# Modelos de Inteligencia Artificial

Curso de Especialización de Inteligencia Artificial y Big Data
IES Gran Capitán 2023/24

# Caso práctico con AWS DeepRacer

# AWS DeepRacer

## A closer look

- 1:18 4WD scale car
- Intel Atom processor
- Intel distribution of OpenVINO toolkit
- Stereo camera (4MP)
- 360-degree, 12-meter scanning radius Lidar sensor
- System memory: 4 GB RAM
- 802.11ac Wi-Fi
- Ubuntu 20.04 Focal Fossa
- ROS 2 Foxy Fitzroy

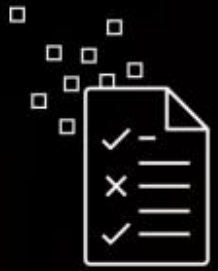
OpenVINO

intel ATOM

aws © 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

# AWS DeepRacer

## ML overview



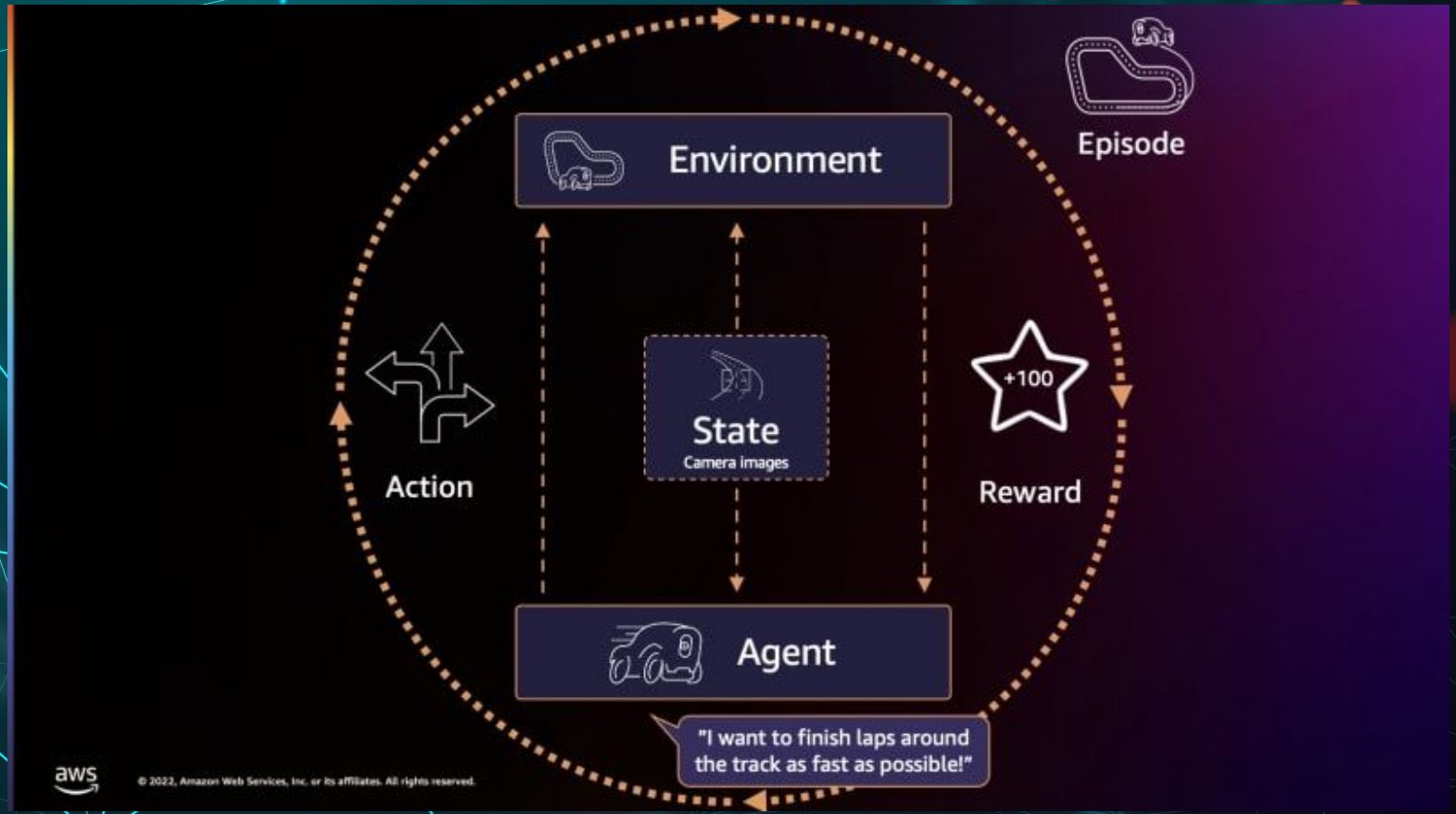
**Supervised**
Example-driven training; every datum has a corresponding label

**Unsupervised**
No labels for training data; useful for clustering like data

**Reinforcement**
Learns through consequences of actions in a specific environment

aws © 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

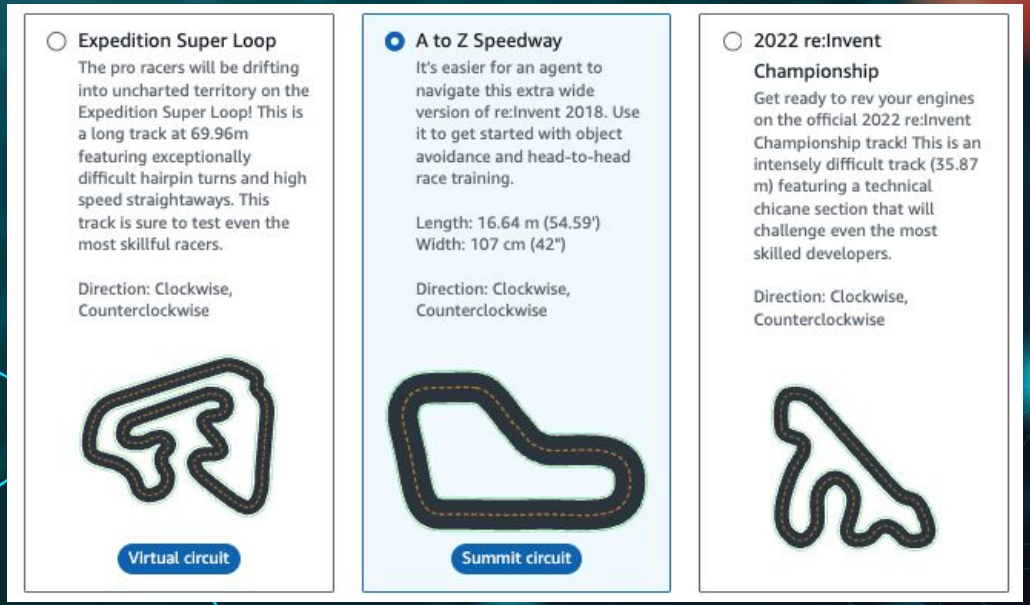
# Elementos de DeepRacer



# Entrenando tu Modelo de AWS DeepRacer

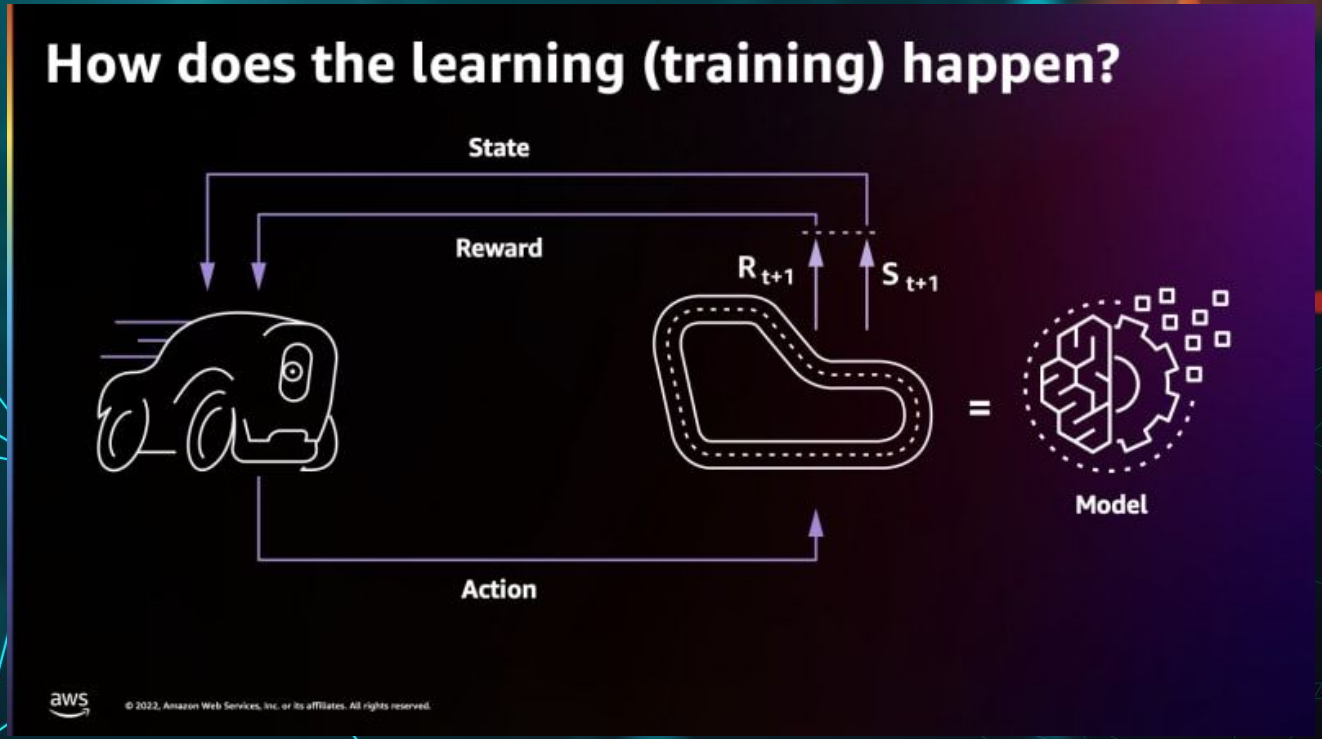
Pasos:

1. Definición del Problema y Recompensas
2. Entorno de Simulación



# Entrenando tu Modelo de AWS DeepRacer

3. Entrenamiento y Evaluación → servicio AWS RoboMaker



# Entrenando tu Modelo de AWS DeepRacer

- 3.1. El agente observa el estado actual (por ejemplo, una imágen de la pista).

3.2. Basándose en ese estado y en lo que ha aprendido hasta ahora, decide realizar una acción (girar a la derecha o izquierda).

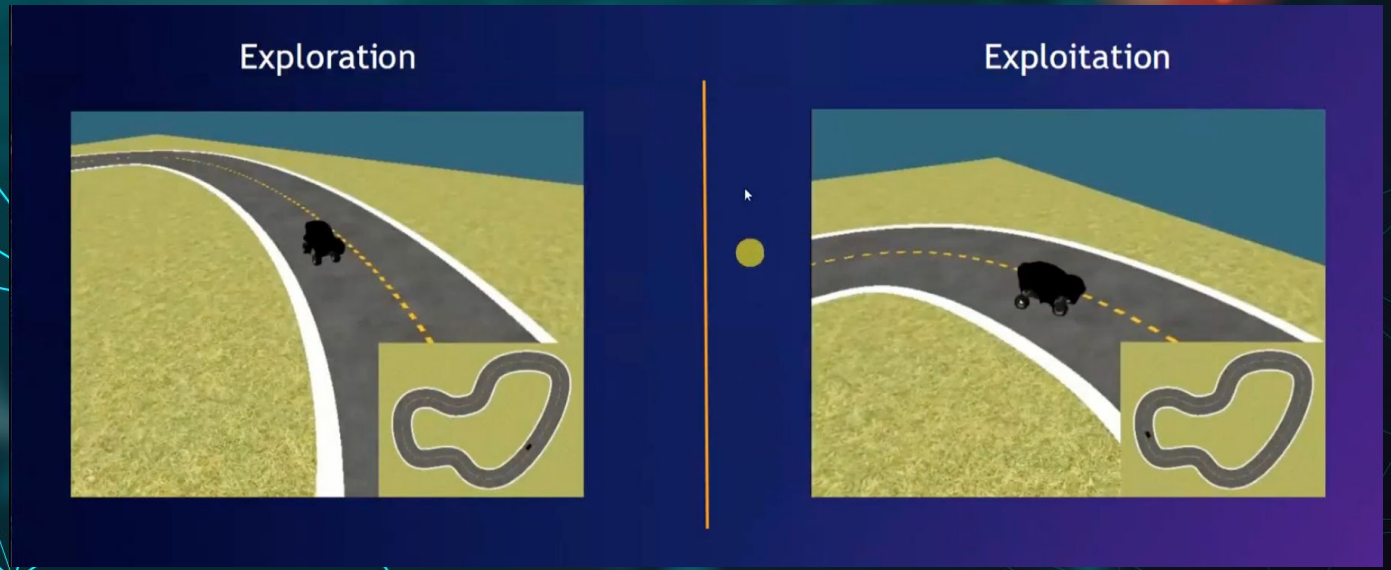
3.3. El agente ejecuta esa acción en la simulación, lo que le lleva a un nuevo estado.

3.4. El agente recibirá una recompensa basada en qué tan buena fue esa acción en ese contexto.

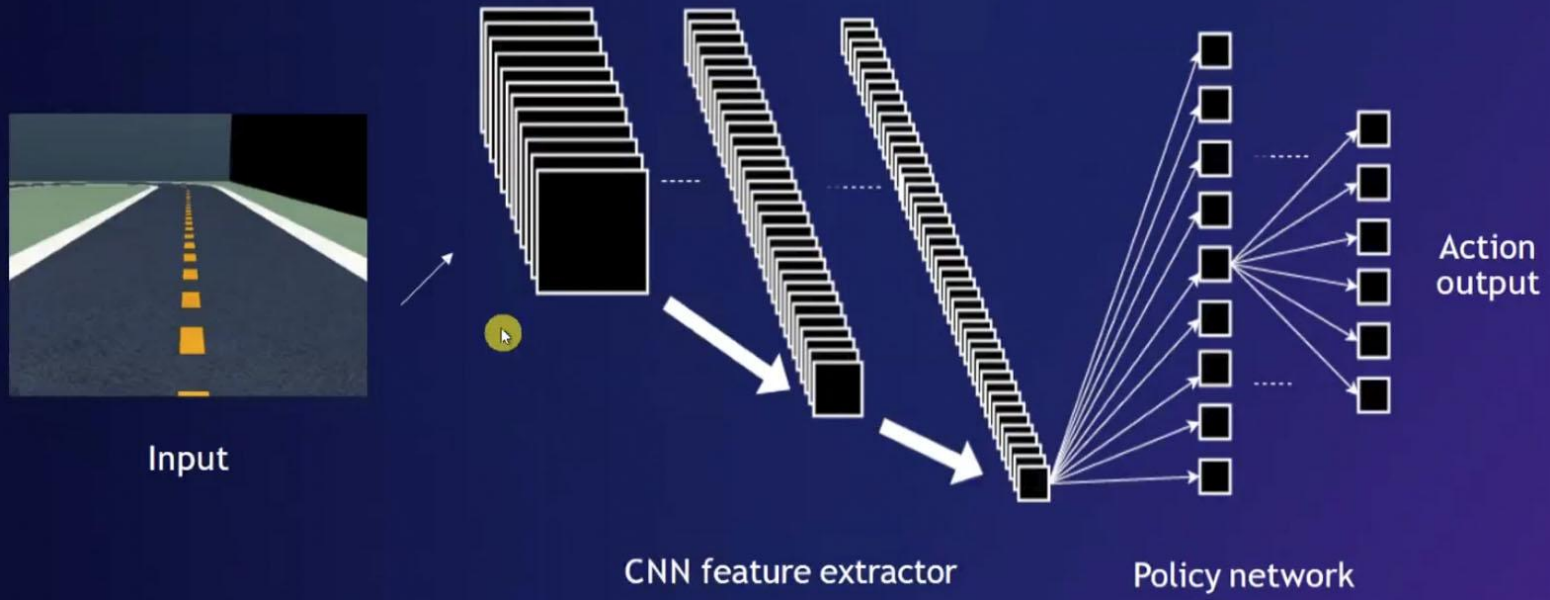
3.5. Con la recompensa recibida, el agente ajusta su comportamiento y mejora sus decisiones futuras.

# Entrenando tu Modelo de AWS DeepRacer

4. Iteración y Ajuste
5. Pruebas en el Mundo Real → cargar el modelo en un coche DeepRacer físico (utilizando OpenVino)



# AWS DeepRacer neural network architecture



# Optimising and inferencing with OpenVINO ™

Input data

OpenVINO ™ optimised model

OpenVINO ™ inference results

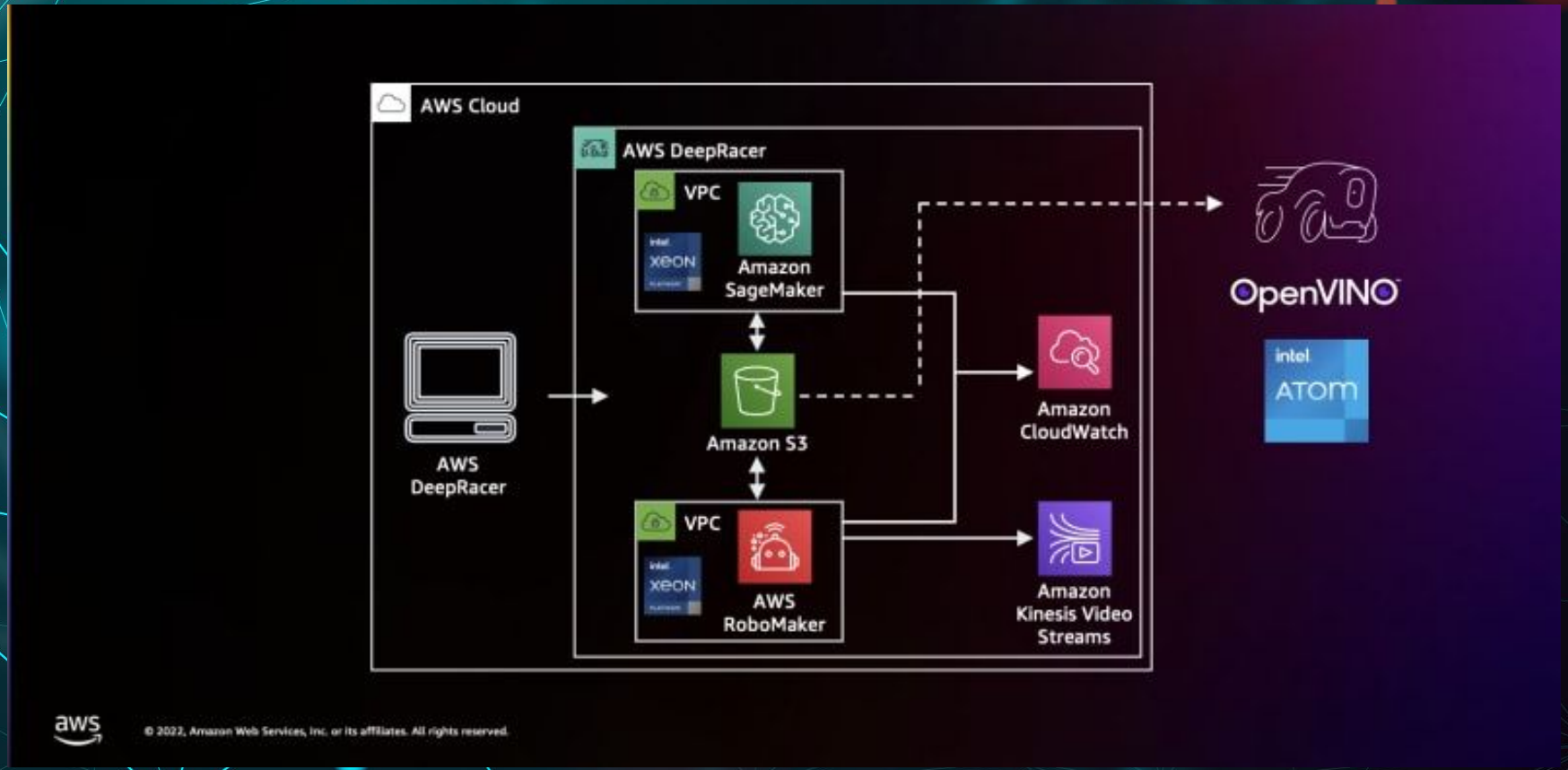
Racing

Free Download: software.intel.com/openvino-toolkit
Open Source version: 01.org/openvinotoolkit

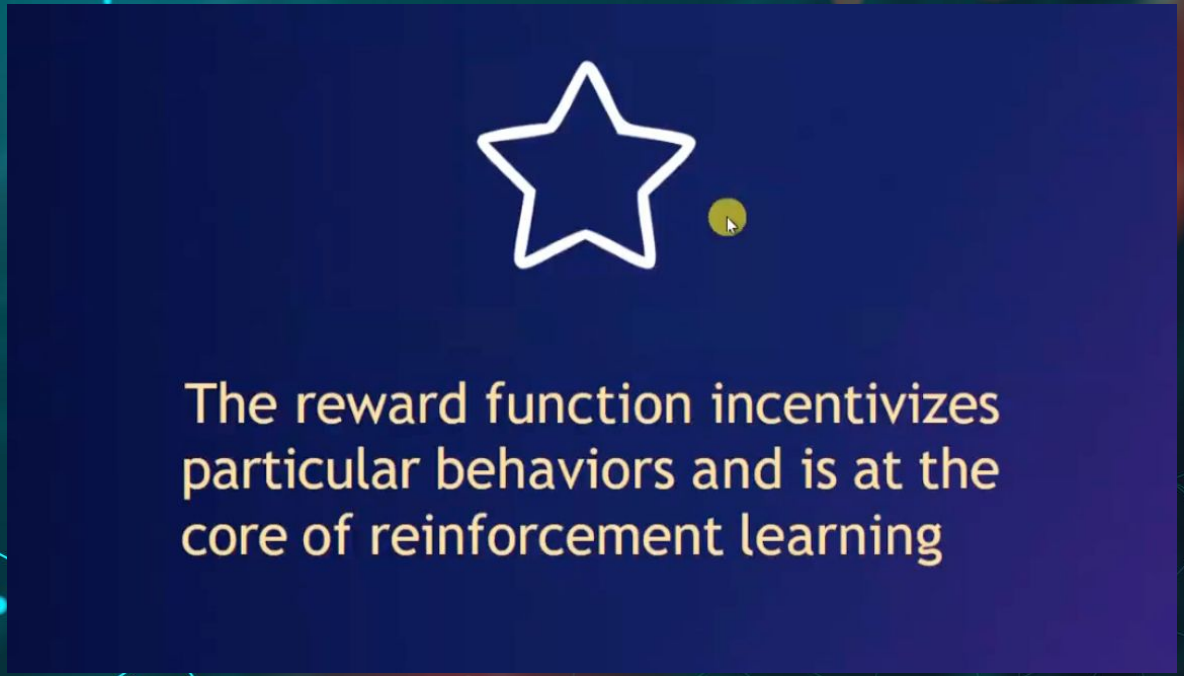
OpenVINO™

intel ATOM

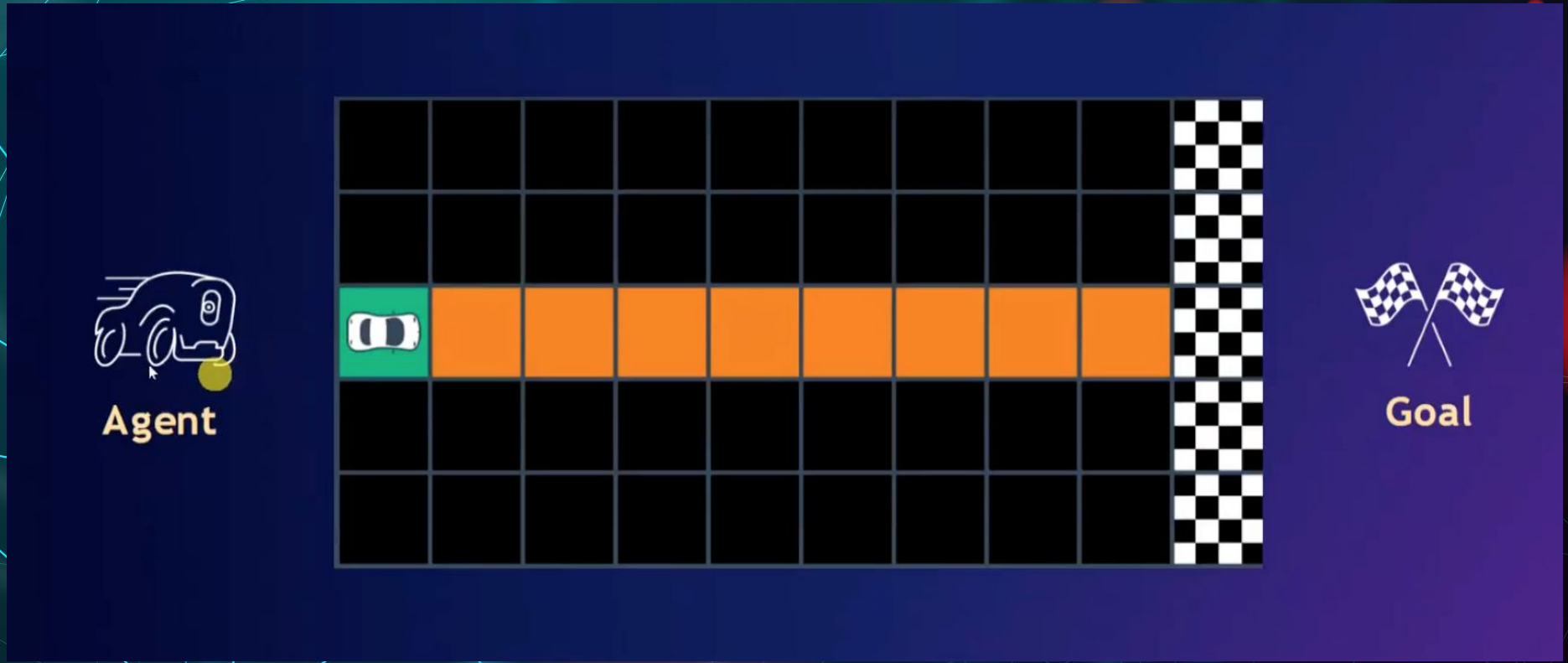
# Arquitectura del Simulador de DeepRacer



# Función de Recompensa



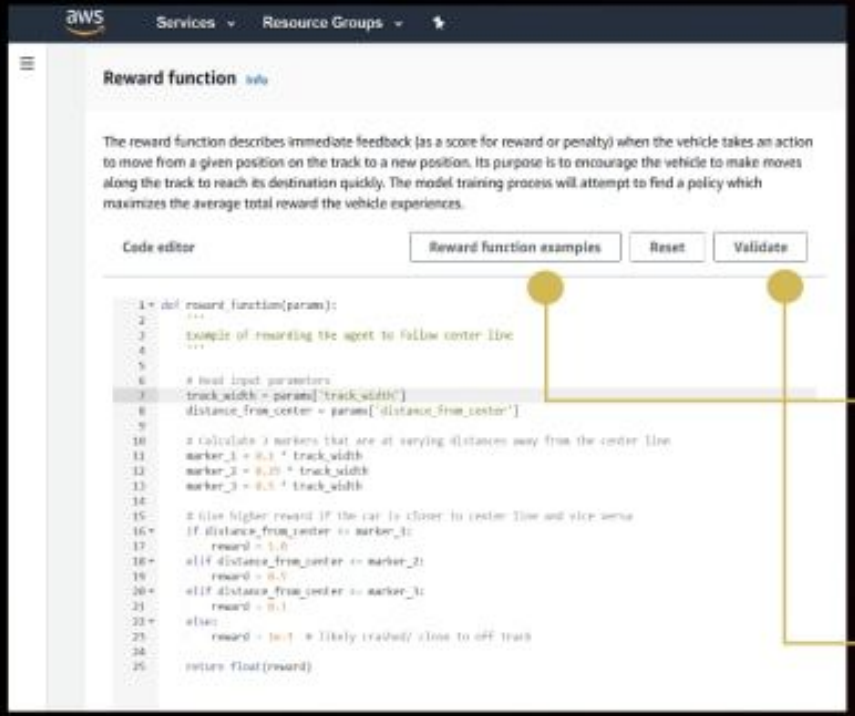
# Función de Recompensa



# Función de Recompensa

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| × | × | × | × | × | × | × | × | × | |
| 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | |
|  | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | |
| 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | |
| × | × | × | × | × | × | × | × | × | |

# Programming your own reward function




Code editor: Python 3 syntax

Three example reward functions

Code validation via Lambda


aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

# Parámetros de la función de Recompensa

```
{
    "all_wheels_on_track": Boolean,        # flag to indicate if the agent is on the track
    "x": float,                            # agent's x-coordinate in meters
    "y": float,                            # agent's y-coordinate in meters
    "closest_objects": [int, int],         # zero-based indices of the two closest objects to the agent's current position of (x, y).
    "closest_waypoints": [int, int],       # indices of the two nearest waypoints.
    "distance_from_center": float,         # distance in meters from the track center
    "is_crashed": Boolean,                 # Boolean flag to indicate whether the agent has crashed.
    "is_left_of_center": Boolean,          # Flag to indicate if the agent is on the left side to the track center or not.
    "is_offtrack": Boolean,                # Boolean flag to indicate whether the agent has gone off track.
    "is_reversed": Boolean,                # flag to indicate if the agent is driving clockwise (True) or counter clockwise (False).
    "heading": float,                      # agent's yaw in degrees
    "objects_distance": [float, ],         # list of the objects' distances in meters between 0 and track_length in relation to the starting line.
    "objects_heading": [float, ],          # list of the objects' headings in degrees between -180 and 180.
    "objects_left_of_center": [Boolean, ], # list of Boolean flags indicating whether elements' objects are left of the center (True) or not (False).
    "objects_location": [(float, float),], # list of object locations [(x,y), ...].
    "objects_speed": [float, ],            # list of the objects' speeds in meters per second.
    "progress": float,                     # percentage of track completed
    "speed": float,                        # agent's speed in meters per second (m/s)
    "steering_angle": float,               # agent's steering angle in degrees
    "steps": int,                          # number steps completed
    "track_length": float,                 # track length in meters.
    "track_width": float,                  # width of the track
    "waypoints": [(float, float), ]        # list of (x,y) as milestones along the track center

}
```

# Ejemplos de funciones de recompensa

1. Recompensa por mantenerse dentro de los límites de la pista:

```
def reward_function(params):
    '''
    Example of rewarding the agent to stay inside the two borders
    '''

    # Read input parameters
    all_wheels_on_track = params['all_wheels_on_track']
    distance_from_center = params['distance_from_center']
    track_width = params['track_width']

    # Give a very low reward by default
    reward = 1e-3

    # Give a high reward if no wheels go off the track and
    # the agent is somewhere in between the track borders
    if all_wheels_on_track and (0.5*track_width - distance_from_ce
        reward = 1.0

    # Always return a float value
    return float(reward)
```

# Ejemplos de funciones de recompensa

2. Recompensa por mantenerse cerca del centro:

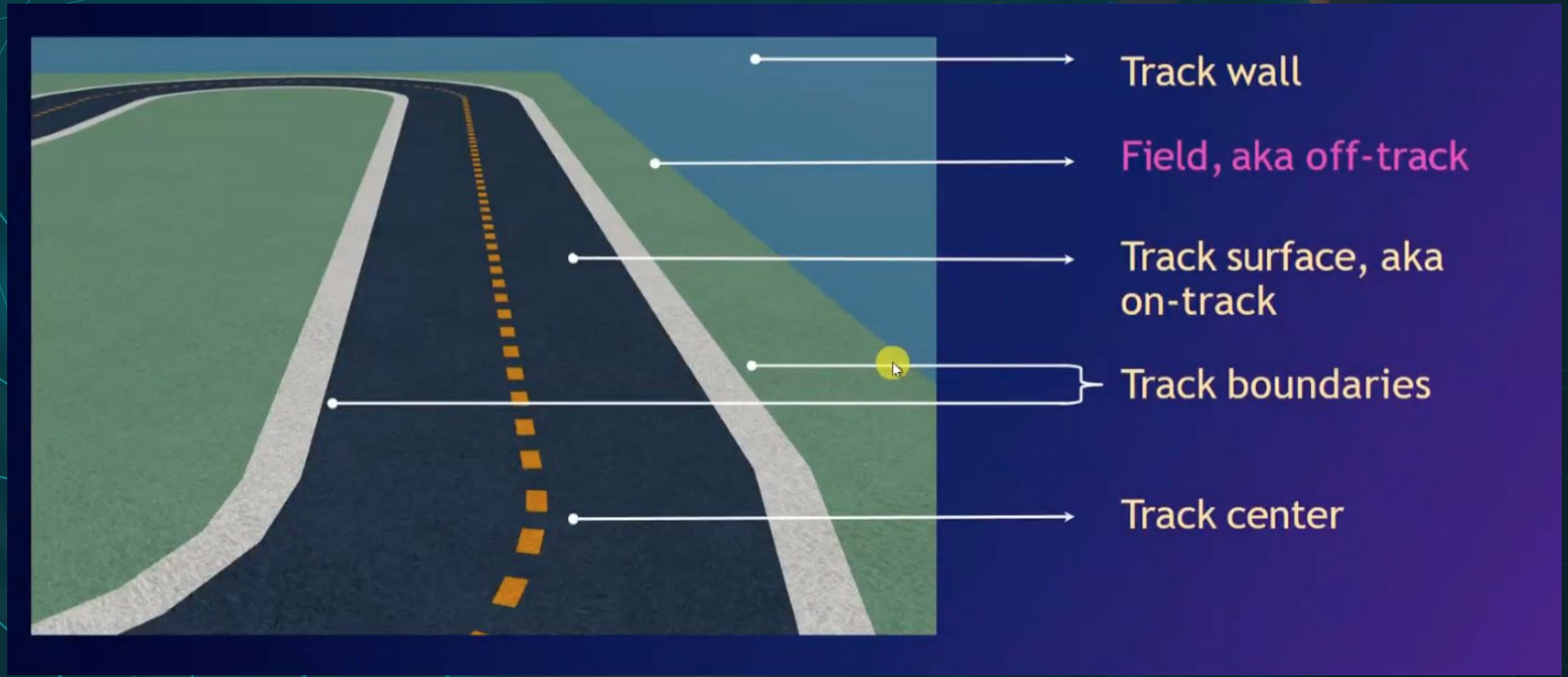
```
def reward_function(params):
    '''
    Example of rewarding the agent to follow center line
    '''

    # Read input parameters
    track_width = params['track_width']
    distance_from_center = params['distance_from_center']

    # Calculate 3 markers that are at varying distances away from
    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 \
    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)
```

# Ejemplos de funciones de recompensa

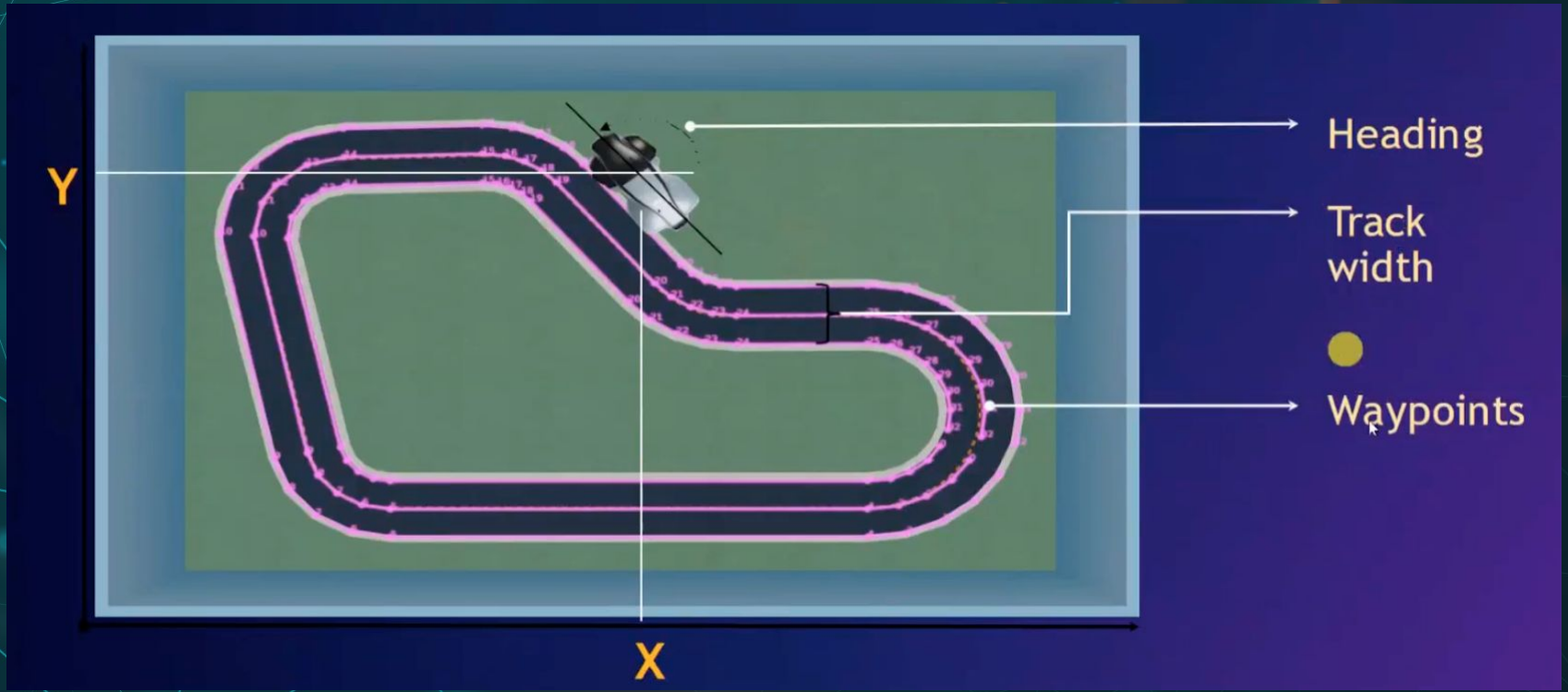
3. Recompensa por evitar zig-zaguear en pista:

```
def reward_function(params):
    '''
    Example of penalize steering, which helps mitigate zig-zag beh
    '''
    # Read input parameters
    distance_from_center = params['distance_from_center']
    track_width = params['track_width']
    abs_steering = abs(params['steering_angle']) # Only need the a
    # Calculate 3 marks that are farther and father away from the
    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 v
    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
    # Steering penality threshold, change the number based on your
    ABS_STEERING_THRESHOLD = 15
    # Penalize reward if the car is steering too much
    if abs_steering > ABS_STEERING_THRESHOLD:
        reward *= 0.8
    return float(reward)
```

# Componentes de la pista

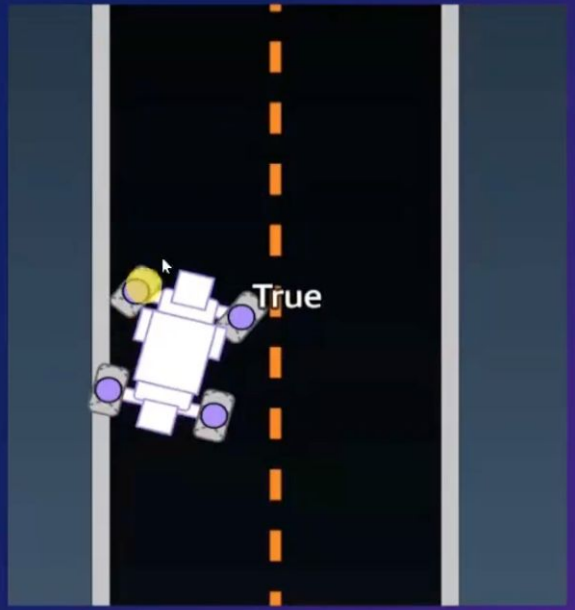


# Componentes de la pista

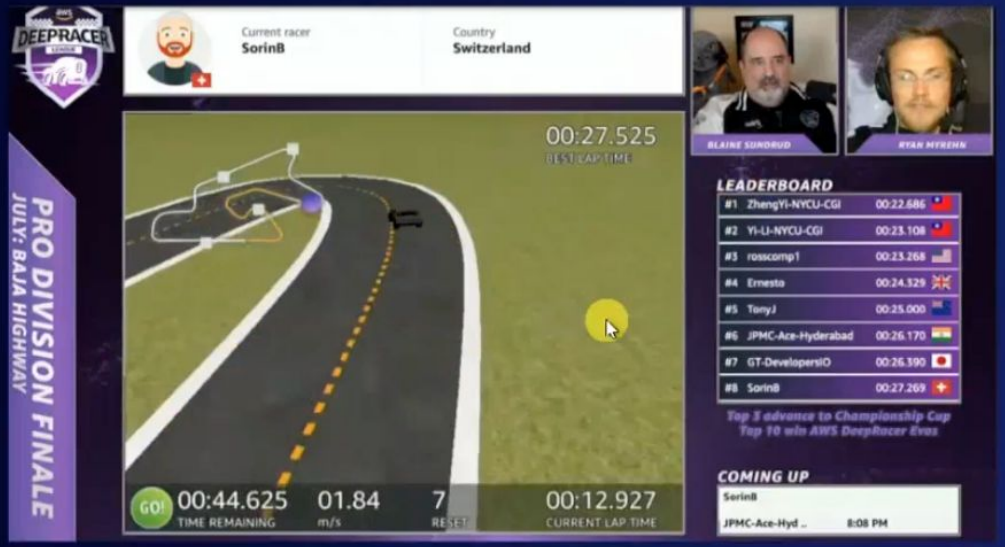


# Componentes de la pista

Example parameter:
all_wheels_ on_track



# Join the 2023 League




# QUALIFY FOR THE MAIN EVENT IN THE PRO DIVISION TO RACE LIVE FOR A CHANCE TO WIN AN AWS DEEPRACER