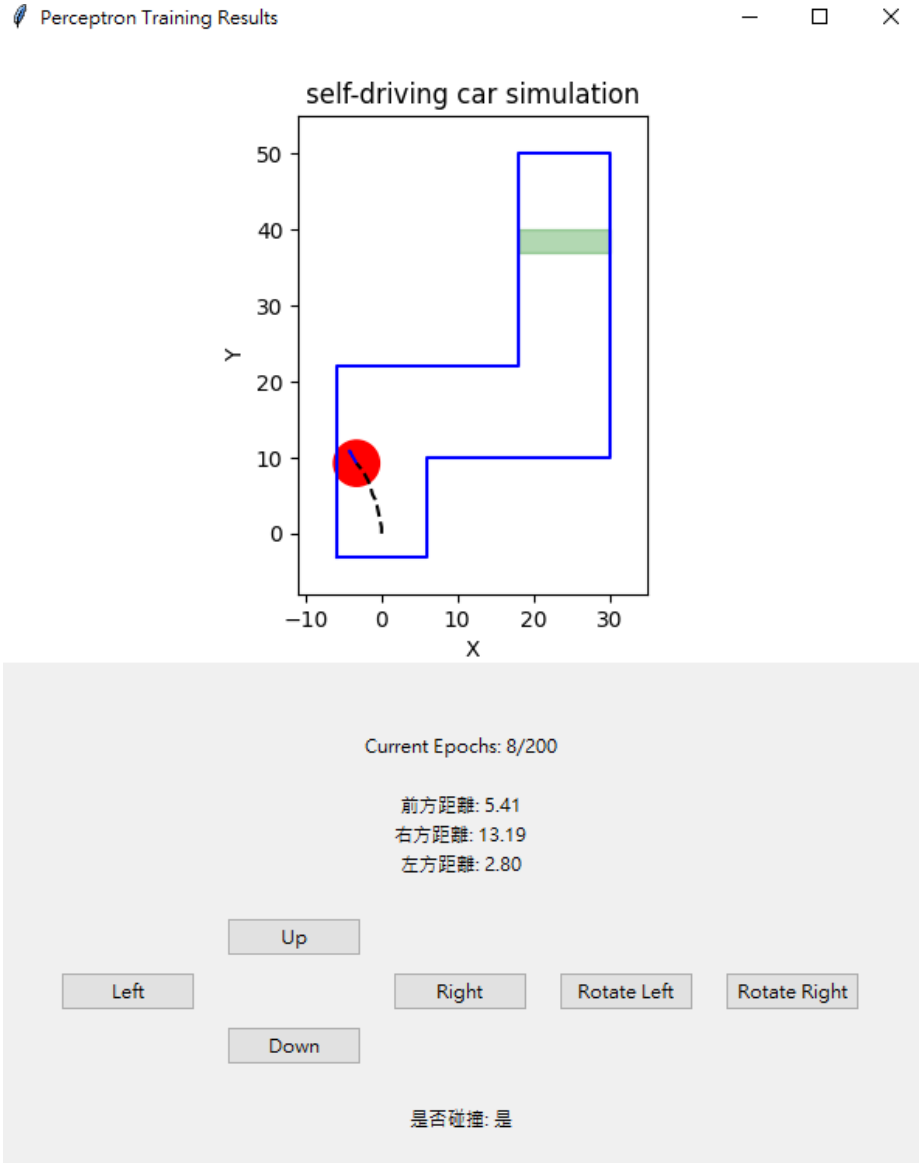
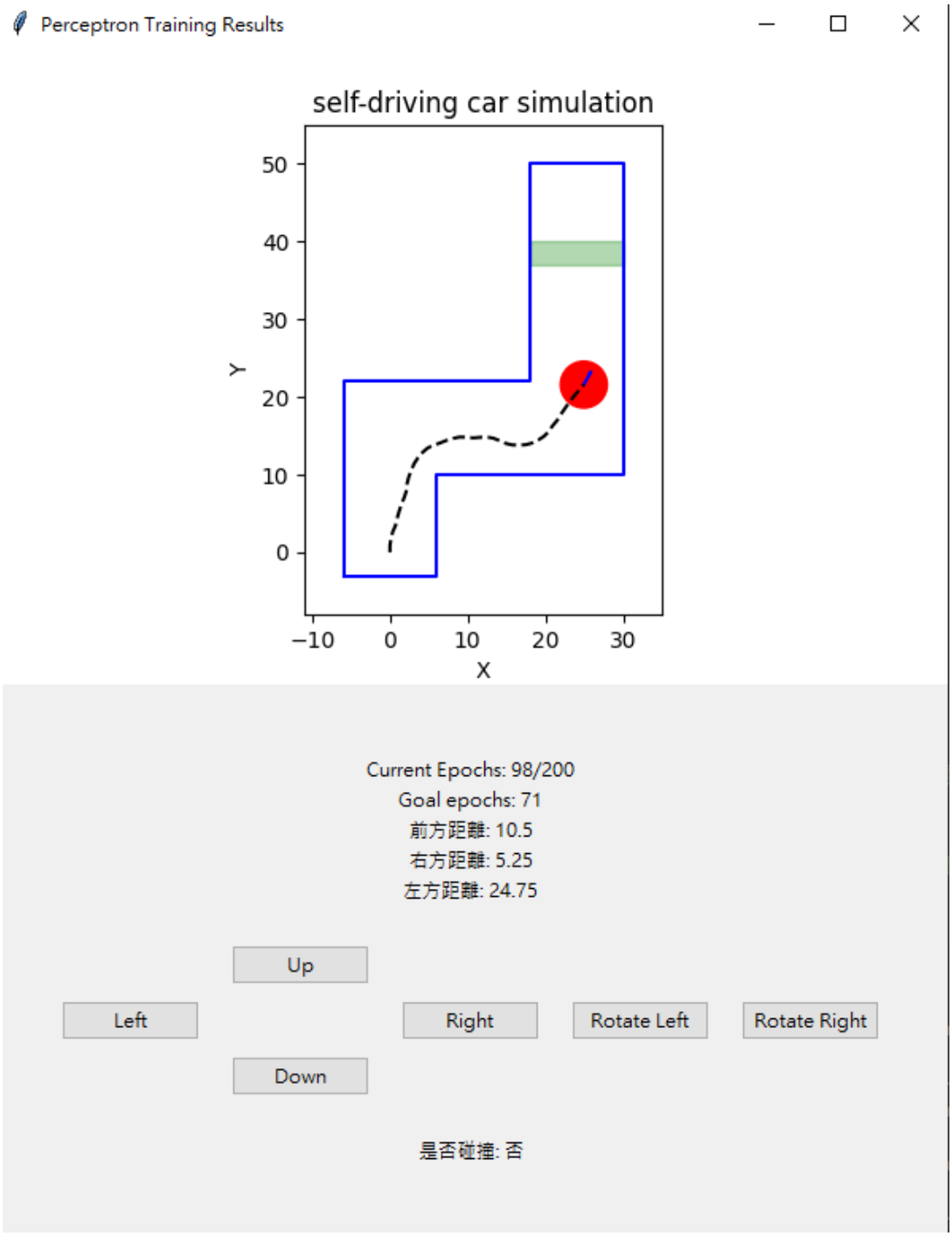
1. 程式介面說明

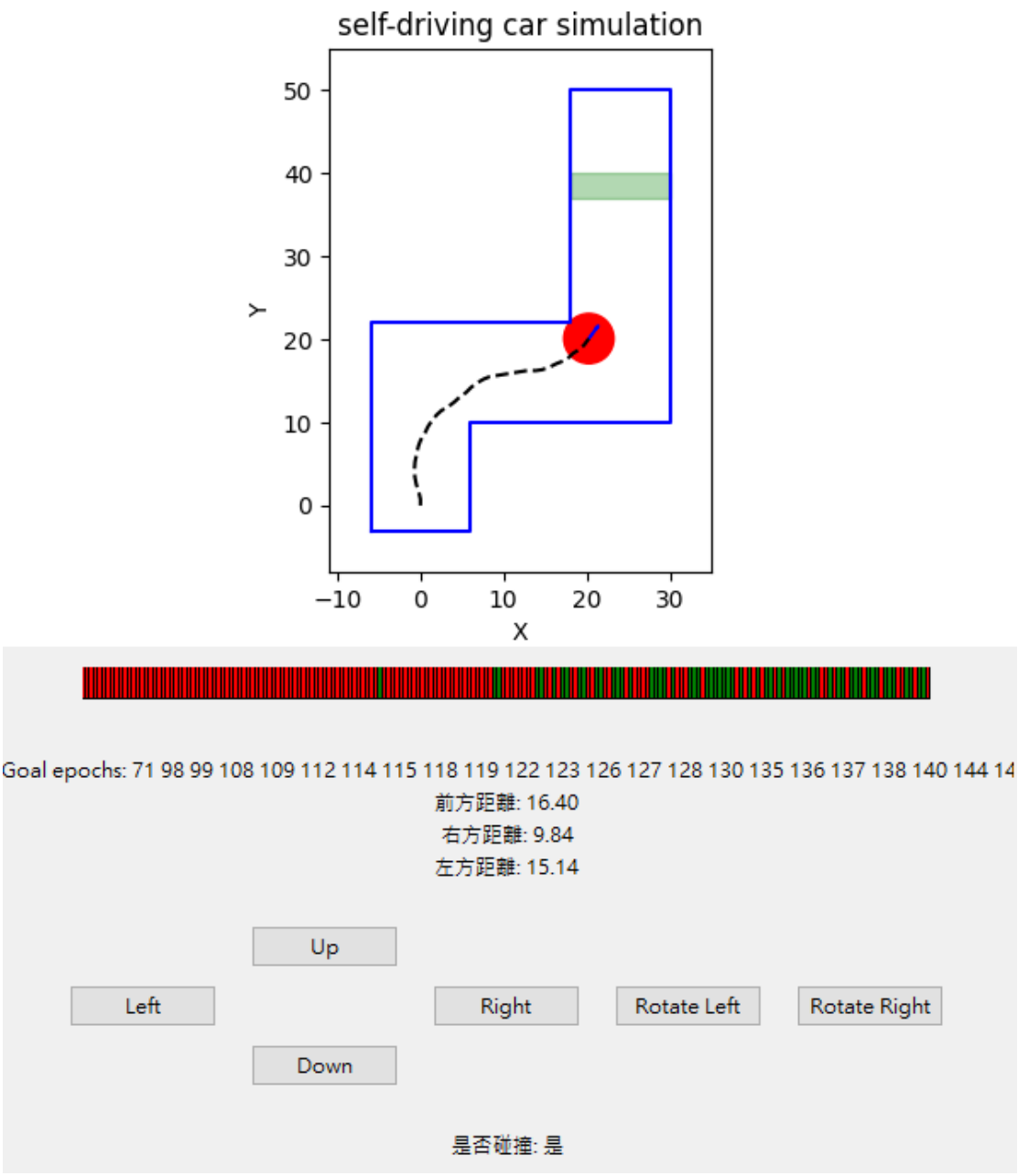
一開始於行程式會直接開始跑 training，然後會持續顯示每次 training 的軌跡在 GUI 上



當有任何的 epochs 抵達終點時，會更新 Goal epochs 到 GUI 上



全部訓練完成後(約 5 分鐘)會顯示訓練結果到 GUI 上，紅色為撞牆，綠色為到終點



下方按鈕為娛樂用，要按也可以按

2. 實驗結果

Q learning 參數

```
def __init__(self, degree_per_actions=4, state=5, learning_rate=0.1,
gamma=0.9, exploration_rate=1.0, exploration_decay=0.992):
```

State 定義

依據 L-R 的值做分配

```
def direction_to_state(self, inputs):
    inputs = np.round(inputs).astype(int)
    #state =      0          1        2         3          4
    #        abs(l-r)<3   3<l-r<10   l-r>10   3<r-l<10   r-l>10
    state=0
    dif = inputs[1]-inputs[2] #r-l
    if dif < 0:
        if -dif > 3 and -dif < 10:
            state=1
        elif -dif >= 10:
            state=2
    else:
        if dif > 3 and dif < 10:
            state=3
        elif dif >= 10:
            state=4
    return state
```

Action 定義

一格為 4 度，0=0 度

```
def convert_action_to_angle(self, action):
    #action 0  1-10  11-20
    # angle 0  -4~-40  4~40
    if action == 0:
        return 0
    elif action < 11:
        return -self.degree_per_action*(action)
    #11<=action<=20
    return self.degree_per_action*(action-10)
```

Q table 定義

Num_state*num_actions

```
self.q_table = np.zeros((self.num_state, self.num_actions))
```

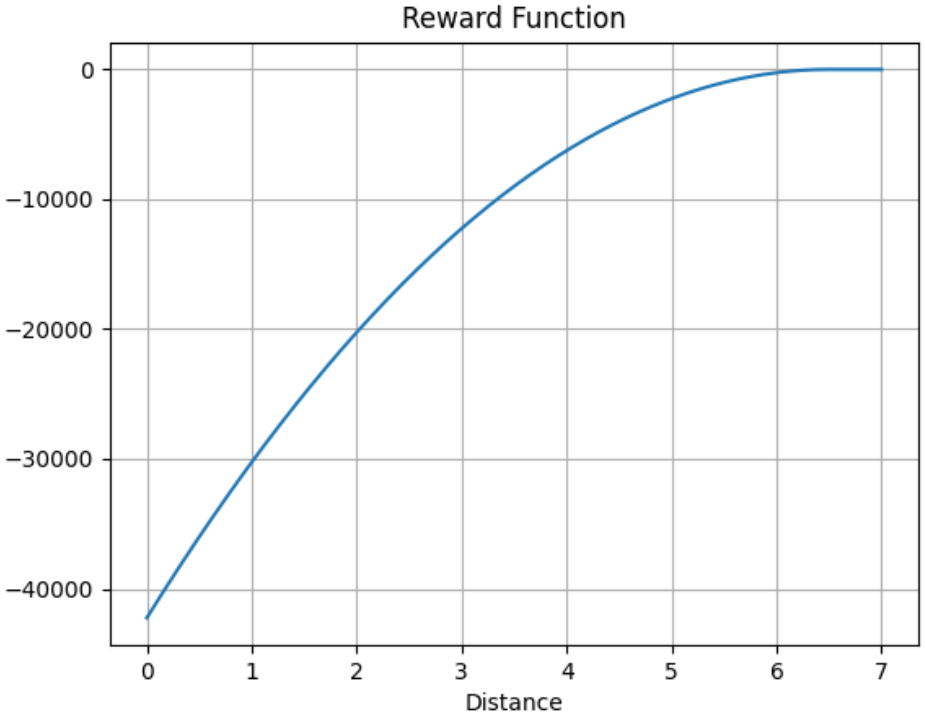
3. 歸屬函數說明

```
    def rew(self,distance):
        return max(6.5-distance,0)**2*1000
    def take_action(self, action):
        self.car.update_state(self.convert_action_to_angle(action))
        next_input=self.car.get_distances()
        next_state = self.direction_to_state(next_input)
        reward=-5
        if self.car.reach_goal():
            reward+=1000000
        elif self.car.check_collision():
            reward=-200
        p=(self.rew(next_input[0])+self.rew(next_input[1])+self.rew(n
ext_input[2]))
        reward-=p #太貼牆會大幅減少 reward
        q=10*(400 -
MathTool.point_to_polygon_distance(self.car.x,self.car.y,self.car.en
d_area)**2) #離終點越近 reward 越高
        reward+= q
        #print("reward:",reward,"wall:",p,"distance:",q)
        return next_state, reward
```

當抵達終點，給予一個極大的正面 reward

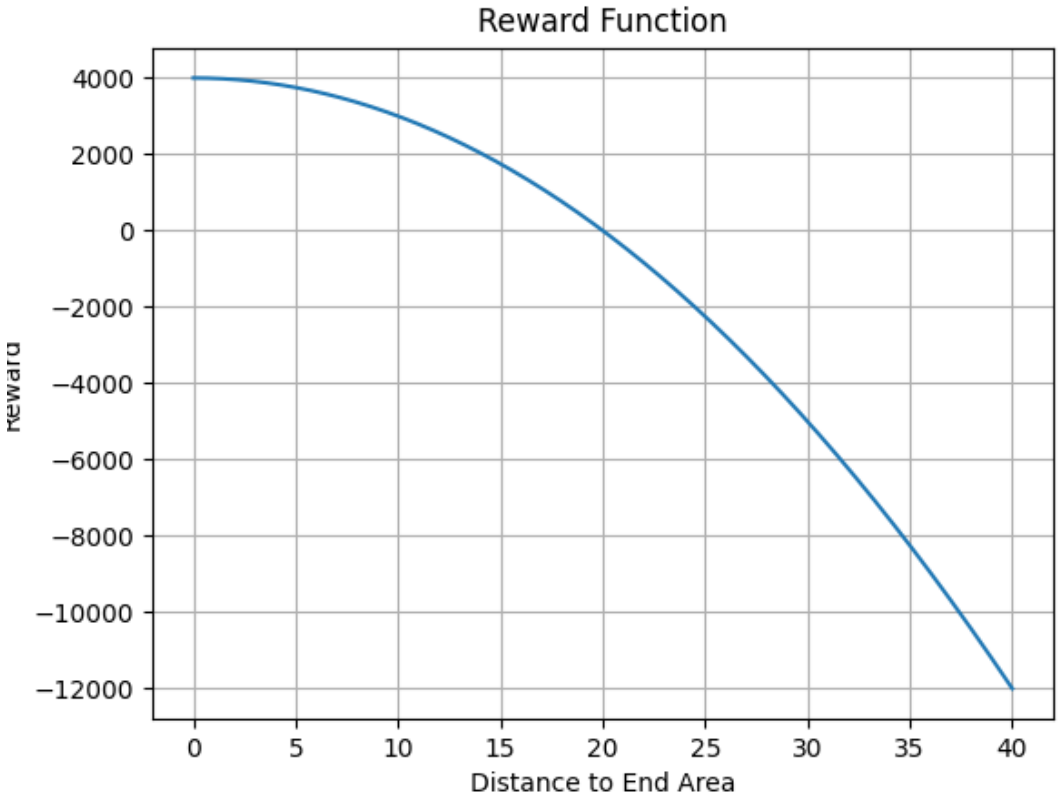
撞到牆壁，給予一個小的負面 reward

接著判斷三個方向的 rew 值，判斷依據是是否離牆很近，如果距離<6.5，則給予蠻大的負面 reward，離>6.5 則無事發生，然後把三個方向的 rew 值加總



rew function 的圖表

再來判斷車輛到終點的距離，距離<20 有中幅的正面 reward，>20 有中幅的負面 reward



4. 分析

exploration_decay 不要設太小，也要給後續的路留一些探索的機會

Q table 不要設太大，否則跑 1000epoch 也跑不完

用離牆距離判斷應該不是最佳解，不然車頭過了他就會認為離牆很遠很安全，然後尾巴就撞車了