策略梯度方法，也就是說他會直接對策略進行建模並優化。也因此我們沒有一個資料庫去讓他學習，取而代之的是我們訓練之前就設定好的獎勵。ppo有著數據效率低、策略更新穩定的優點，並且常用於強化學習，這也是我們選擇這個算法的主因。

(使用工具):

軟體:

1.UnityEngine 2021.3.12

2.Python 3.9

3.Anaconda Prompt

4.Pytorch 11.7

硬體:

RTX 3060 (筆電版)

(前置環境與參數程式):

**參數:**

**Car Agent**:

Max Steps: 每次訓練最大量的步進(修整學習速度、學習參數)

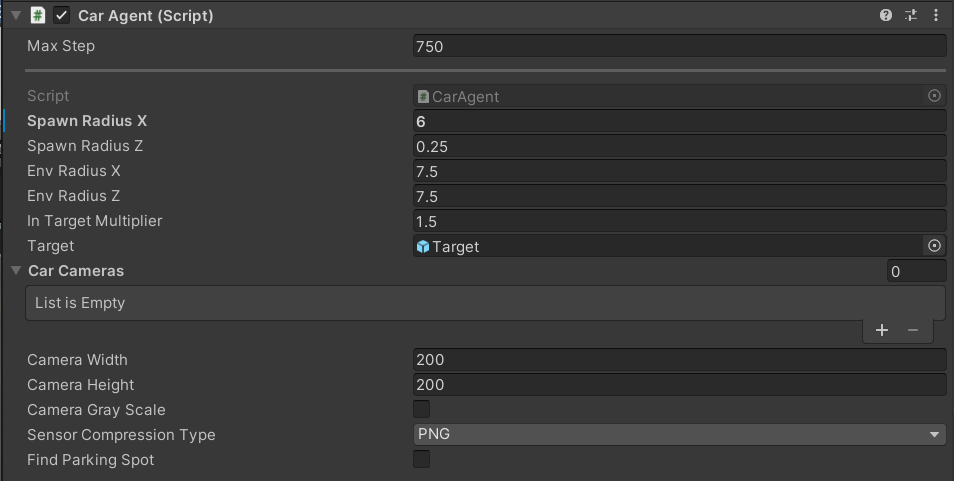
Spawn Rad.x/z: 生成位置的 X/Z

Env Rad. X/Z: 超出指定範圍後初始化位置

Target: prefab , 終點

Cam Gray Scale: 灰階影像

Sensor Compression: 存檔模式



**Behavior Parameters**:

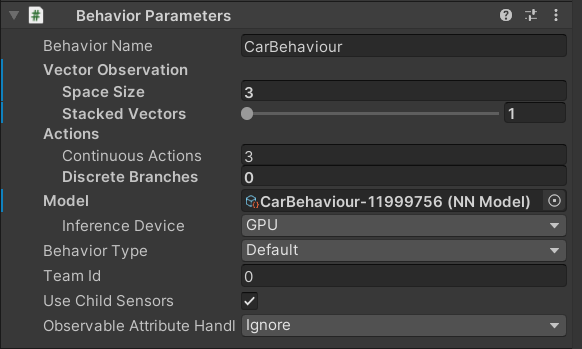
Vector Observation: 如何儲存資料到我的訓練模型(控制傳感器偵測與回傳)

Stacked Vectors: 幾個射線判斷一邊的移動(利用距離)

Model: 存放訓練模型的地方(新訓練就是保持空物件)

Actions:

Continuous: 連續動作(混和動作)



**Descision Requester**:

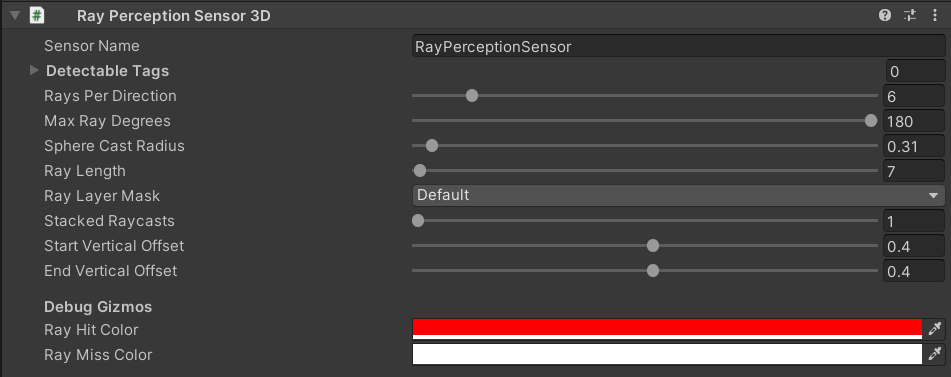
處理Agents的決策過程，DecisionPeriod 為1代表Agents將每1個步驟請求一個決定。

一張含有 文字, 螢幕擷取畫面, 裝置 的圖片

自動產生的描述

**Ray Perception Sensor**:

ml agents套件內建的車子傳感器



**Demonstration Recorder**:

範例程式，開啟後會訓練，並產生一個策略文件。後續會再提及。

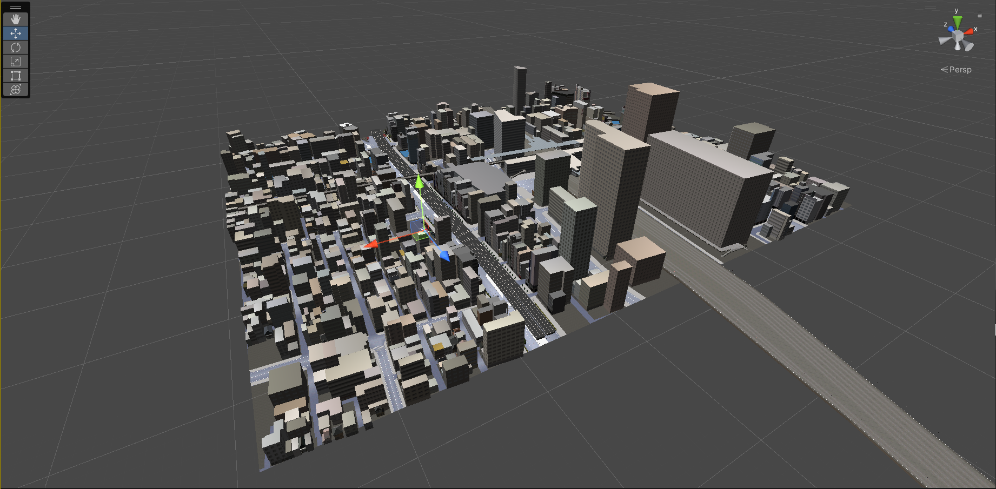


環境架設:

下圖是我們的訓練環境。在停車時會有一輛車從紅色區域行駛至左側。

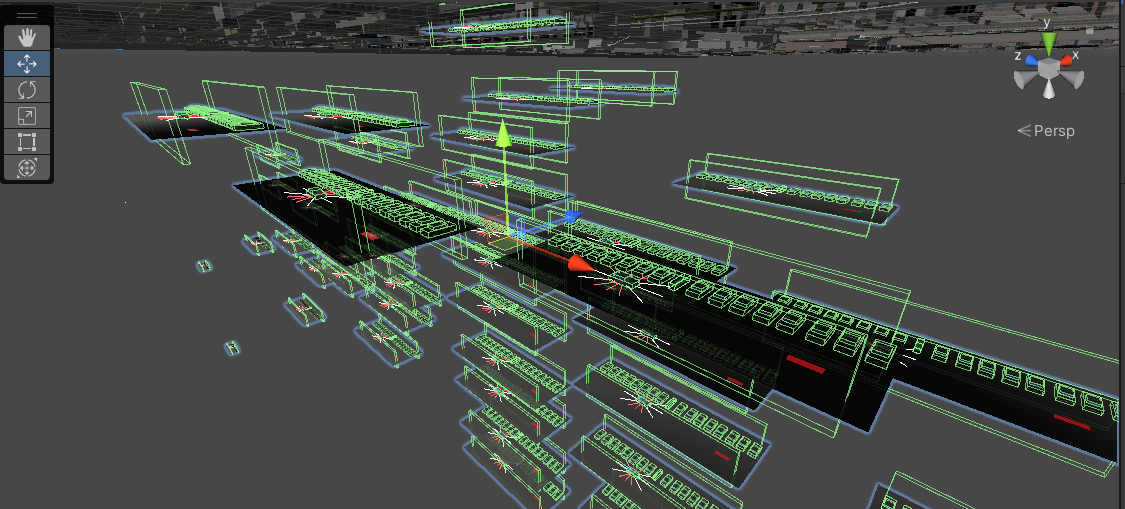
**模型:**

運用別人做的城市模型進行擬真訓練。<https://assetstore.unity.com/packages/3d/environments/urban/japanese-otaku-city-20359#reviews>

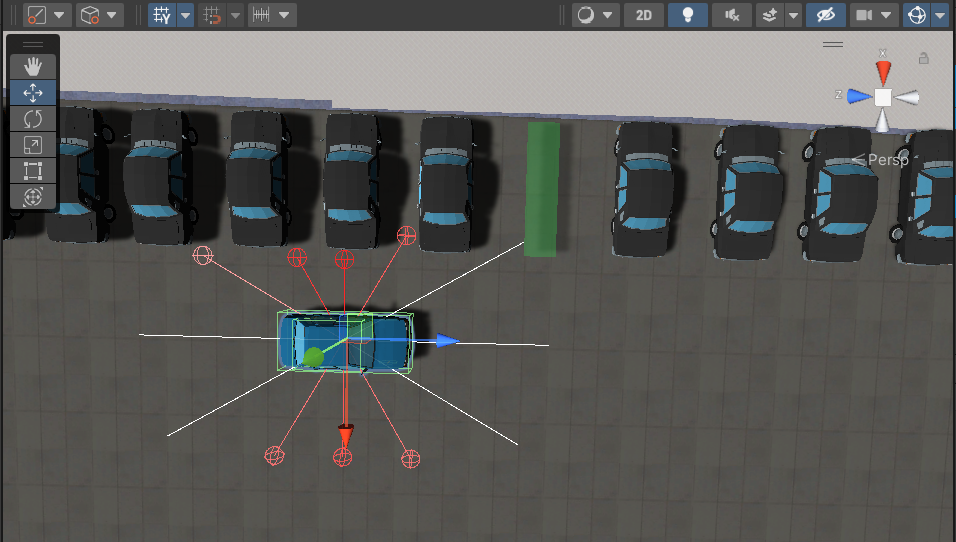


**訓練陣列:**

我們可以在訓練的時候增多我們的訓練場景，加快我們訓練速度(但過量的訓練可能會因為硬體吃不消以致我們訓練速度不增反降)。



**感測器:**

將許多攝影機放置在要學習的車輛四周，來做為車輛控制與距離控制或是碰撞偵測。

(獎勵機制):

增加獎勵:成功停車

成功停車時，也就是車子停在指定的位置(Target)會增加100點獎勵。但是，我們會給他乘上速度與角度的倍率。讓他停得更快，更準確。

扣除獎勵: 撞牆、撞車以及超出訓練範圍。

撞牆與超出訓練範圍均是減10點獎勵

撞車則是扣除(-car\_speed\*50-5)點獎勵。假如車子的速度越快並撞到車子，便會扣除越高的獎勵

一張含有 文字 的圖片

自動產生的描述(訓練文件，與過程):

策略文件:

用於生成對抗模仿學習與與監督是學習的必要文件(訓練耗時30分鐘)，需開啟Demonstration Recorder。目的是給真正要訓練的模型幾個成功廷吃的例子，讓它訓練得更好。

<https://www.youtube.com/watch?v=upyeNufLjs8>

一張含有 文字, 電子用品 的圖片

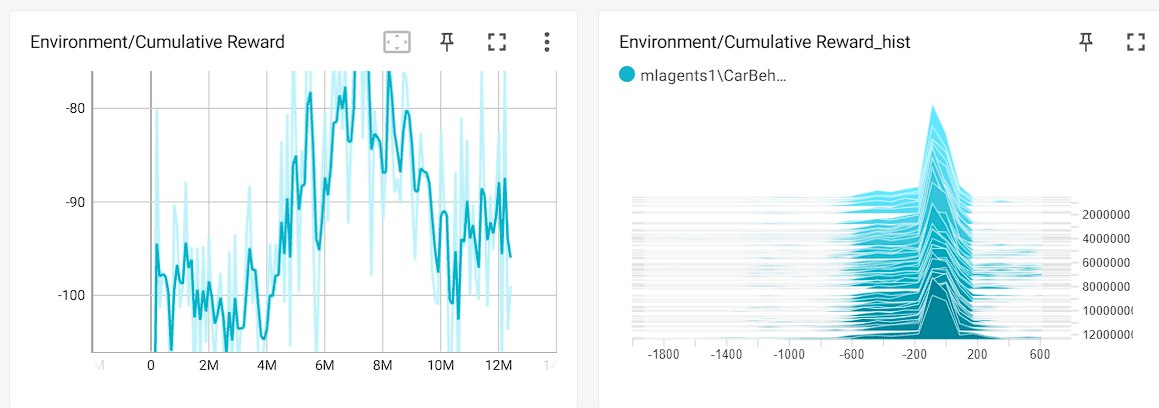
自動產生的描述訓練過程:

(於conda prompt裡的虛擬環境訓練)

由訓練成果的圖中，可以看出，每走100000步後，就會記錄這100000部的數據，並且每走1000000步，會儲存一個節點，並儲存模型。Time Elapsed則是從目前為止所耗費的時間。Mean reward 則是在這100000步得到的獎勵。Std reward則是衡量平均獎勵分佈的量度。較大的值表示收到的獎勵變化很大，而較小的值則相反。

Tensorboard:

左為生成對抗網路的Loss，右為決策過程的Loss

下圖皆是Mean reward 的分布。幾乎都是落在-80的區間

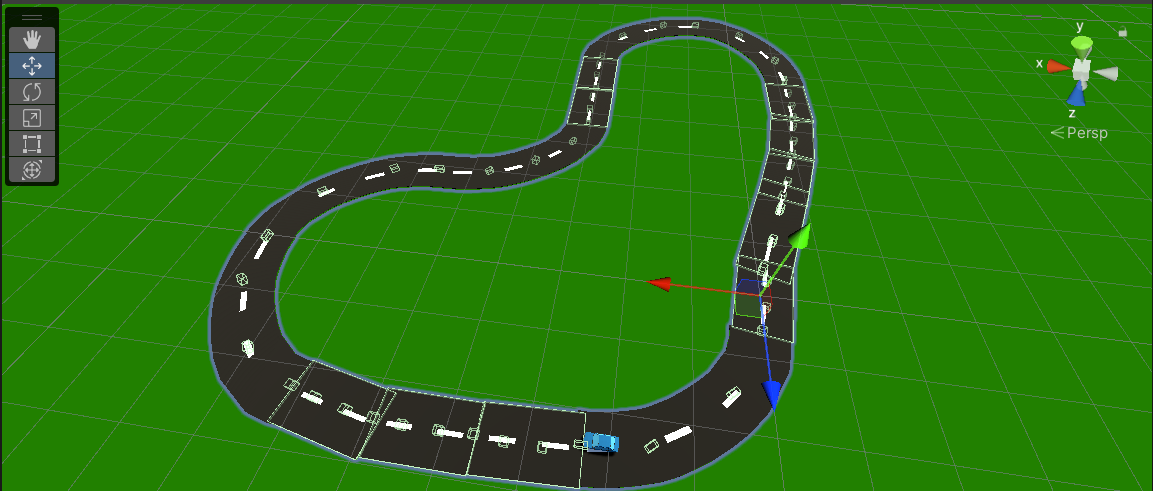
執行結果:

由於我的電腦不能開機過長時間，會自動關機。因此原本預計的五千萬步變得只有一千兩百萬多步。但依舊可以看出訓練出來的成效。到停車位時會停止，並且會試圖想要插進去車位裡，並且不會到處亂衝亂撞。

<https://www.youtube.com/watch?v=KmknP3KVYIs>

額外討論

我們有用原本我們裝在停車的程式去改成自走車。但只有半成品，並且出了一些Bug，所以只能講它的原理。我們使用的方式是在笛版上放置許多獎勵，並且在他超出道路時就會扣除獎勵，所以是有點類似貪食蛇的概念，藉由在路上放獎勵，來讓車子知道要往哪裡走。



貢獻:

1. 更改環境，更改與優化參數

2. Tensorboard視覺化分析

3. 優化car spawner

4. 利用原本程式去改成自走車半成品(無法正常轉彎)

檔案連結:

<https://www.mediafire.com/folder/4wv0m4mf7kkul/Assets>

<https://www.mediafire.com/file/ooqg9akjrutm47w/90gays.demo/file>

<https://www.mediafire.com/file/g5yz55qwjy97e9j/trainer_settings.yaml/file>

參考資料:

<https://www.youtube.com/watch?v=_Bzw2B-9QkM&t=1560>

<https://cloud.tencent.com/developer/news/841692>

<https://technews.tw/2018/05/23/unity-ai-ml-agents/>

<https://iter01.com/444175.html>

<http://13.231.129.69/2019/12/19/unity3d-ml-agent-1-%E8%A7%80%E5%BF%B5%E7%AF%87/>

<https://blog.csdn.net/c20081052/article/details/88839479>

<https://blog.csdn.net/qq_30615903/article/details/86308045>

<https://blog.csdn.net/kuvinxu/article/details/109606369>

<https://cloud.tencent.com/developer/article/1760006>

<https://docs-unity3d-com.translate.goog/Packages/com.unity.ml-agents@1.0/api/Unity.MLAgents.DecisionRequester.html?_x_tr_sl=en&_x_tr_tl=zh-TW&_x_tr_hl=zh-TW&_x_tr_pto=sc>