AWS DeepRacer



Curso : IABD

Asignatura : MIAR

Autor: Guillermo Fora Goncer

# ÍNDICE

[**ÍNDICE 2**](#_we3wwj5qmghf)

[**Modelo: Ferni 3**](#_8nv0lkesud3w)

[Parámetros 3](#_5o6fadq67v2p)

[Función de recompensa 4](#_77iietnjv0)

[Evaluación 6](#_olc44k42bts4)

[**Modelo: COMPETI 6**](#_yao992y9pk6q)

[Parámetros 6](#_o9zmvgjadkj)

[Función de recompensa 7](#_567pilyb2mmr)

[Gráfica de progreso de entrenamiento 8](#_lr1l5enw0so7)

[Evaluación 8](#_co5swuukil7n)

[Modelo: Competi-clone 9](#_2p8q5y8og7w)

[Parámetros 10](#_wl6lox6bjyz8)

[Función de recompensa 10](#_gev2knj9h95i)

[Gráfica de progreso de entrenamiento 12](#_uxtxm4tj0d5j)

[Evaluación 12](#_octhxczgomc0)

[**Modelo- Megacar 13**](#_kaak2htnj3we)

[Parámetros 13](#_hr8t1z3tw1zv)

[Función de recompensa 13](#_kyzieefmjtwv)

[Gráfica de progreso de entrenamiento 15](#_lycotvplth1)

[Evaluación 16](#_pgqlpsx1tlth)

[Conclusión 17](#_gmv3lbv4n5fo)

[**Modelo: Waypointer 18**](#_mtop9xmd421v)

[Parámetros 18](#_pbvroleu2m58)

[Función de recompensa 18](#_pp39i4wp24lb)

[Gráfica de progreso de entrenamiento 20](#_1vlhappwmpk1)

[Evaluación 21](#_fko845giidc8)

[Conclusión 22](#_zdvhae7uqlzy)

[**Finally 23**](#_set7g7457edw)

[Parámetros 23](#_3hky2zhovti5)

[Función de recompesa 23](#_fnq3v8485457)

[Gráfica de progreso de entrenamiento 24](#_p2iym855bx1e)

[Evaluación 25](#_vf2yqq60iyay)

[**Finally-clone 25**](#_goo77vbbd8by)

[Parámetros 25](#_g9rs9d230a94)

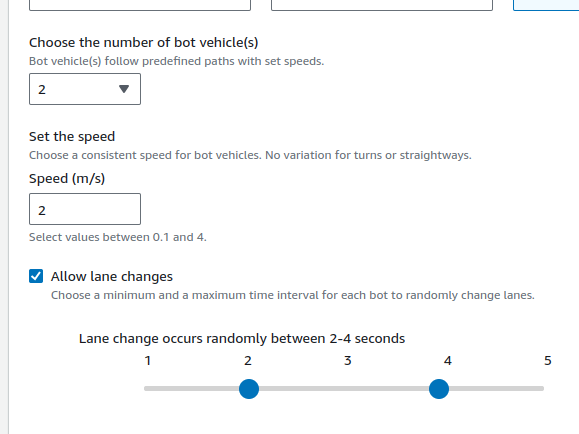
[Función de recompensa 25](#_k87qw2e7gajp)

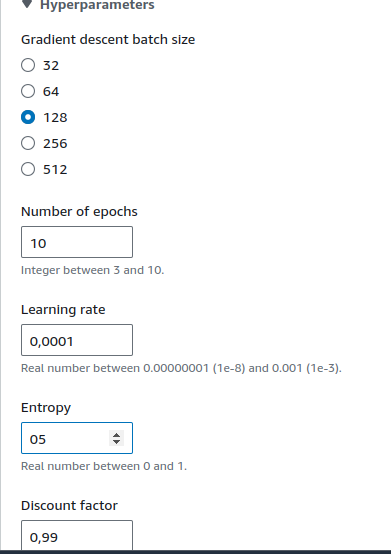
[Gráfica de progreso de entrenamiento 26](#_8esjzwlor37k)

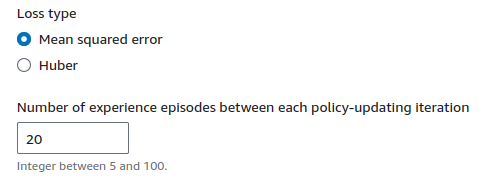
[Evaluación 26](#_lenstvklepvg)

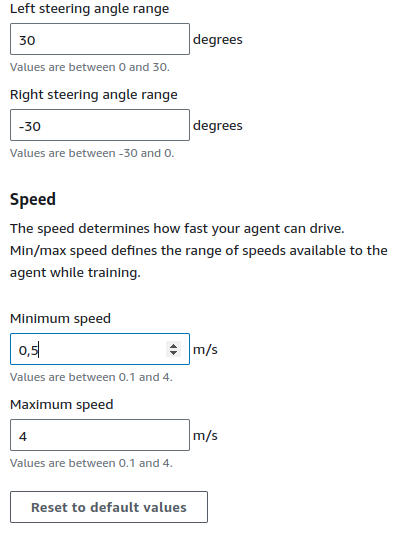
# Modelo: Ferni

## Parámetros









## Función de recompensa

def reward\_function(params):

'''

Function that rewards the agent to stay on the track, avoid collisions,

and overtake other cars. It penalizes when the agent goes off-track

or crashes into other cars.

'''

all\_wheels\_on\_track = params['all\_wheels\_on\_track']

distance\_from\_center = params['distance\_from\_center']

track\_width = params['track\_width']

objects\_distance = params['objects\_distance']

closest\_objects = params['closest\_objects']

objects\_left\_of\_center = params['objects\_left\_of\_center']

is\_left\_of\_center = params['is\_left\_of\_center']

speed = params['speed']

progress = params['progress'] # Progreso actual de la carrera (0 a 100)

# Inicializa la recompensa con un valor pequeño

reward = 1.0

# Penaliza si el coche se sale de la pista

if not all\_wheels\_on\_track:

reward = 1e-3 # Penalización grave por salirse de la pista

return reward # Si se sale de la pista, devuelve una recompensa muy baja

# Recompensa por mantenerse en el carril y a una distancia segura del centro

reward\_lane = 1.0

if distance\_from\_center <= 0.1 \* track\_width:

reward\_lane = 1.0 # Más cerca del centro de la pista (posición ideal)

elif distance\_from\_center <= 0.25 \* track\_width:

reward\_lane = 0.8 # En la parte interna de la pista

elif distance\_from\_center <= 0.5 \* track\_width:

reward\_lane = 0.5 # Un poco alejado del centro pero aún en pista

else:

reward\_lane = 0.1 # Cerca de los bordes de la pista (riesgoso)

# Si el coche está demasiado cerca del borde, dividimos la recompensa

if distance\_from\_center > 0.5 \* track\_width:

reward \*= 0.5 # Recompensa reducida si el coche está fuera de los límites de seguridad

# Recompensa por velocidad - intenta ir lo más rápido posible

reward\_speed = speed / 4 # La velocidad está normalizada, max velocidad puede ser 4 m/s

# Evitar colisiones con otros coches

reward\_avoid\_collision = 1.0

if isinstance(closest\_objects, tuple) and len(closest\_objects[0]) > 0:

for i in range(len(closest\_objects[0])): # Itera a través de todos los objetos más cercanos

# Obtiene el índice del objeto más cercano

index = closest\_objects[0][i]

# Verifica la distancia al coche más cercano

distance\_closest\_object = objects\_distance[index]

is\_same\_lane = objects\_left\_of\_center[index] == is\_left\_of\_center

if is\_same\_lane:

# Penaliza si está demasiado cerca del objeto

if 0.0 < distance\_closest\_object < 0.3:

reward\_avoid\_collision = 1e-3 # Choque inminente, recompensa muy baja

elif 0.3 <= distance\_closest\_object < 0.5:

reward\_avoid\_collision = 0.5 # Bastante cerca del coche

elif 0.5 <= distance\_closest\_object < 0.8:

reward\_avoid\_collision = 0.8 # Distancia segura, pero cuidado

# Recompensa por adelantar a otros coches - si el coche está ganando terreno

reward\_overtake = 1.0

if progress > 0: # Si el coche ha avanzado en la carrera

reward\_overtake = 1.5 # Premio extra por avanzar en la carrera

# Calcular la recompensa final, combinando todos los aspectos

reward += reward\_lane \* 1.5 + reward\_speed \* 2 + reward\_avoid\_collision \* 2 + reward\_overtake

return reward

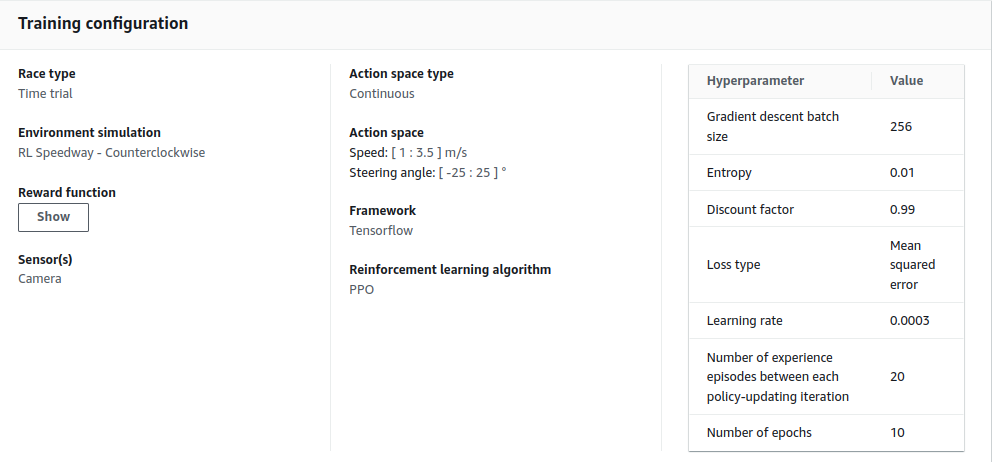
## Evaluación

Este modelo se borro en la primera cuenta asi que no puedo sacar evaluación ni conclusion.

# Modelo: COMPETI

(time trial, RL Speedway - Counterclockwise,PPO, 60 min de entrenamiento)

## Parámetros



## Función de recompensa

def reward\_function(params):

'''

Recompensa al agente por mantener las ruedas dentro de la pista, ser eficiente en el progreso,

penalizarse si se sale y premiarlo por ir al centro de la pista.

'''

# Leer parámetros de entrada

all\_wheels\_on\_track = params['all\_wheels\_on\_track'] # Si todas las ruedas están en la pista

distance\_from\_center = params['distance\_from\_center'] # Distancia del centro de la pista

track\_width = params['track\_width'] # Ancho de la pista

progress = params['progress'] # Progreso en la pista (0 a 100)

steps = params['steps'] # Número de pasos completados

# Inicializar recompensa base

reward = 1.0

# Recompensa por mantener las ruedas dentro de la pista

if all\_wheels\_on\_track:

reward += 2 # Si todas las ruedas están en la pista, se suman 2 puntos

else:

reward -= 1.25 # Penalización si alguna rueda está fuera de la pista

# Recompensa por eficiencia: Progreso rápido con pocos pasos

if progress >= 80:

if steps < 15:

reward \*= 1000 # Recompensa masiva por ser muy eficiente

elif steps < 25:

reward \*= 100 # Gran recompensa por ser muy eficiente

elif steps < 50:

reward \*= 10 # Buena recompensa por ser eficiente

elif steps < 200:

reward \*= 3 # Multiplicador por eficiencia media

else:

reward \*= 1 # Sin multiplicador si no es eficiente

# Calculate 3 markers that are at varying distances away from the center line

marker\_1 = 0.1 \* track\_width

marker\_2 = 0.25 \* track\_width

marker\_3 = 0.5 \* track\_width

# Give higher reward if the car is closer to center line and vice versa

if distance\_from\_center <= marker\_1:

reward = 1.0

elif distance\_from\_center <= marker\_2:

reward = 0.5

elif distance\_from\_center <= marker\_3:

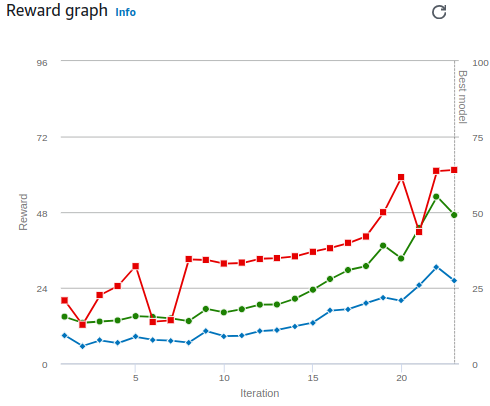
reward = 0.1

else:

reward = 1e-3 # likely crashed/ close to off track

return float(reward)

## Gráfica de progreso de entrenamiento



## Evaluación

Pista : RL Speedway

Sentido: Counterclockwise

| **Trial** | **Time (MM:SS.mmm)** | **Trial results (% track completed)** | **Status** | **Off-track** | **Off-track penalty** | **Crashes** | **Crash penalty** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 00:15.934 | 100% | Lap complete | 1 | 2 seconds | 0 | -- |
| 2 | 00:18.340 | 100% | Lap complete | 2 | 4 seconds | 0 | -- |
| 3 | 00:18.206 | 100% | Lap complete | 2 | 4 seconds | 0 | -- |

Pista : RL Speedway

Sentido: Clockwise

| **Trial** | **Time (MM:SS.mmm)** | **Trial results (% track completed)** | **Status** | **Off-track** | **Off-track penalty** | **Crashes** | **Crash penalty** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 00:34.383 | 100% | Lap complete | 8 | 16 seconds | 0 | -- |
| 2 | 00:35.144 | 100% | Lap complete | 8 | 16 seconds | 0 | -- |
| 3 | 00:32.081 | 100% | Lap complete | 7 | 14 seconds | 0 | -- |

## Conclusión

Como lo hemos entrenado en sentido contrario las agujas del reloj, si lo evaluamos al revés se sale muchísimas veces. Pasa de salirse 2 veces por vuelta a salirse 8 veces por vuelta. Este modelo podría competir en el circuito entrenado pero se le podria penaliza mas las salidas para que sea mas rapido.

# Modelo: Competi-clone

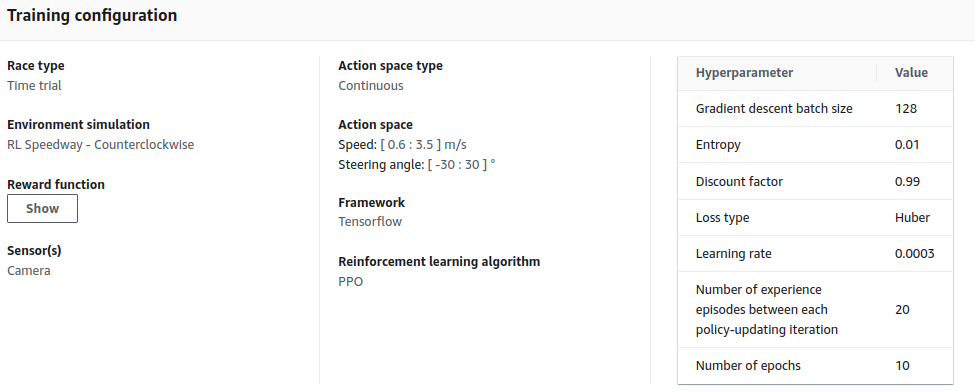
(time trial, RL Speedway - Counterclockwise,PPO, 60 min de entrenamiento)

Vamos a clonar el modelo Competi pero vamos a cambiar varias cosas como el loss type que en el modelo anterior hemos puesto mean square error y entre modelo vamos a probar huber.

Vamos a cambiar también la velocidad y vamos a darle que pueda ir a 0,6 para ver si se sale menos en las curvas.

También vamos a cambiar un poco la función de recompensa, vamos a cambiar la recompensa por eficacia y aplicamos una penalización si se va demasiado lejos de la pista

## Parámetros



## Función de recompensa

def reward\_function(params):

'''

Recompensa al agente por mantener las ruedas dentro de la pista, ser eficiente en el progreso,

penalizarse si se sale y premiarlo por ir al centro de la pista.

'''

# Leer parámetros de entrada

all\_wheels\_on\_track = params['all\_wheels\_on\_track'] # Si todas las ruedas están en la pista

distance\_from\_center = params['distance\_from\_center'] # Distancia del centro de la pista

track\_width = params['track\_width'] # Ancho de la pista

progress = params['progress'] # Progreso en la pista (0 a 100)

steps = params['steps'] # Número de pasos completados

# Inicializar recompensa base

reward = 1.0

# Recompensa por mantener las ruedas dentro de la pista

if all\_wheels\_on\_track:

reward += 2 # Si todas las ruedas están en la pista, se suman 2 puntos

else:

reward -= 3.0 # Penalización más fuerte si alguna rueda está fuera de la pista

# Recompensa por eficiencia: Progreso rápido con pocos pasos

if progress >= 80:

if steps < 20:

reward \*= 500 # Recompensa masiva por ser muy eficiente

elif steps < 30:

reward \*= 50 # Gran recompensa por ser muy eficiente

elif steps < 50:

reward \*= 5 # Buena recompensa por ser eficiente

elif steps < 200:

reward \*= 2 # Multiplicador por eficiencia media

else:

reward \*= 1 # Sin multiplicador si no es eficiente

# Cálculo de las zonas en la pista (distancia al centro)

marker\_1 = 0.1 \* track\_width

marker\_2 = 0.25 \* track\_width

marker\_3 = 0.5 \* track\_width

# Recompensa por estar más cerca del centro de la pista

if distance\_from\_center <= marker\_1:

reward += 1.0 # Buena recompensa por estar cerca del centro

elif distance\_from\_center <= marker\_2:

reward += 0.5 # Recompensa moderada

elif distance\_from\_center <= marker\_3:

reward += 0.1 # Recompensa menor

else:

reward += 0.05 # Recompensa pequeña, ya que está fuera del centro pero dentro de la pista

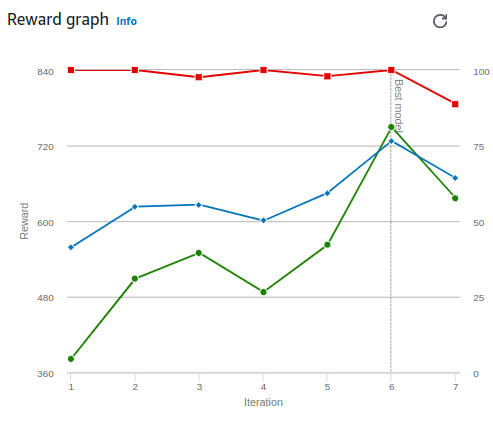
# Penalización si se va demasiado lejos de la pista

if distance\_from\_center > track\_width / 2:

reward = 1e-3 # Penalización fuerte por estar cerca del borde

return float(reward)

## Gráfica de progreso de entrenamiento



## Evaluación

RL Speedway counterclockwise 3 laps

| **Trial** | **Time (MM:SS.mmm)** | **Trial results (% track completed)** | **Status** | **Off-track** | **Off-track penalty** | **Crashes** | **Crash penalty** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 00:16.259 | 100% | Lap complete | 0 | -- | 0 | -- |
| 2 | 00:16.202 | 100% | Lap complete | 0 | -- | 0 | -- |
| 3 | 00:15.733 | 100% | Lap complete | 0 | -- | 0 | -- |

RL Speedway clockwise 3 laps

| **Trial** | **Time (MM:SS.mmm)** | **Trial results (% track completed)** | **Status** | **Off-track** | **Off-track penalty** | **Crashes** | **Crash penalty** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 00:25.520 | 100% | Lap complete | 3 | 6 seconds | 0 | -- |
| 2 | 00:23.330 | 100% | Lap complete | 2 | 4 seconds | 0 | -- |
| 3 | 00:26.077 | 100% | Lap complete | 3 | 6 seconds | 0 | -- |

## Conclusión

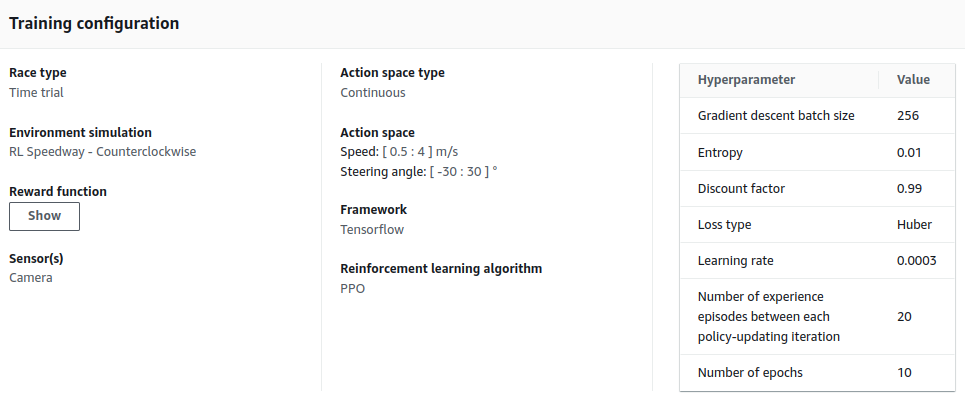
Comprobamos que ahora no se ha salido ninguna vez en el circuito entrenado y el tiempo no está nada mal para la velocidad que va, ya que al poner 0.6 como mínimo el coche no suele ir a más de 2. Si ponemos este modelo en el circuito al revés podemos comprobar que se sale menos veces (3 veces la primera vuelta, 2 la segunda y 3 la tercera). Eso significa que podría ser un modelo bueno para utilizar en otros circuitos.

# Modelo- Megacar

(time trial, RL Speedway - Counterclockwise,PPO, 60 min de entrenamiento)

En este modelo intentamos tocar muchas cosas como premiar por mantener una velocidad rápida sin salirse, intentamos utilizar waypoints para alinear el coche a la pista, penalizar el zigzagueo y le recompensamos si ha terminado el 80 % del circuito.

## Parámetros



## Función de recompensa

import math

def reward\_function(params):

# Extracción de variables

speed = params['speed']

steering\_angle = abs(params['steering\_angle']) # Usamos el valor absoluto para evitar direcciones negativas

distance\_from\_center = params['distance\_from\_center']

is\_offtrack = params['is\_offtrack']

all\_wheels\_on\_track = params['all\_wheels\_on\_track']

track\_width = params['track\_width']

waypoints = params['waypoints']

car\_position = [params['x'], params['y']]

# Nuevas variables necesarias

progress = params['progress'] # Progreso en la pista (porcentaje completado)

steps = params['steps'] # Número de pasos realizados

# Parámetros de la pista

max\_speed = 3.0 # Velocidad máxima permitida (ajústalo según el entorno)

min\_distance\_from\_center = 0.1 \* track\_width # 10% del ancho de la pista, ajustable

# Inicializar la recompensa

reward = 1.0

# Penalizar si el coche se sale de la pista

if is\_offtrack:

return 1e-3 # Recompensa muy pequeña si se sale de la pista

# Recompensar por mantener las 4 ruedas dentro de la pista

if not all\_wheels\_on\_track:

reward \*= 0.5 # Penalización moderada si no todas las ruedas están en la pista

# Recompensar por velocidad alta sin perder el control

if speed < max\_speed:

reward += 0.5 # Recompensa por estar cerca de la velocidad máxima

# Penalizar si el coche está demasiado cerca de los bordes de la pista

# Esto asegura que el coche no se salga en las curvas

if distance\_from\_center > min\_distance\_from\_center:

reward \*= 0.5 # Penaliza si está demasiado lejos del centro de la pista

# Penalizar zigzagueo, es decir, si el ángulo de dirección es muy alto

if steering\_angle > 15: # Umbral ajustable según el comportamiento deseado

reward \*= 0.8 # Penalización suave por zigzagueo

# Usar los waypoints para evaluar la alineación del coche con la pista

# El coche debe seguir la dirección de los waypoints para estar alineado con la pista

closest\_waypoint = params['closest\_waypoints']

waypoint\_a = waypoints[closest\_waypoint[0]]

waypoint\_b = waypoints[closest\_waypoint[1]]

# Dirección del siguiente segmento de la pista

track\_direction = math.atan2(waypoint\_b[1] - waypoint\_a[1], waypoint\_b[0] - waypoint\_a[0]) # Dirección ideal hacia el siguiente waypoint

# Calculamos el ángulo de alineación (sin usar yaw)

angle\_difference = abs(math.atan2(car\_position[1] - waypoint\_a[1], car\_position[0] - waypoint\_a[0]) - track\_direction)

# Recompensar la alineación con la pista

if angle\_difference < math.radians(10): # Si el coche está alineado con la pista

reward += 1.0 # Recompensa por estar alineado

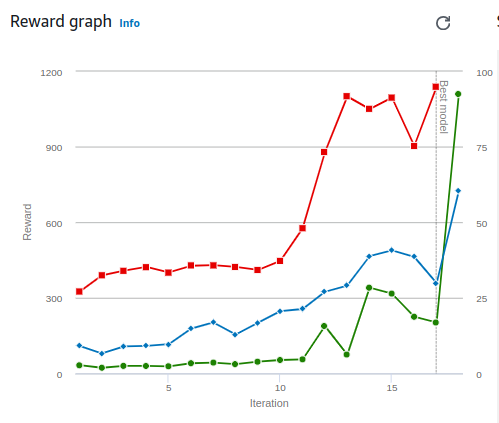
# Recompensar el progreso en el circuito, usando el avance global (progreso)

if progress >= 80:

reward += 50

return reward

## Gráfica de progreso de entrenamiento



## Evaluación

RL Speedway counterclockwise 3 laps

| **Trial** | **Time (MM:SS.mmm)** | **Trial results (% track completed)** | **Status** | **Off-track** | **Off-track penalty** | **Crashes** | **Crash penalty** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 00:14.192 | 100% | Lap complete | 0 | -- | 0 | -- |
| 2 | 00:15.691 | 100% | Lap complete | 1 | 2 seconds | 0 | -- |
| 3 | 00:17.142 | 100% | Lap complete | 1 | 2 seconds | 0 | -- |

RL Speedway clockwise 3 laps

| **Trial** | **Time (MM:SS.mmm)** | **Trial results (% track completed)** | **Status** | **Off-track** | **Off-track penalty** | **Crashes** | **Crash penalty** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 00:26.792 | 100% | Lap complete | 4 | 8 seconds | 0 | -- |
| 2 | 00:26.337 | 100% | Lap complete | 4 | 8 seconds | 0 | -- |
| 3 | 00:25.953 | 100% | Lap complete | 4 | 8 seconds | 0 | -- |

Forever Raceway , counterclockwise, 3 laps

| **Trial** | **Time (MM:SS.mmm)** | **Trial results (% track completed)** | **Status** | **Off-track** | **Off-track penalty** | **Crashes** | **Crash penalty** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 00:24.391 | 100% | Lap complete | 3 | 6 seconds | 0 | -- |
| 2 | 00:22.214 | 100% | Lap complete | 3 | 6 seconds | 0 | -- |
| 3 | 00:24.158 | 100% | Lap complete | 4 | 8 seconds | 0 | -- |

## Conclusión

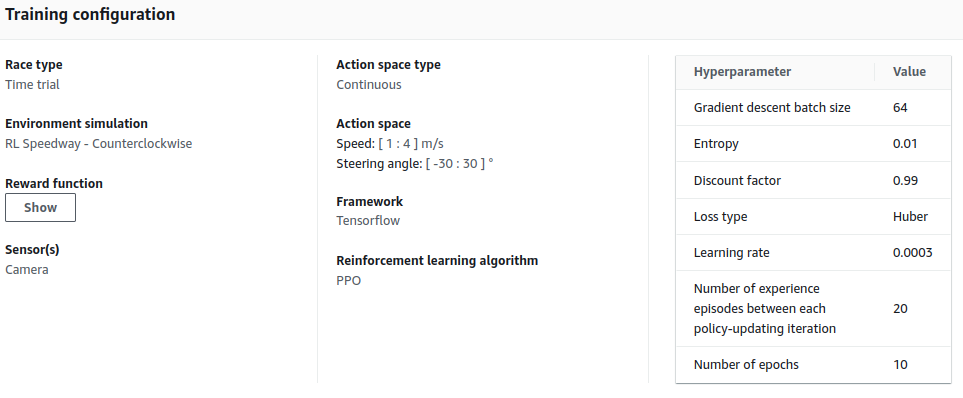
Podemos decir que en la primera evaluación, que es donde se ha entrenado, el coche hace un buen tiempo y solo se sale 2 veces en las 3 vueltas pero si ponemos ese modelo en el mismo circuito (al revés), ya se sale 4 veces en cada vuelta, que tampoco esta mal pero se le podría penalizar más las salidas. De momento este el modelo que ha hecho la vuelta mas rápido de todos. (14 segungos)

# Modelo: Waypointer

(time trial, Forever raceaway - Counterclockwise,PPO, 60 min de entrenamiento)

En este modelo intentamos solo utilizare los waypointer para intentar conseguir que el coche vaya lo mas centrado posible, no se salga de la pista y coga bien las curvas.

## Parámetros



## Función de recompensa

import math

def reward\_function(params):

# Extracción de variables

car\_position = [params['x'], params['y']] # Posición del coche (x, y)

waypoints = params['waypoints'] # Waypoints de la pista

closest\_waypoints = params['closest\_waypoints'] # Los dos waypoints más cercanos al vehículo

speed = params['speed'] # Velocidad del coche

steering\_angle = abs(params['steering\_angle']) # Ángulo de dirección (positivo)

is\_offtrack = params['is\_offtrack'] # Si el coche está fuera de la pista

track\_width = params['track\_width'] # Ancho de la pista

distance\_from\_center = params['distance\_from\_center'] # Distancia al centro de la pista

# Penalización si el coche está fuera de la pista

if is\_offtrack:

return 1e-3 # Recompensa muy pequeña si el coche está fuera de la pista

# Obtener los dos waypoints más cercanos

waypoint\_a = waypoints[closest\_waypoints[0]]

waypoint\_b = waypoints[closest\_waypoints[1]]

# Calcular la distancia perpendicular del vehículo a la línea entre los waypoints

x0, y0 = car\_position

x1, y1 = waypoint\_a

x2, y2 = waypoint\_b

# Fórmula para la distancia perpendicular de un punto a una línea

numerator = abs((x2 - x1) \* (y1 - y0) - (x1 - x0) \* (y2 - y1))

denominator = math.sqrt((x2 - x1)\*\*2 + (y2 - y1)\*\*2)

distance\_to\_ideal\_path = numerator / denominator

# Parámetros de penalización

ideal\_distance\_threshold = 0.2 # Distancia umbral para estar "cerca" de la trayectoria ideal

reward = 1.0 # Recompensa base

# Penalización por estar lejos del centro de la pista

min\_distance\_from\_center = 0.1 \* track\_width # 10% del ancho de la pista

if distance\_from\_center > min\_distance\_from\_center:

reward \*= 0.5 # Penalización si está demasiado lejos del centro

# Penalización proporcional al desvío de la línea ideal

if distance\_to\_ideal\_path > 0.2: # Si está lejos de la línea ideal

reward \*= math.exp(-distance\_to\_ideal\_path \* 5) # Penalización exponencial proporcional al desvío

# Recompensar la proximidad a la trayectoria ideal

if distance\_to\_ideal\_path < ideal\_distance\_threshold:

reward += math.exp(-distance\_to\_ideal\_path \* 10) # Exponencial para incentivar cercanía

# Recompensar velocidad adecuada con la proximidad

if speed > 0.5: # Premio adicional por mantener una velocidad razonable

reward += 0.2

# Penalizar ángulos de dirección grandes (por movimientos bruscos)

if steering\_angle > 15: # Umbral ajustable según la agresividad deseada

reward \*= 0.7 # Penalización suave por giros bruscos

# Recompensar alineación adicional con la trayectoria ideal en función de la dirección

# Evaluamos si el coche está alineado con la línea entre los waypoints

car\_direction = math.atan2(y2 - y1, x2 - x1)

angle\_diff = abs(car\_direction - math.atan2(y0 - y1, x0 - x1))

if angle\_diff < math.radians(10): # Alineación cerca de los waypoints

reward += 0.5 # Aumento adicional si está alineado

# Progreso (opcional, dependiendo del entorno)

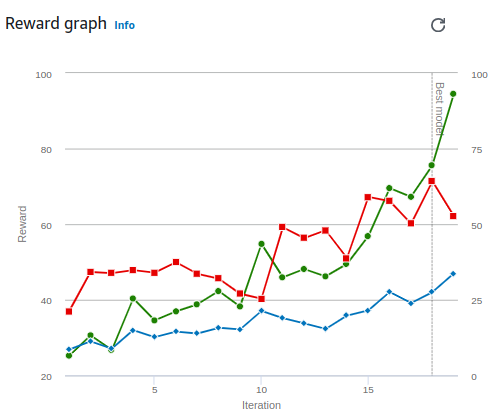
progress = params['progress'] # Porcentaje del circuito completado

if progress >= 80: # Progreso significativo en la pista

reward \*= 2 # Doble recompensa por progreso alto

return reward

## Gráfica de progreso de entrenamiento



## 

## Evaluación

RL Speedway counterclockwise 3 laps

| **Trial** | **Time (MM:SS.mmm)** | **Trial results (% track completed)** | **Status** | **Off-track** | **Off-track penalty** | **Crashes** | **Crash penalty** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 00:22.693 | 100% | Lap complete | 3 | 6 seconds | 0 | -- |
| 2 | 00:19.611 | 100% | Lap complete | 1 | 2 seconds | 0 | -- |
| 3 | 00:20.146 | 100% | Lap complete | 2 | 4 seconds | 0 | -- |

RL Speedway clockwise 3 laps

| **Trial** | **Time (MM:SS.mmm)** | **Trial results (% track completed)** | **Status** | **Off-track** | **Off-track penalty** | **Crashes** | **Crash penalty** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 00:26.792 | 100% | Lap complete | 4 | 8 seconds | 0 | -- |
| 2 | 00:26.337 | 100% | Lap complete | 4 | 8 seconds | 0 | -- |
| 3 | 00:25.953 | 100% | Lap complete | 4 | 8 seconds | 0 | -- |

Forever raceway, counterclockwise, 3 laps

| **Trial** | **Time (MM:SS.mmm)** | **Trial results (% track completed)** | **Status** | **Off-track** | **Off-track penalty** | **Crashes** | **Crash penalty** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 00:24.391 | 100% | Lap complete | 3 | 6 seconds | 0 | -- |
| 2 | 00:22.214 | 100% | Lap complete | 3 | 6 seconds | 0 | -- |
| 3 | 00:24.158 | 100% | Lap complete | 4 | 8 seconds | 0 | -- |

## Conclusión

Este modelo hemos intentado utilizar los waypoints pero no ha sido el más rápido de todos, ni el más efectivo. En la primera evaluación, en el circuito entrenado, podemos comprobar que se ha salido 3 veces en la primera vuelta, 1 en la segunda y 3 en la tercera.

Si ponemos el mismo circuito pero al revés, podemos observar que se sale 4 veces en cada vuelta, así que no es un modelo que sea eficaz aprendiendo y si lo ponemos en otro circuito, comprobamos que es un poco mejor que si lo ponemos en el circuito al pero se sigue saliendo muchas veces (3 veces en la primera vuelta, 3 en la segunda vuelta y 4 en la última vuelta)

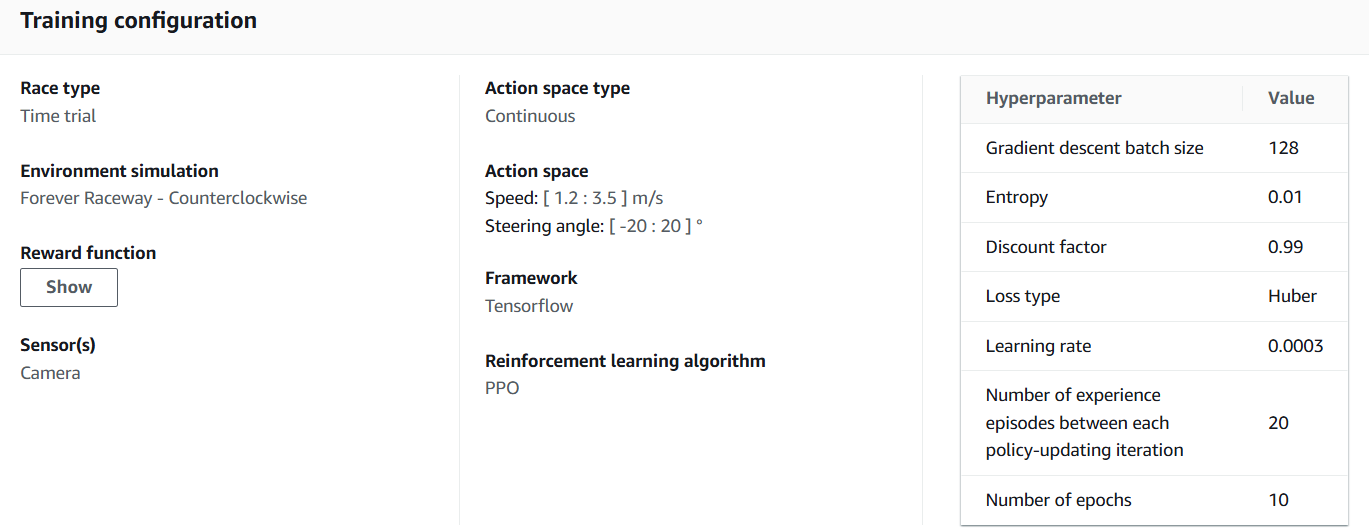
# 

# Finally

(time trial, Forever raceaway - Counterclockwise,PPO, 60 min de entrenamiento)

En este modelo intentamos no tocar mucho la función de recompensa para ver que tiempos hace sin modificarla mucho la recompensa. Le pondremos más velocidad para ver si podemos clasificarnos en la AWS virtual circuit.

## Parámetros



## Función de recompensa

def reward\_function(params):

'''

Recompensa al agente por mantener las ruedas dentro de la pista, ser eficiente en el progreso,

penalizarse si se sale y premiarlo por ir al centro de la pista.

'''

# Leer parámetros de entrada

all\_wheels\_on\_track = params['all\_wheels\_on\_track'] # Si todas las ruedas están en la pista

distance\_from\_center = params['distance\_from\_center'] # Distancia del centro de la pista

track\_width = params['track\_width'] # Ancho de la pista

progress = params['progress'] # Progreso en la pista (0 a 100)

steps = params['steps'] # Número de pasos completados

# Inicializar recompensa base

reward = 1.0

# Recompensa por mantener las ruedas dentro de la pista

if all\_wheels\_on\_track:

reward += 2 # Si todas las ruedas están en la pista, se suman 2 puntos

else:

reward -= 1.25 # Penalización si alguna rueda está fuera de la pista

# Calculate 3 markers that are at varying distances away from the center line

marker\_1 = 0.1 \* track\_width

marker\_2 = 0.25 \* track\_width

marker\_3 = 0.5 \* track\_width

# Give higher reward if the car is closer to center line and vice versa

if distance\_from\_center <= marker\_1:

reward = 1.0

elif distance\_from\_center <= marker\_2:

reward = 0.5

elif distance\_from\_center <= marker\_3:

reward = 0.1

else:

reward = 1e-3 # likely crashed/ close to off track

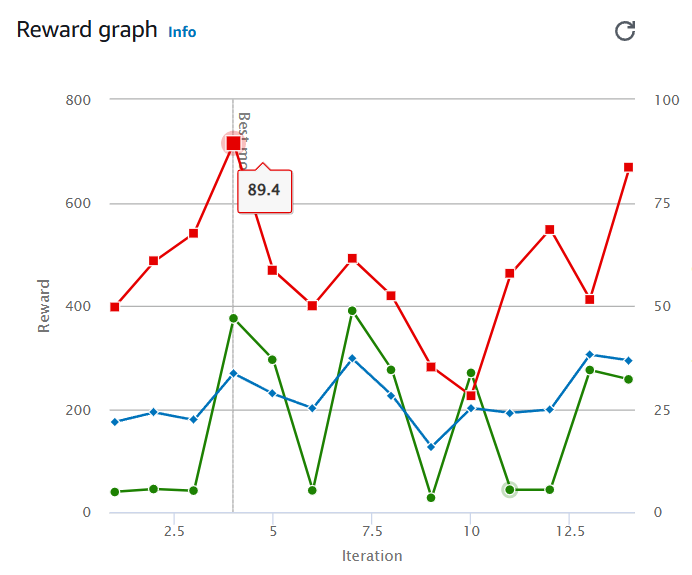
# Recompensa por eficiencia: Progreso rápido con pocos pasos

if progress >= 80:

reward +=100

return float(reward)

## Gráfica de progreso de entrenamiento



## Evaluación

Forever raceaway, counterclockwise, 3 laps

| **Trial** | **Time (MM:SS.mmm)** | **Trial results (% track completed)** | **Status** | **Off-track** | **Off-track penalty** | **Crashes** | **Crash penalty** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 00:18.734 | 100% | Lap complete | 1 | 2 seconds | 0 | -- |
| 2 | 00:17.194 | 100% | Lap complete | 0 | -- | 0 | -- |
| 3 | 00:16.877 | 100% | Lap complete | 0 | -- | 0 | -- |

Forever raceaway, clockwise, 3 laps

| **Trial** | **Time (MM:SS.mmm)** | **Trial results (% track completed)** | **Status** | **Off-track** | **Off-track penalty** | **Crashes** | **Crash penalty** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 00:33.876 | 100% | Lap complete | 7 | 14 seconds | 0 | -- |
| 2 | 00:31.819 | 100% | Lap complete | 6 | 12 seconds | 0 | -- |
| 3 | 00:31.921 | 100% | Lap complete | 6 | 12 seconds | 0 | -- |

## Conclusión

Comprobamos que el coche solo se ha salido una vez en el circuito entrenado y para la velocidad que le hemos puesto me parece que está bastante bien. Si ponemos este coche al revés, vemos que se sale muchas veces así que no servirá para evaluarlo en otro circuito. Se sale 7 veces en la primera vuelta, 6 veces en la segunda y 6 veces en la tercera. De momento ha conseguido un puesto 7 en la clasificación de España.