註:我符合加分資格，有自行編寫模擬程式，並未使用範例程式

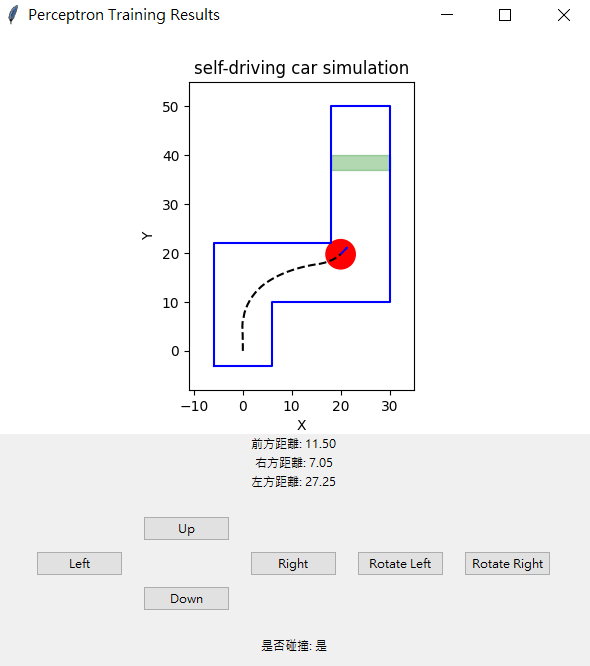
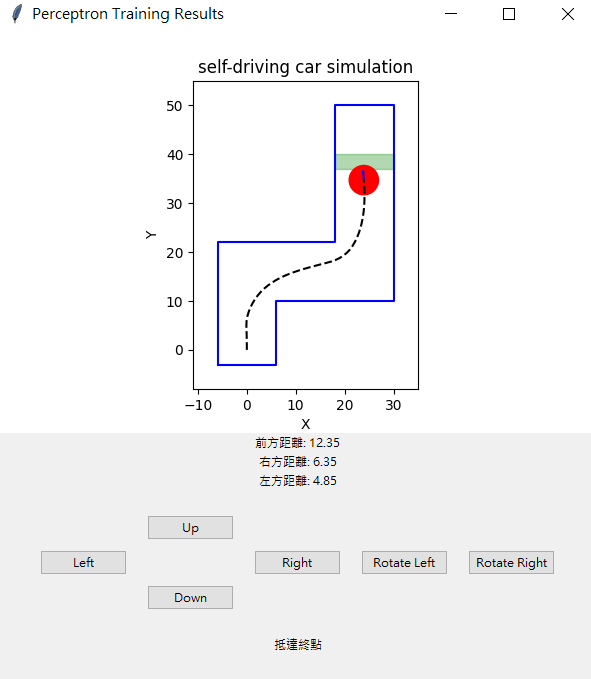
1. 介面介紹:

左為成功到終點，右為失敗撞牆

打開exe檔會先後台進行訓練約5-10秒，訓練完畢會直接打開GUI開始跑程式

0.5秒更新一次畫面

因為模擬程式是自己寫的，故我加了方向鍵跟控制旋轉的(可以介入當前再播的動畫)



1. 程式碼說明

Math\_tool.py:

數學運算用class，裡面全是static function

MLPnetwork.py:

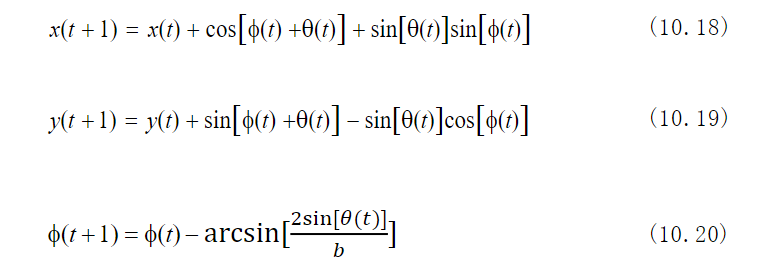
MLP模型，初始化參數可以調整4D/6D

裡面包含load\_data、train、和依據輸入產生下一個方向盤的state

這邊我是先將輸入正規化後，把正規化後的輸出(-1 ~ 1)直接\*40(-40 ~ 40)

Self\_driving\_car.py:

- `load\_data(self)`: 讀取軌道座標和終點區域的資料。

- `update\_state(self)`: 根據下圖產生下一個state

- `calculate\_distances(self)`: 計算車體前方、左方、右方的距離。

- `calculate\_distance\_in\_direction(self, angle)`: 根據給定的角度計算車體在該方向上的距離。用有點土法煉鋼的方法，所以距離超過100就會變inf且效能不大好

以0.05為一基本單位來計算

- `check\_collision(self)`: 檢查車體是否與任何牆壁發生碰撞。

- `reach\_goal(self)`: 檢查車體是否抵達目標終點。但其實就是檢查車體與終點的rectangle發生碰撞與否

- `check\_car\_collision(self, target)`: 檢查車體是否與指定目標發生碰撞。檢查方法為判斷線段(拉兩個連續的頂點)與圓(車體)有沒有相交

Gui.py(4D/6D沒差很多，但我拆兩個方便debug):

- `\_\_init\_\_(self)`: 初始化 GUI 窗口，包括模擬軌跡、車輛、感測器等元素。

畫圖相關函式

- `draw\_map(self)`: 繪製地圖，包含軌道和終點區域。

- `draw\_car(self)`: 繪製車輛，包括車子外圓和朝向箭頭。

- `draw\_fig(self)`: 繪製整個模擬圖形，包括地圖和車輛。

- `draw\_track\_trace(self)`: 繪製軌跡追蹤線，展示車輛運動軌跡。

- `fig\_setting(self)`: 設定圖形顯示屬性，如坐標軸範圍、標題等。

- `create\_simulation\_figure(self)`: 創建模擬圖形的主窗口。

模擬相關函式

- `start\_simulation(self)`: 啟動模擬運行的主循環。

- `run\_simulation(self)`: 模擬主循環，更新車輛狀態並實時更新 GUI。

畫面更新相關函式

- `update\_gui(self)`: 更新 GUI 元素，包括軌跡、車輛、距離感測器顯示。

- `update\_distances\_labels(self)`: 更新感測器距離的顯示文字。

碰撞顯示相關函式

- `create\_sensor(self)`: 創建感測器顯示區域，展示前、右、左方距離。

- `create\_collision\_label(self)`: 創建碰撞標籤，顯示車輛是否碰撞。

- `update\_collision\_label(self)`: 更新碰撞標籤的文字內容。

- `create\_buttons(self)`: 創建控制按鈕，包括上、下、左、右移動和旋轉按鈕。

方向鍵相關函式

- `move\_car(self, dx, dy)`: 移動車輛，根據給定的 dx 和 dy 值。

- `rotate\_car(self, angle)`: 旋轉車輛，根據給定的角度。

- `move\_up(self)`: 向上移動車輛的操作函數。

- `move\_down(self)`: 向下移動車輛的操作函數。

- `move\_left(self)`: 向左移動車輛的操作函數。

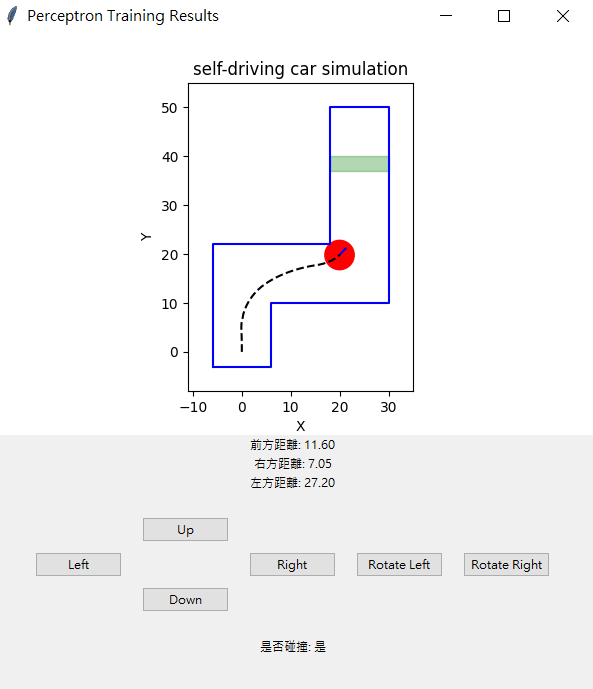
- `move\_right(self)`: 向右移動車輛的操作函數。

- `rotate\_left(self)`: 左轉車頭的操作函數。

- `rotate\_right(self)`: 右轉車頭的操作函數。

1. 實驗結果與分析

4D跟6D有顯著的成功率差異，6D成功率高很多，但兩者都不是百分百成功，

且發現失敗的案例通常會在下圖的位置撞到  
推測試資料樣本不足，軌道內的蠻多點都沒有case的，模型就只能沿著主要路徑走，但會因為訓練的部分隨機因素而撞牆

下圖為將train6D的每個點印在圖上的結果

