



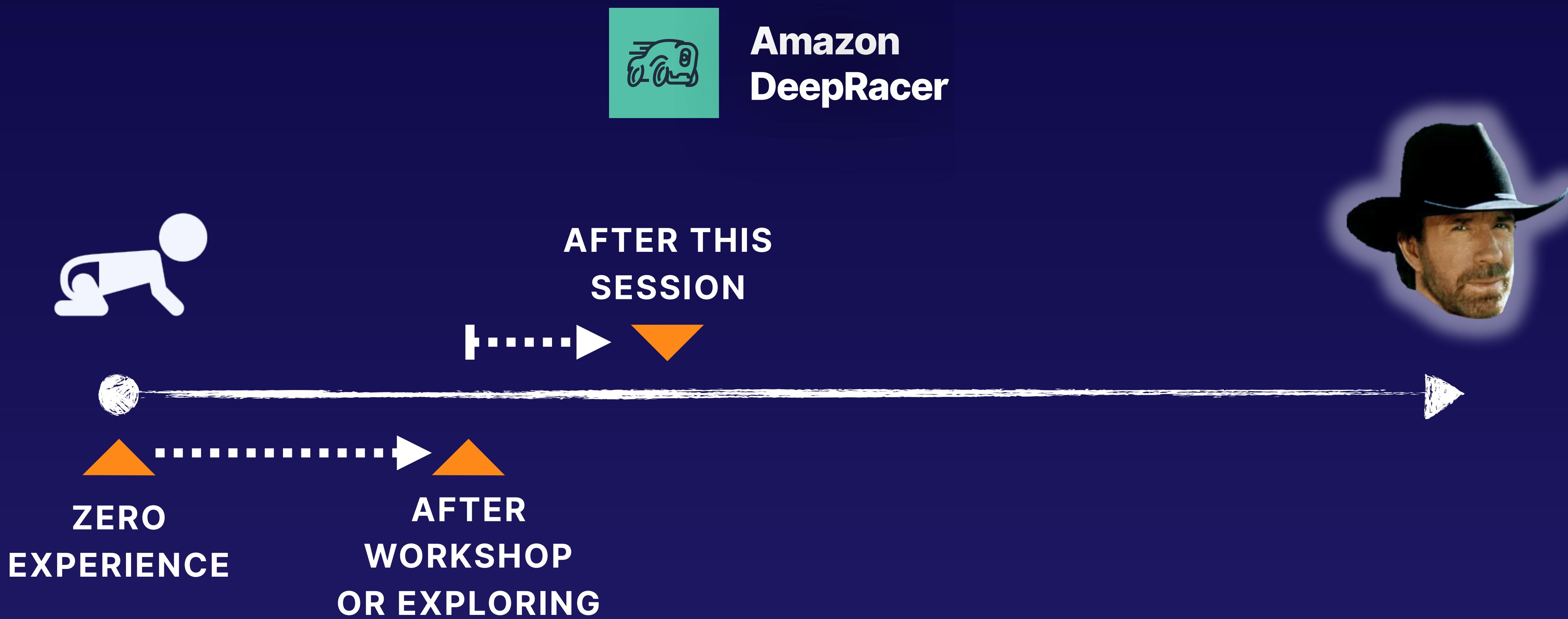
# AWS DeepRacer ~~101~~<sup>102</sup>: Lessons from Victory Lane

Scott Pletcher  
Instructor  
A Cloud Guru



© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

# Continuum of AWS Bad-Assery



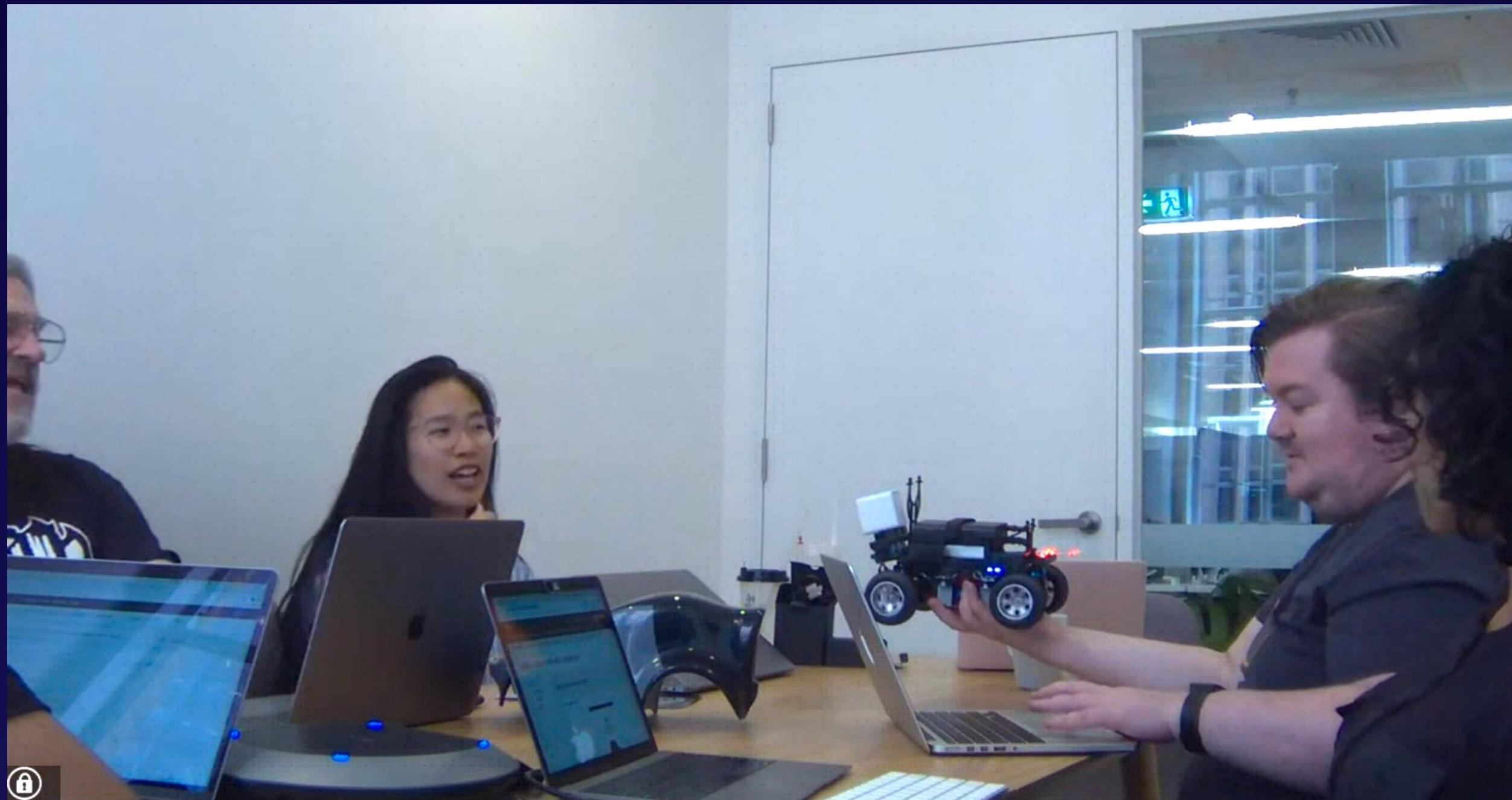


**“AWS DeepRacer is a fully autonomous 1/18th scale race car designed to get you rolling with reinforcement learning.”**

**Dr. Matt Wood**

GM, Deep Learning and AI  
Amazon Web Services



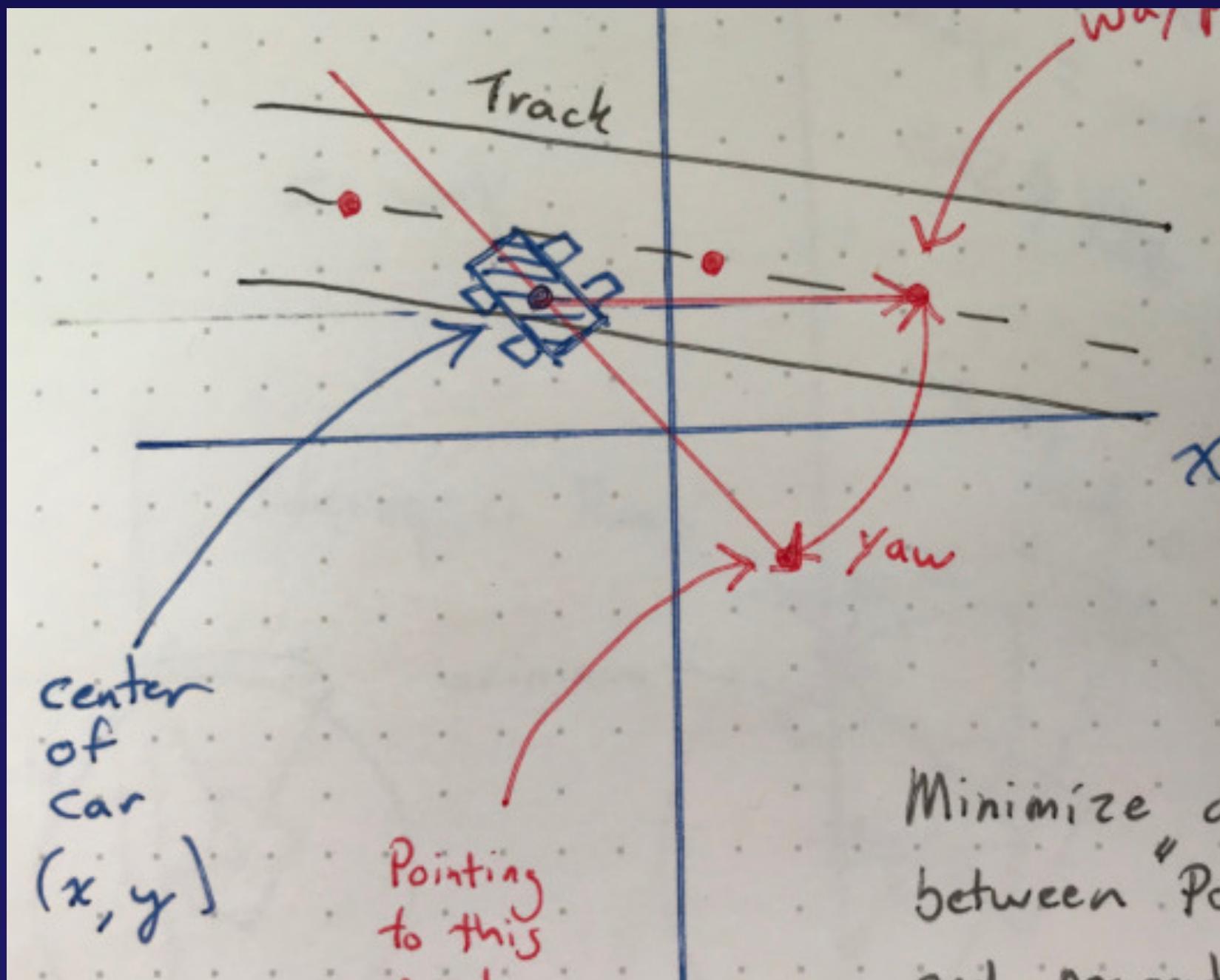
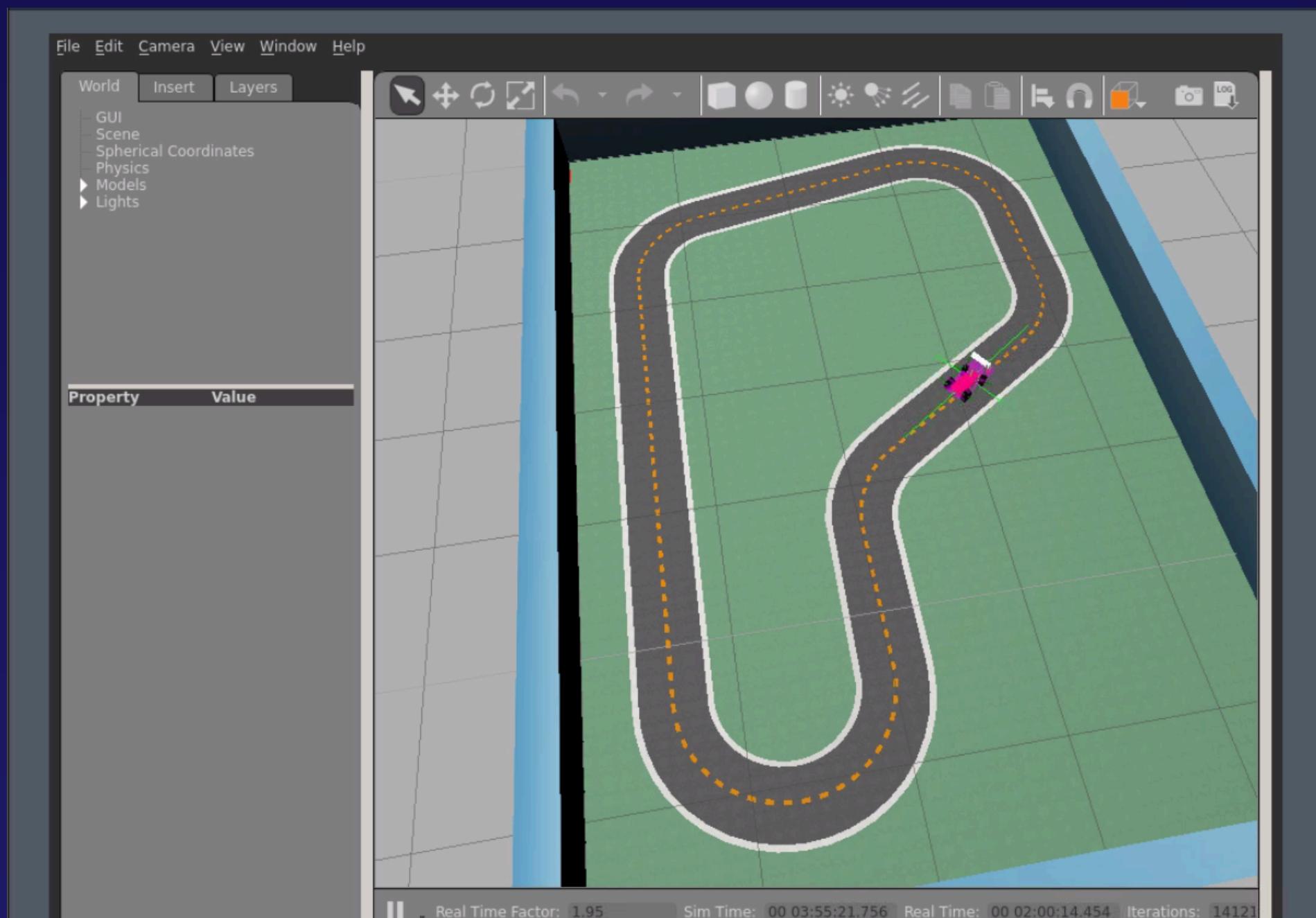


Amazon SageMaker > Training jobs

### Training jobs

Search training jobs

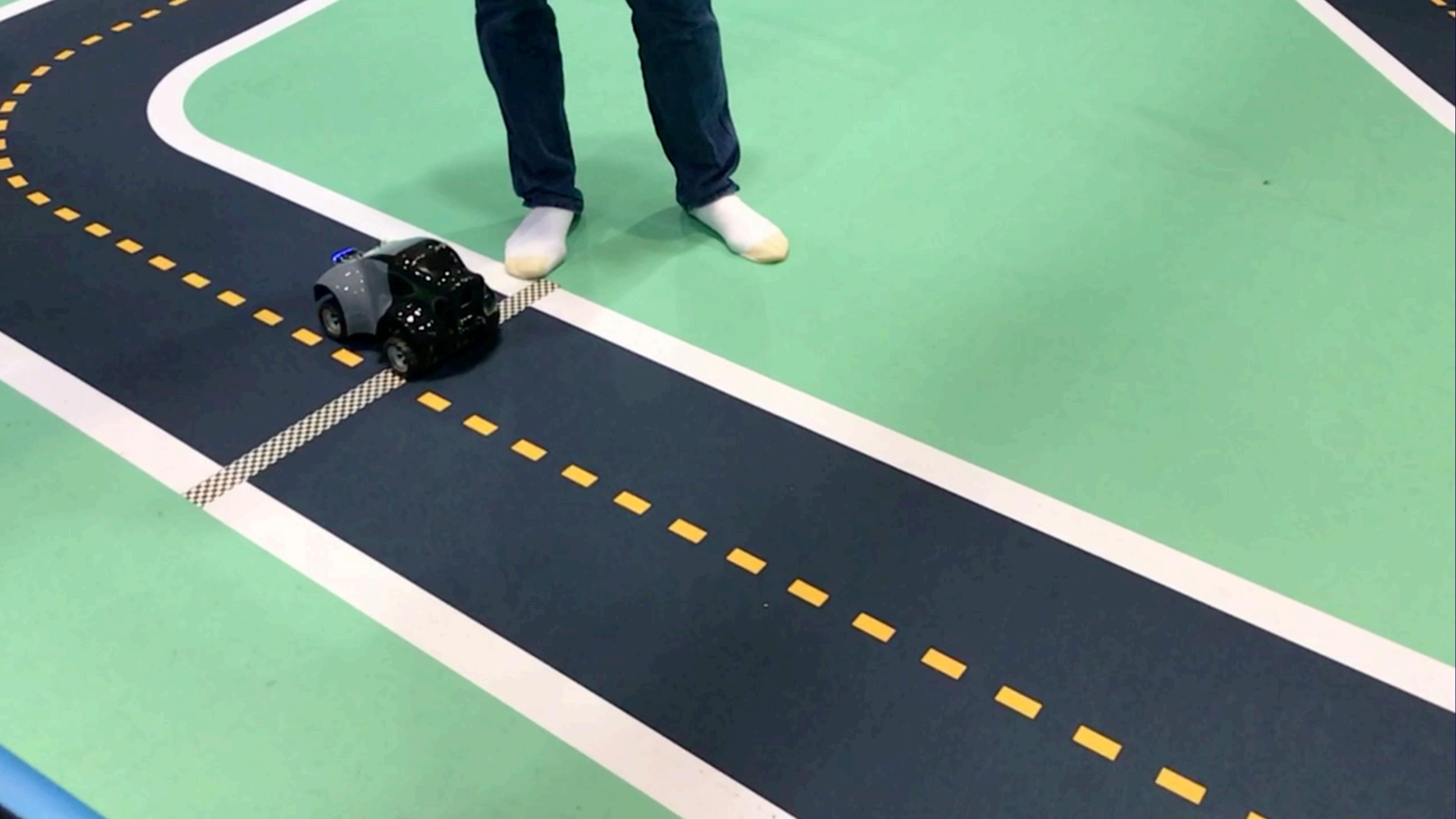
Name	Creation time
rl-deepracer-sagemaker-190413-004838	Apr 13, 2019
rl-deepracer-sagemaker-190413-000754	Apr 13, 2019
rl-deepracer-sagemaker-190407-000559	Apr 07, 2019
rl-deepracer-sagemaker-190406-235509	Apr 06, 2019
rl-deepracer-sagemaker-190405-223857	Apr 05, 2019
rl-deepracer-sagemaker-190325-235850	Mar 25, 2019
rl-deepracer-sagemaker-190325-234010	Mar 25, 2019
rl-deepracer-sagemaker-190324-032006	Mar 24, 2019
rl-deepracer-sagemaker-190324-030723	Mar 24, 2019
rl-deepracer-sagemaker-190324-025907	Mar 24, 2019





# AWS DeepRacer League

## Santa Clara



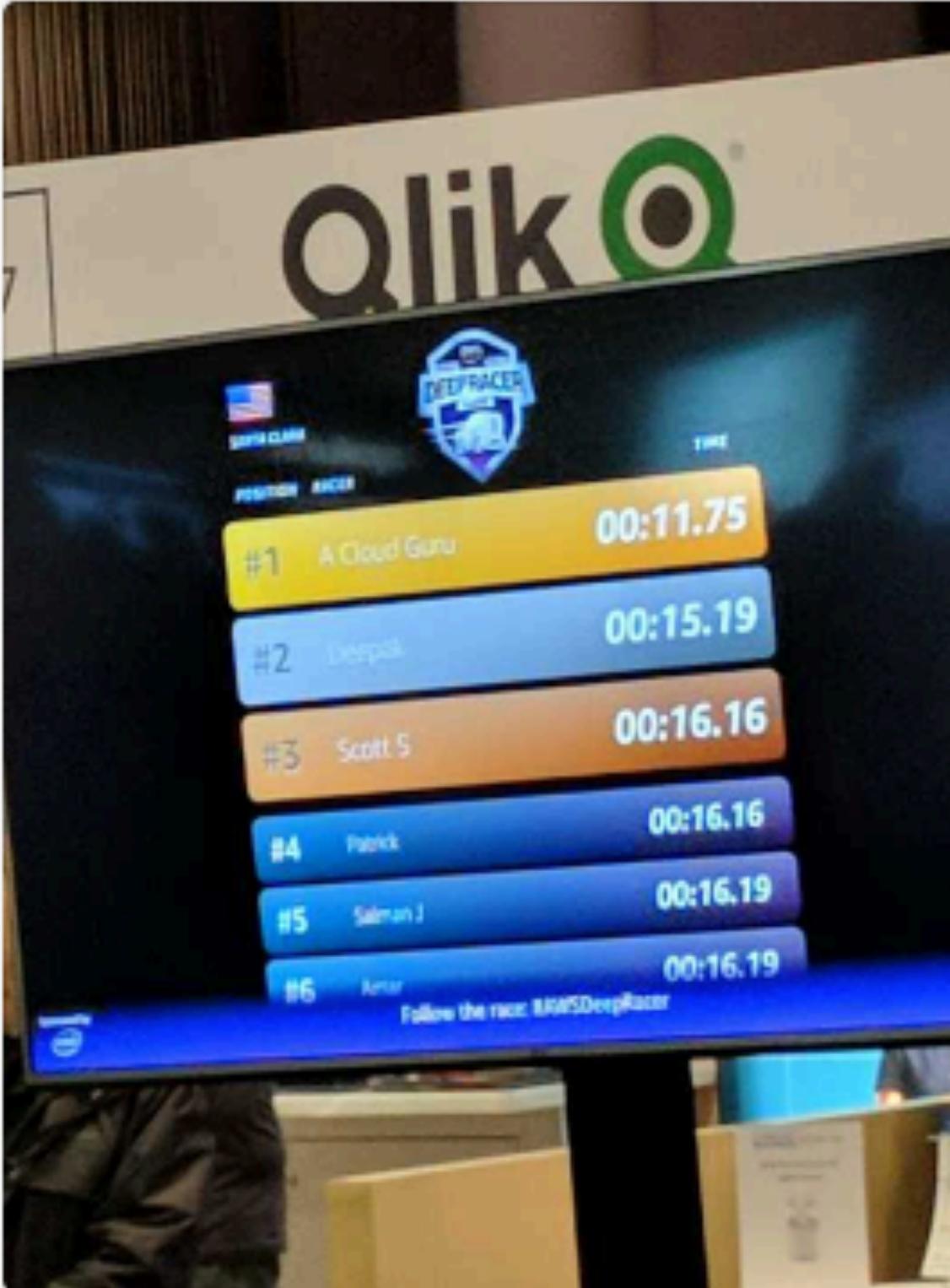




 **Angela Molina** 5:31 PM

Scott just gave us some fantastic news!!!!  
Our DeepRacer is no. 1 🚗

 **Scott Pletcher**  
Yep that just happened



Posted in #official-deepracer | Mar 27th | [View message](#)

12 10 7 3





SANTA CLARA		
POSITION RACER TIME		
#1	A Cloud Guru	00:11.75
#2	JMichaelMotors	00:13.52
#3	Deepak	00:15.19
#4	Scott S	00:16.16
#5	Patrick	00:16.16
#6	Salman J	00:16.19





SANTA CLARA		DEEPRACER LEAGUE	TIME
POSITION	RACER		
#1	Cloud Brigade		<b>00:10.43</b>
#2	Rahul Shah		<b>00:11.10</b>
#3	A Cloud Guru		<b>00:11.75</b>
#4	aamir		<b>00:12.04</b>
#5	Patrick		<b>00:16.16</b>
#6	Salman J		<b>00:16.19</b>





# AWS DeepRacer League - Santa Clara

## Santa Clara - Preliminary Results\*

Search for a racer

POSITION	NAME	TIME	POINTS
1st	Cloud Brigade	00:10.4	989.57
2nd	Rahul Shah	00:11.1	988.9
3rd	Adrian Sarno	00:11.6	988.38
4th	A Cloud Guru	00:11.7	988.25
5th	aamir	00:12.0	987.96
6th	Cprime Time	00:14.3	985.74





# *Scott's Slightly Credible* Tips For DeepRacer Glory™

Please consult physician before DeepRacing. Common side effects of DeepRacer include insomnia, headaches and junk food consumption. Scott is not responsible for obscene AWS bills incurred by hours of training DeepRacer.

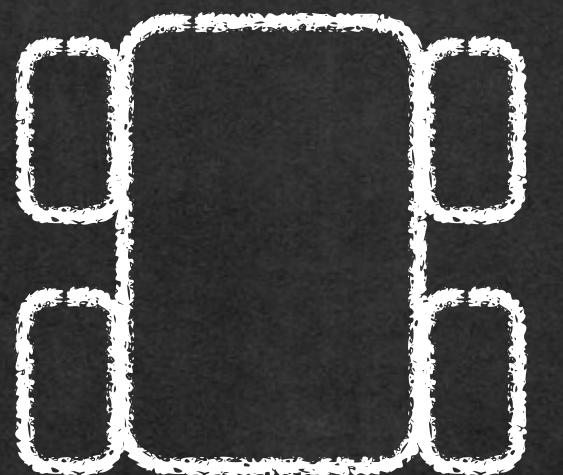


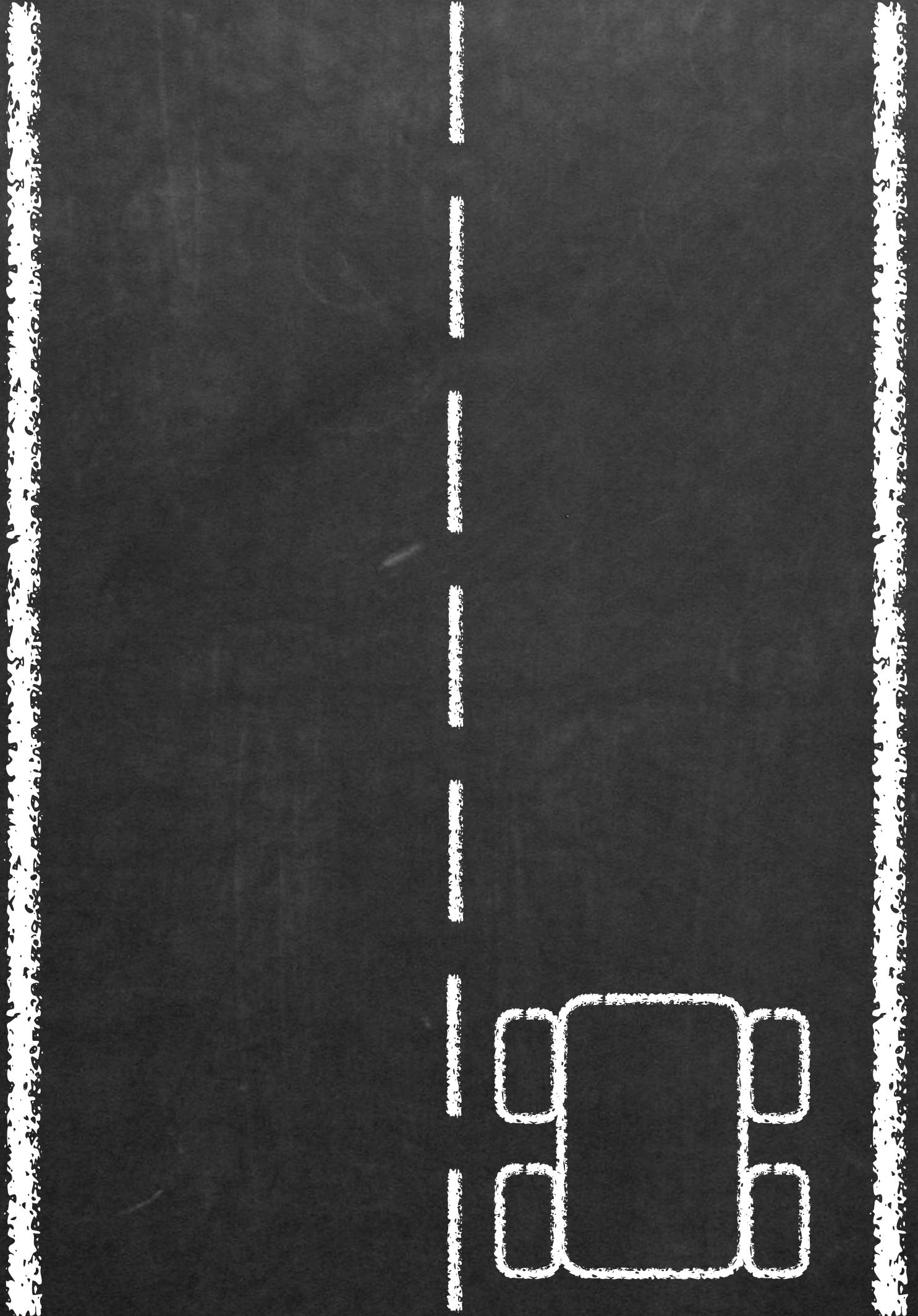
\*for about 4 hours





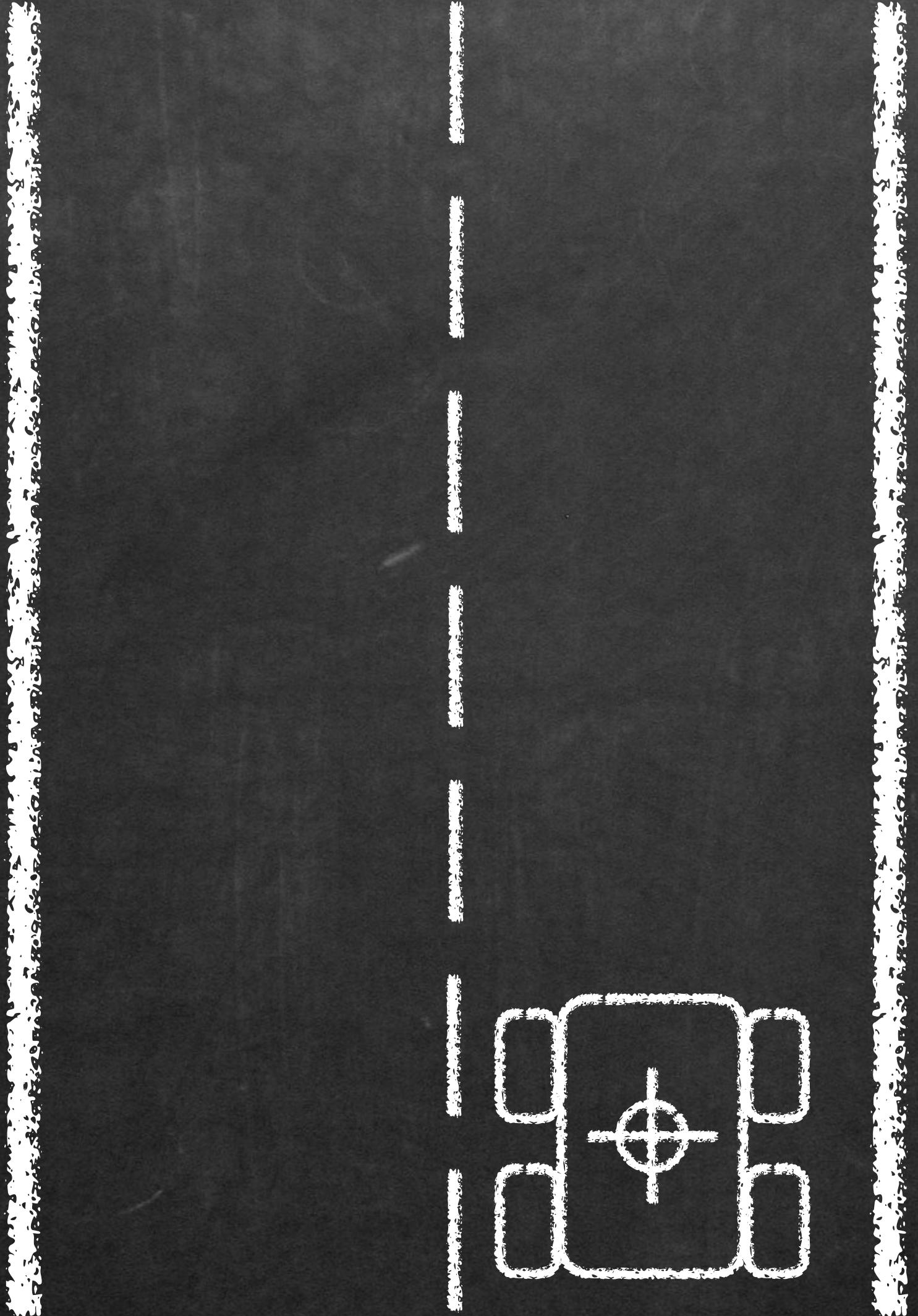
# Markov Decision Process

 AGENT



# Markov Decision Process

- ▶ **AGENT**
- ▶ **ENVIRONMENT**

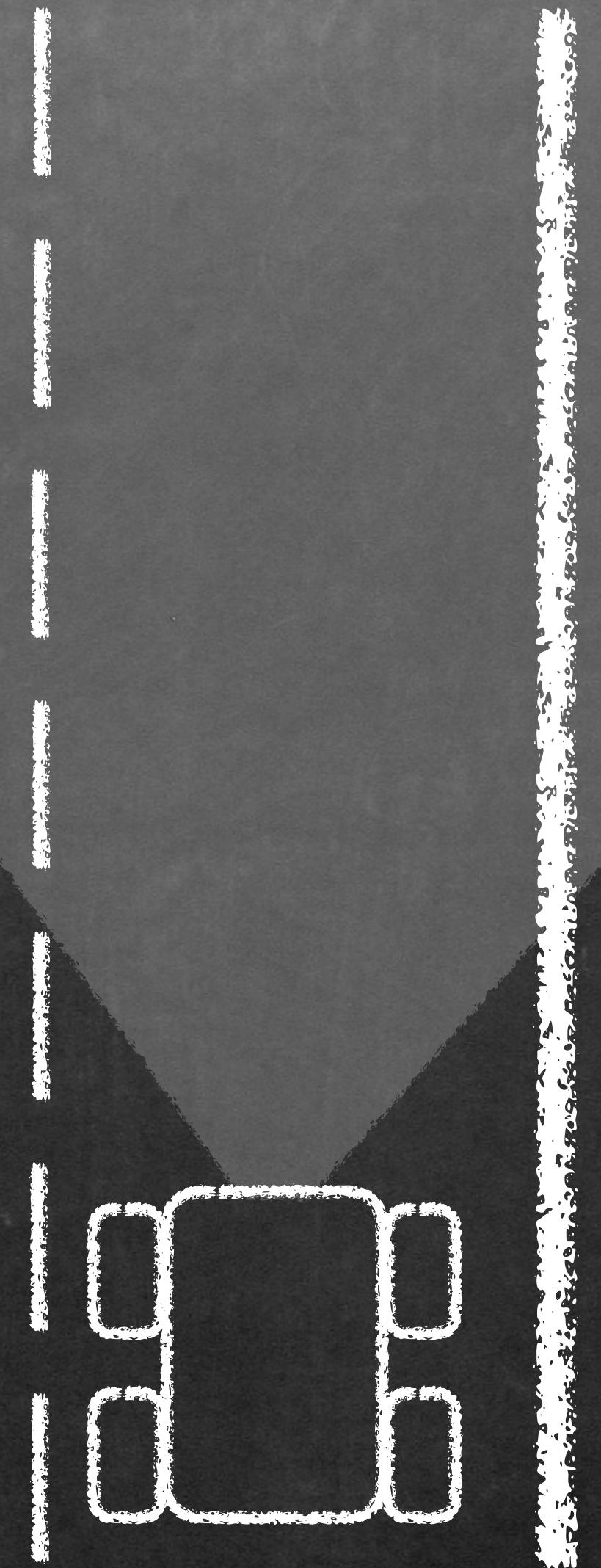


# Markov Decision Process

- ▶ **AGENT**
- ▶ **ENVIRONMENT**
- ▶ **STATE**

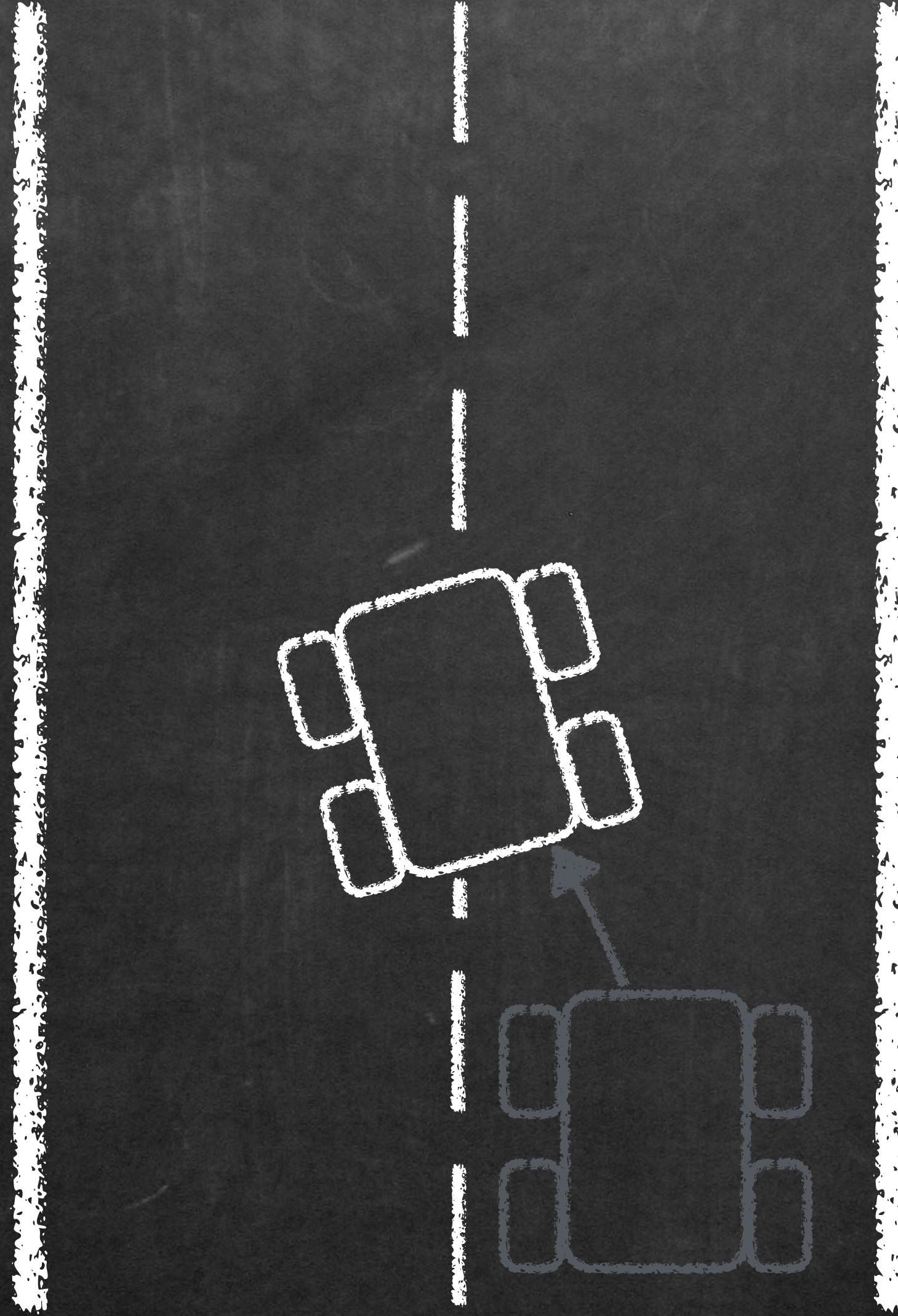


© 2023 A Cloud Guru. All rights reserved. This material may not be published, reproduced, or distributed without the express written consent of A Cloud Guru.



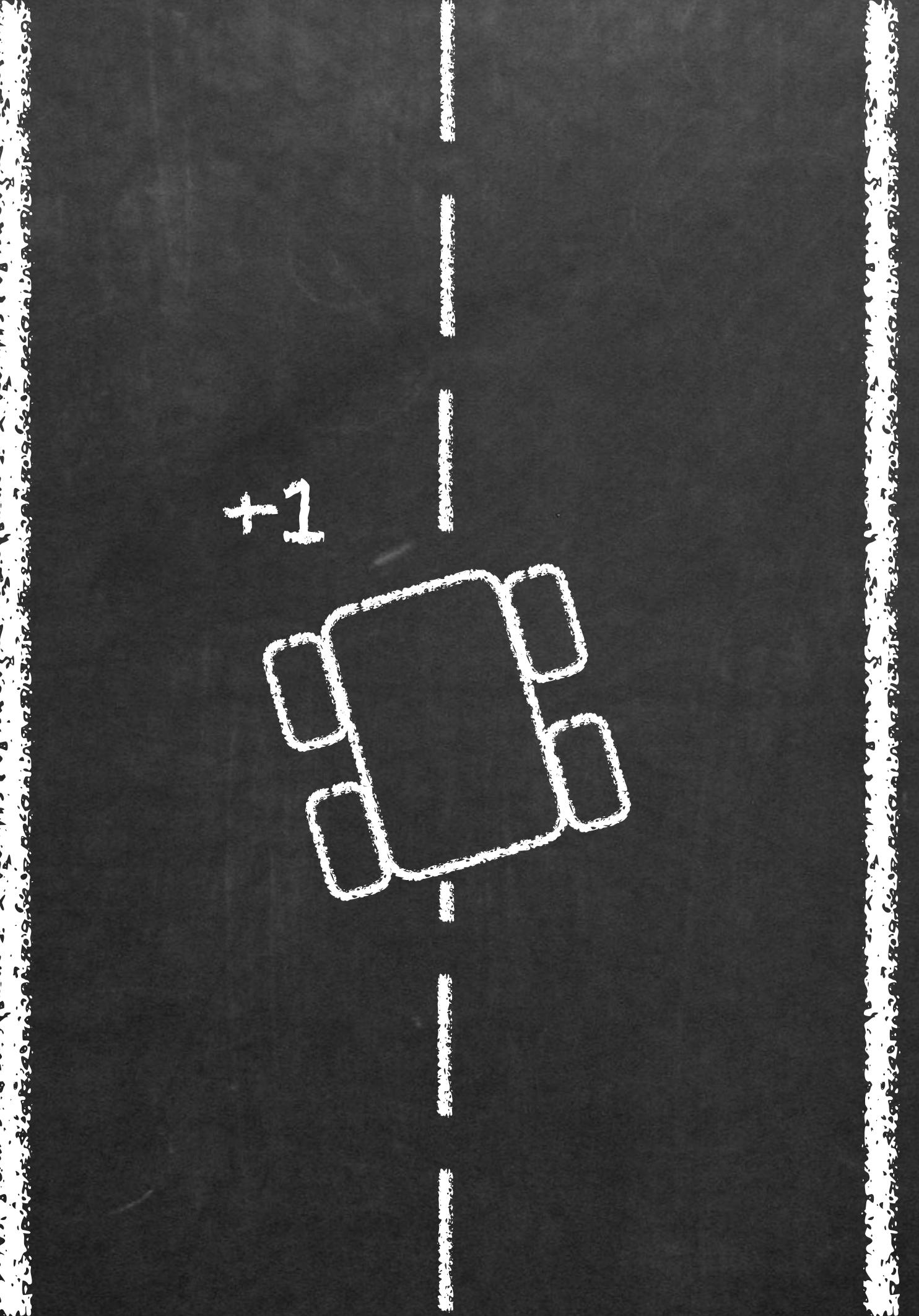
# Markov Decision Process

- ▶ **AGENT**
- ▶ **ENVIRONMENT**
- ▶ **STATE**
- ▶ **OBSERVATION**



# Markov Decision Process

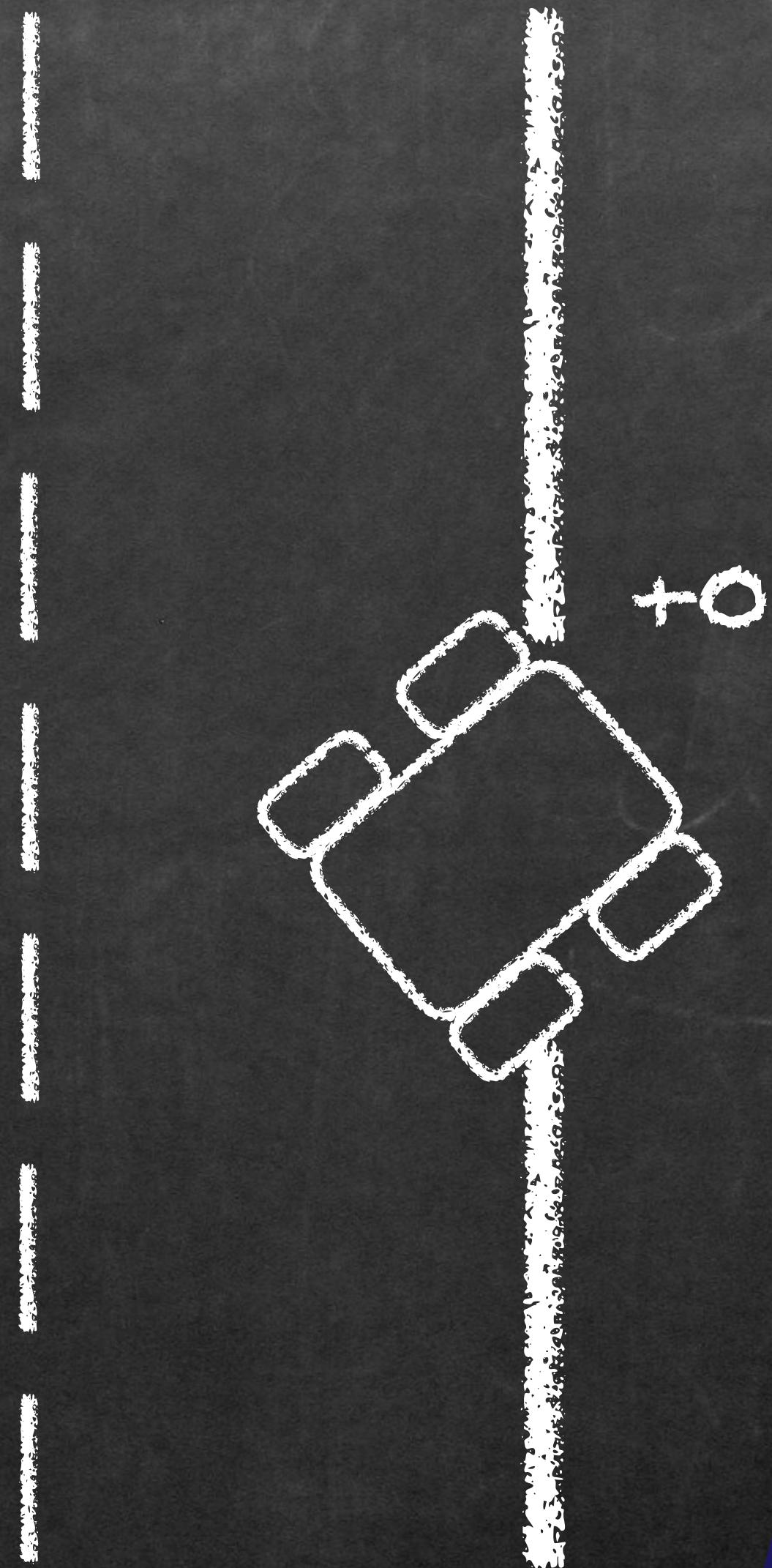
- ▶ **AGENT**
- ▶ **ENVIRONMENT**
- ▶ **STATE**
- ▶ **OBSERVATION**
- ▶ **ACTION**



# Markov Decision Process

- ▶ AGENT
- ▶ ENVIRONMENT
- ▶ STATE
- ▶ OBSERVATION
- ▶ ACTION
- ▶ REWARD

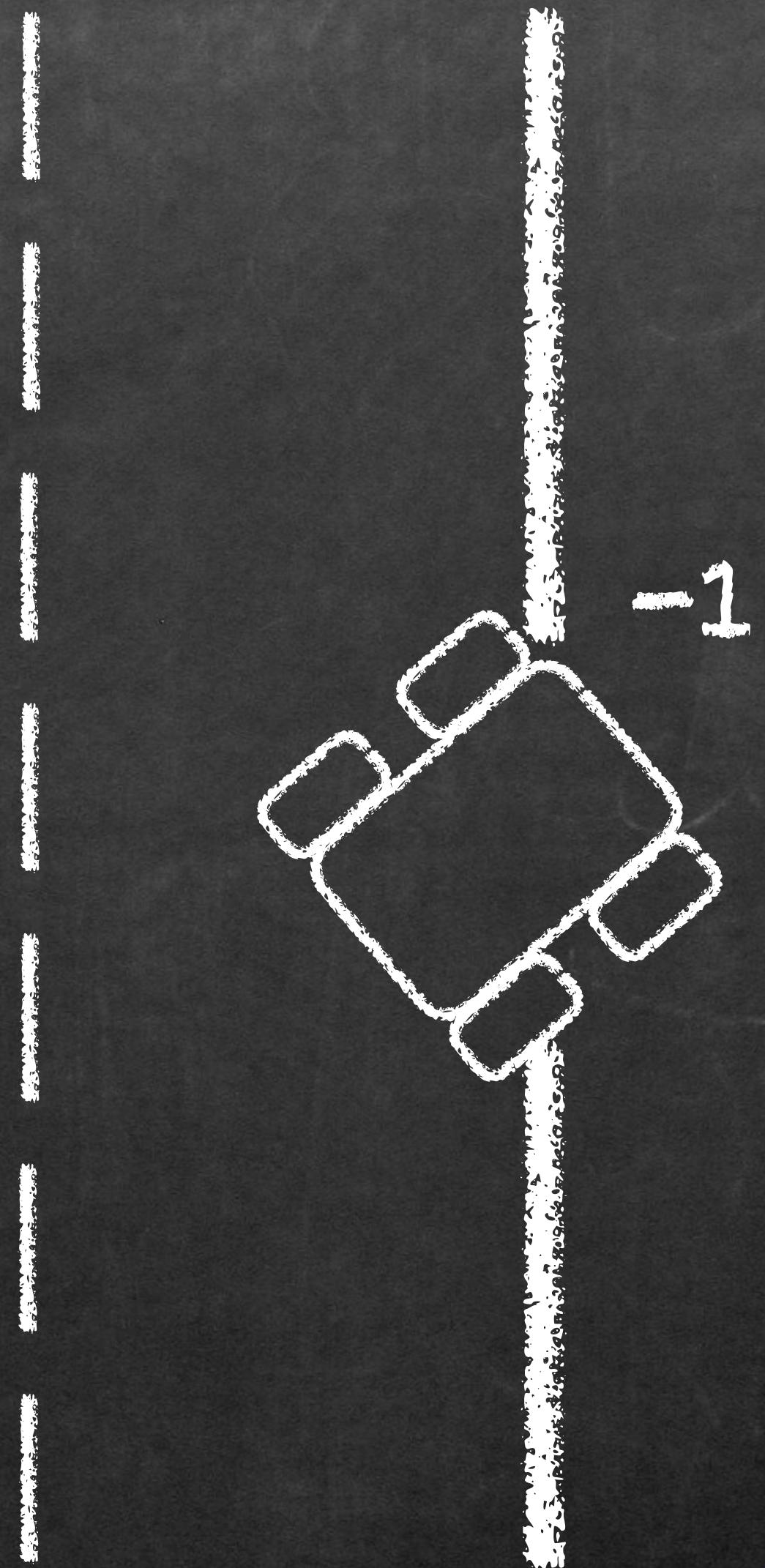
DATA SCIENCE FOR CLOUD COMPUTING



# Markov Decision Process

- ▶ AGENT
- ▶ ENVIRONMENT
- ▶ STATE
- ▶ OBSERVATION
- ▶ ACTION
- ▶ REWARD

DATA SCIENCE FOR CLOUD COMPUTING

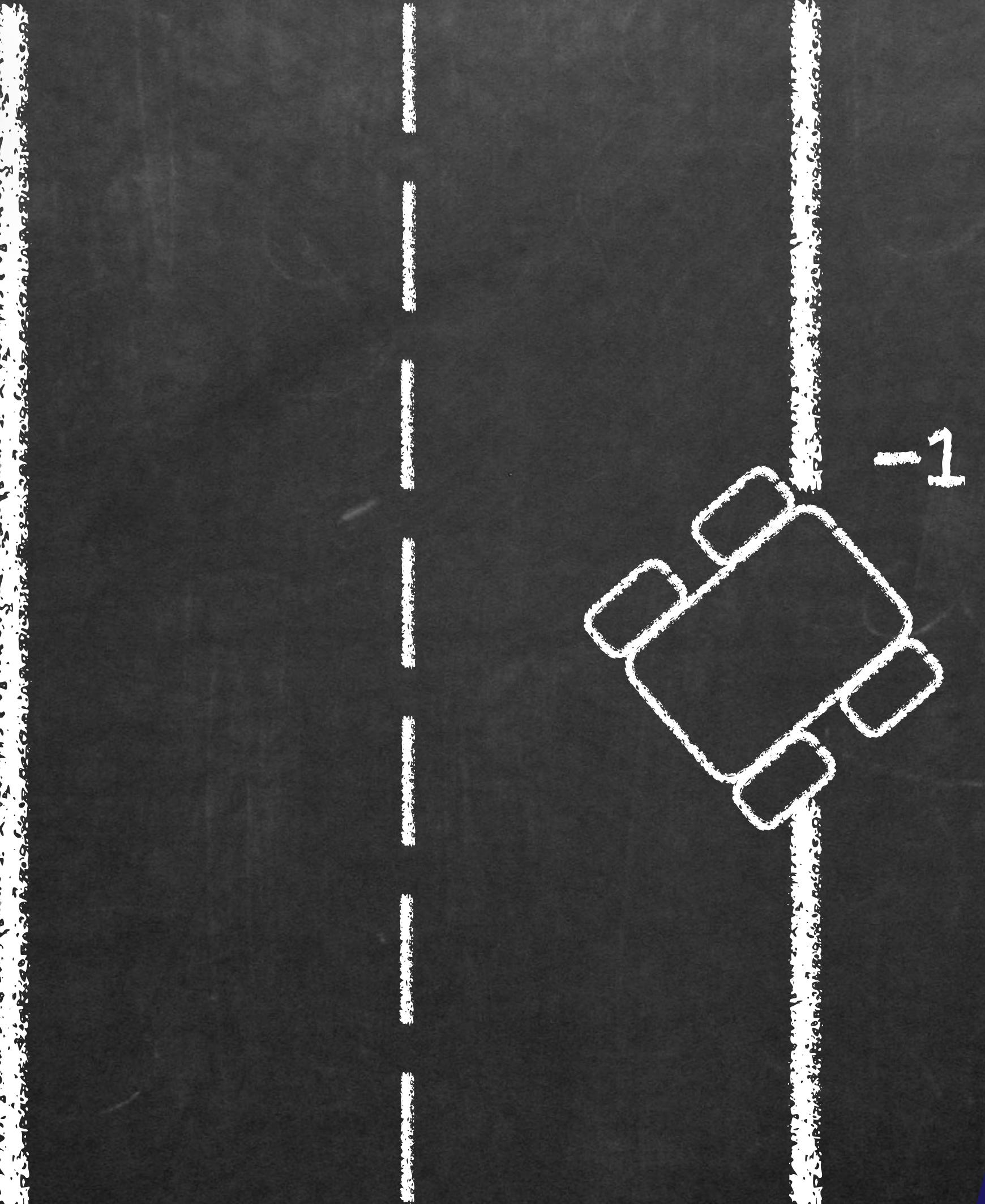


# Markov Decision Process

- ▶ AGENT
- ▶ ENVIRONMENT
- ▶ STATE
- ▶ OBSERVATION
- ▶ ACTION
- ▶ REWARD

Total Reward

+1
+1
+0
+2
-1
+0
<hr/>
+3



## Markov Decision Process

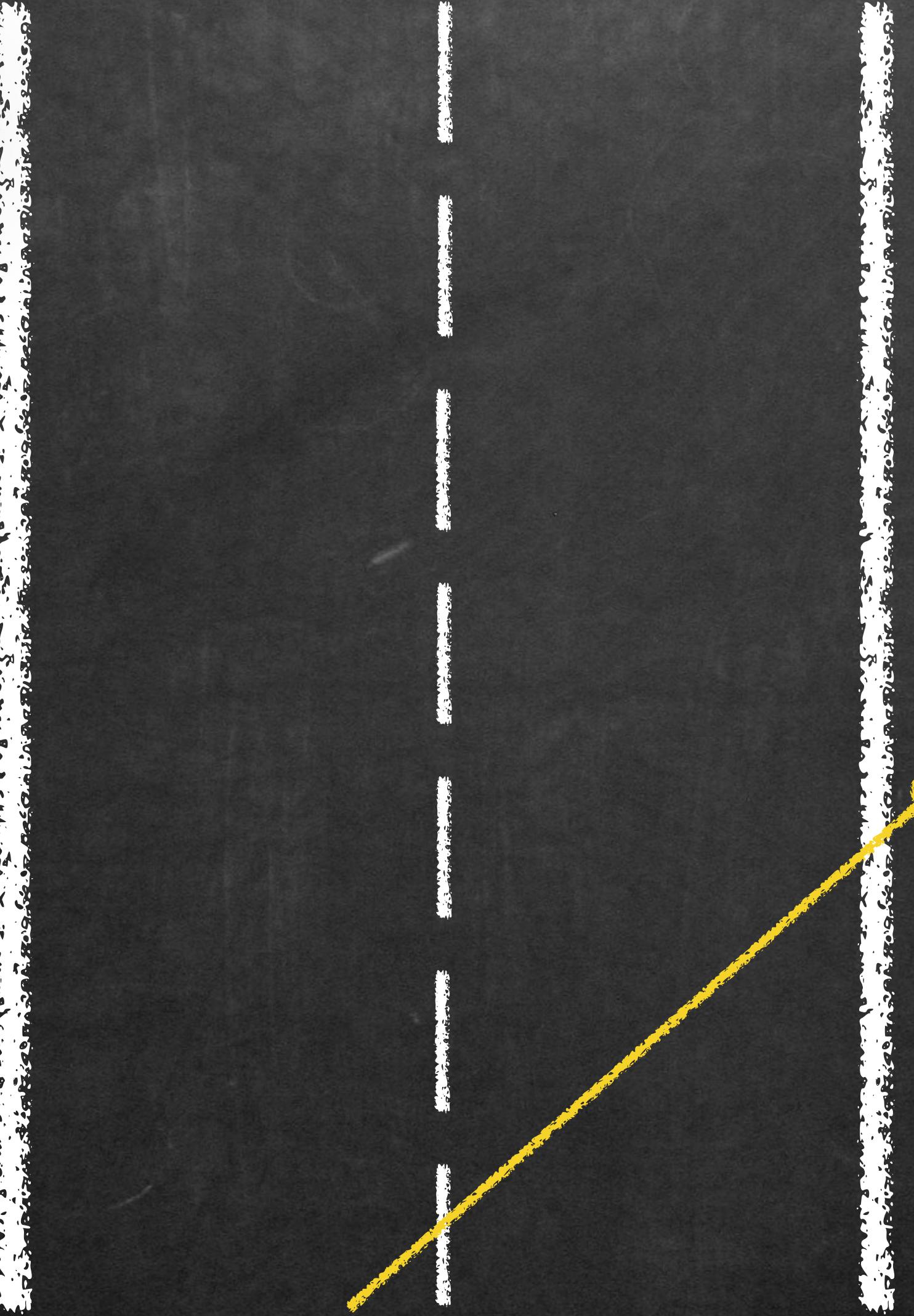
- ▶ AGENT
- ▶ ENVIRONMENT
- ▶ STATE
- ▶ OBSERVATION
- ▶ ACTION
- ▶ REWARD

Total  
Reward

+0

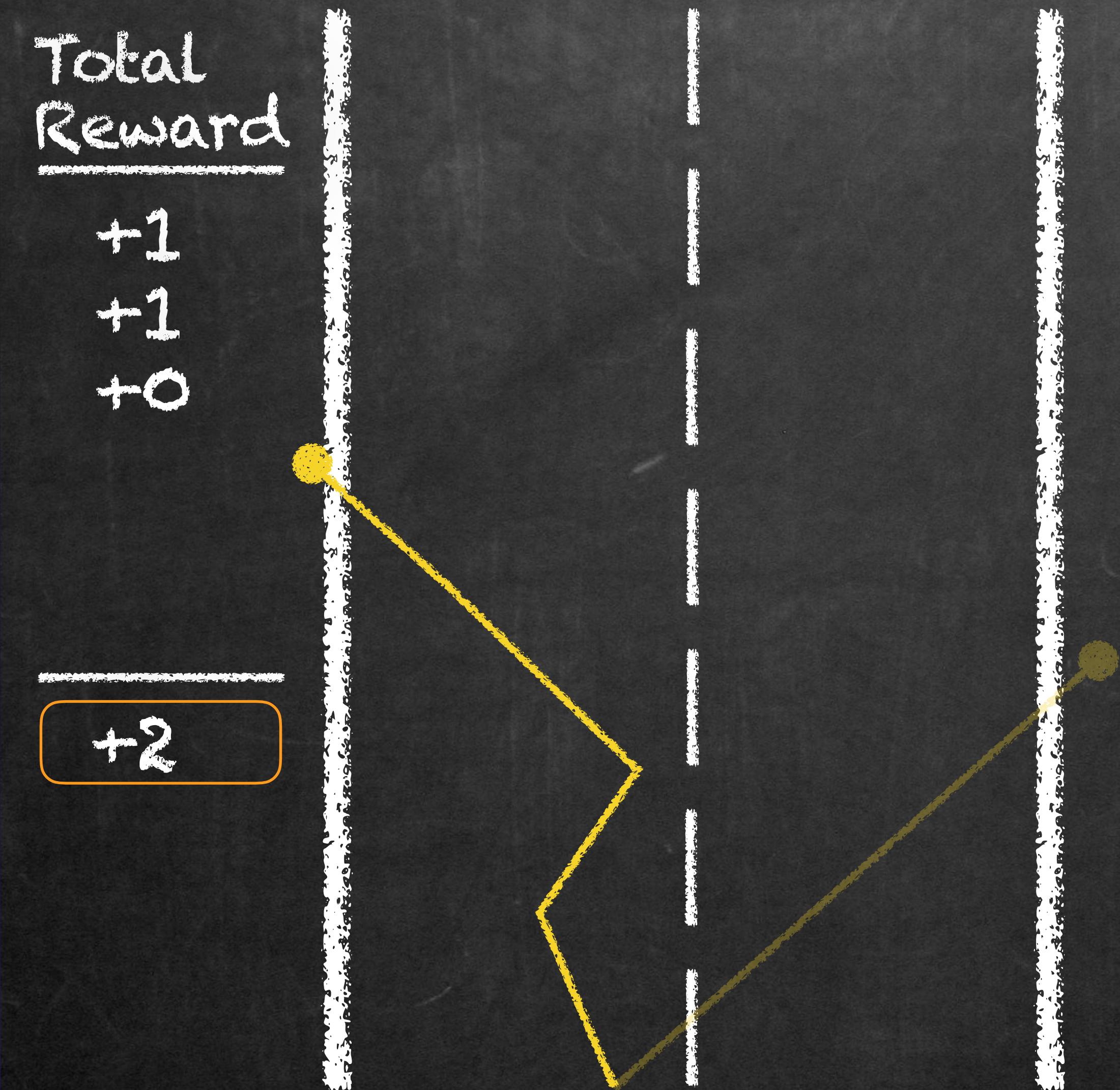
Total  
Reward

+0



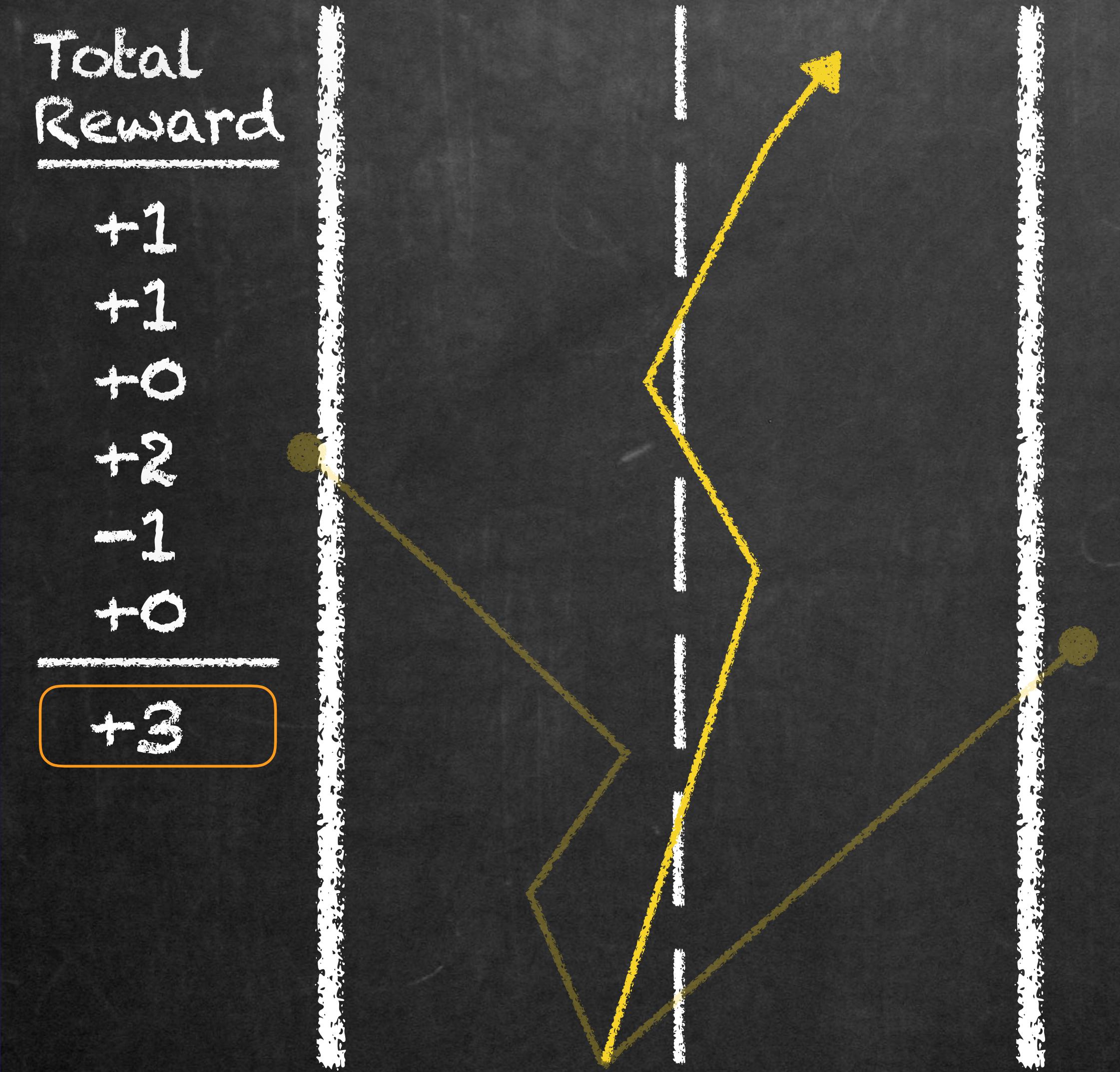
## Markov Decision Process

- ▶ AGENT
- ▶ ENVIRONMENT
- ▶ STATE
- ▶ OBSERVATION
- ▶ ACTION
- ▶ REWARD
- ▶ EPISODES



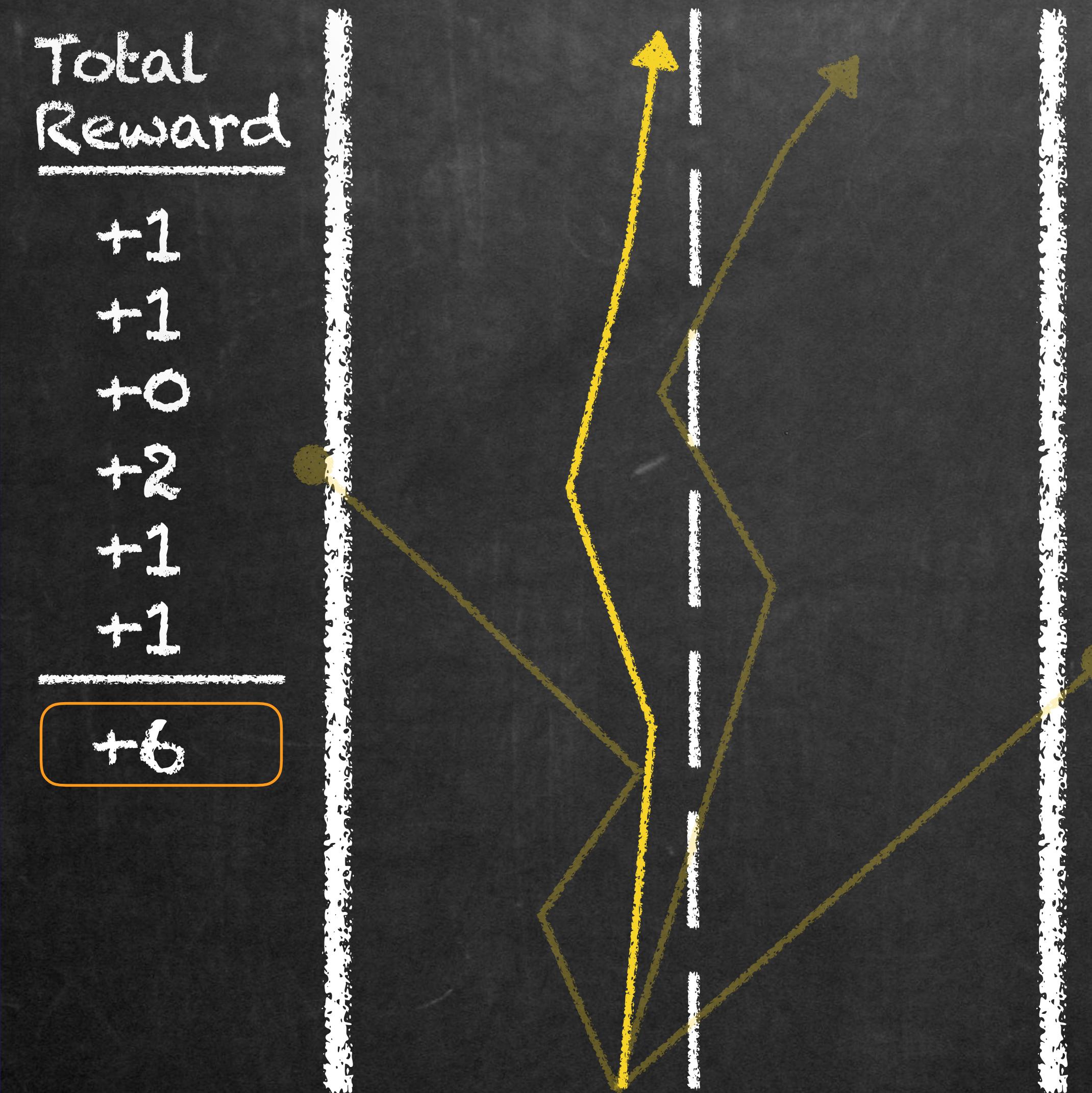
## Markov Decision Process

- ▶ AGENT
- ▶ ENVIRONMENT
- ▶ STATE
- ▶ OBSERVATION
- ▶ ACTION
- ▶ REWARD
- ▶ EPISODES



## Markov Decision Process

- ▶ AGENT
- ▶ ENVIRONMENT
- ▶ STATE
- ▶ OBSERVATION
- ▶ ACTION
- ▶ REWARD
- ▶ EPISODES



## Markov Decision Process

- ▶ AGENT
- ▶ ENVIRONMENT
- ▶ STATE
- ▶ OBSERVATION
- ▶ ACTION
- ▶ REWARD
- ▶ EPISODES



## Reward graph

2019-04-30 (13:07:00) - 2019-04-30 (14:07:00) ▾



Total reward over time

5.00k

4.00k

3.00k

2.00k

1.00k

0

17:10 17:15 17:20 17:25 17:30 17:35 17:40 17:45 17:50 17:55 18:00 18:05





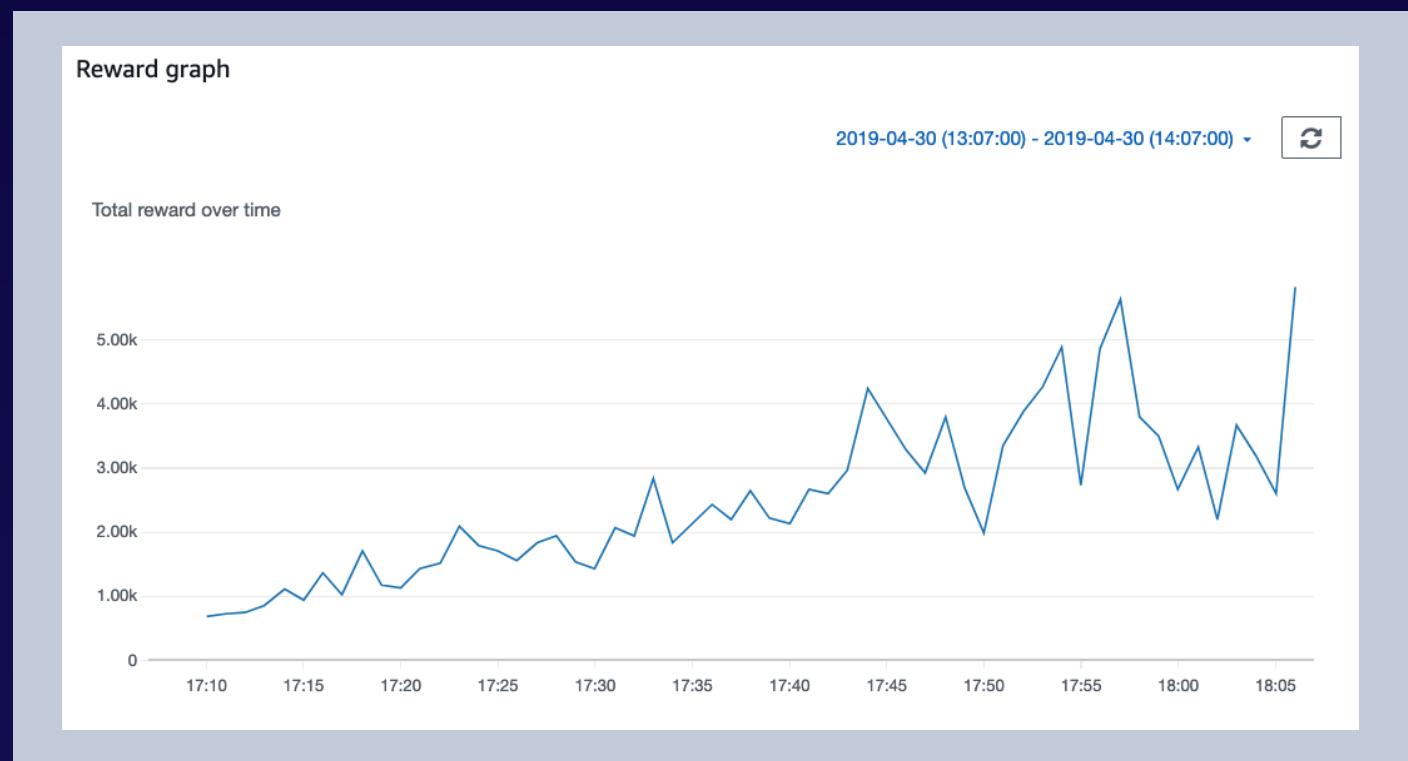
## Reward graph

2019-04-30 (13:07:00) - 2019-04-30 (14:07:00) ▾



Total reward over time





# *Scott's Slightly Credible* Tips For DeepRacer Glory™



Dare to be Different



Copy.

Paste.

Race.



Copy.  
Paste.  
Race.



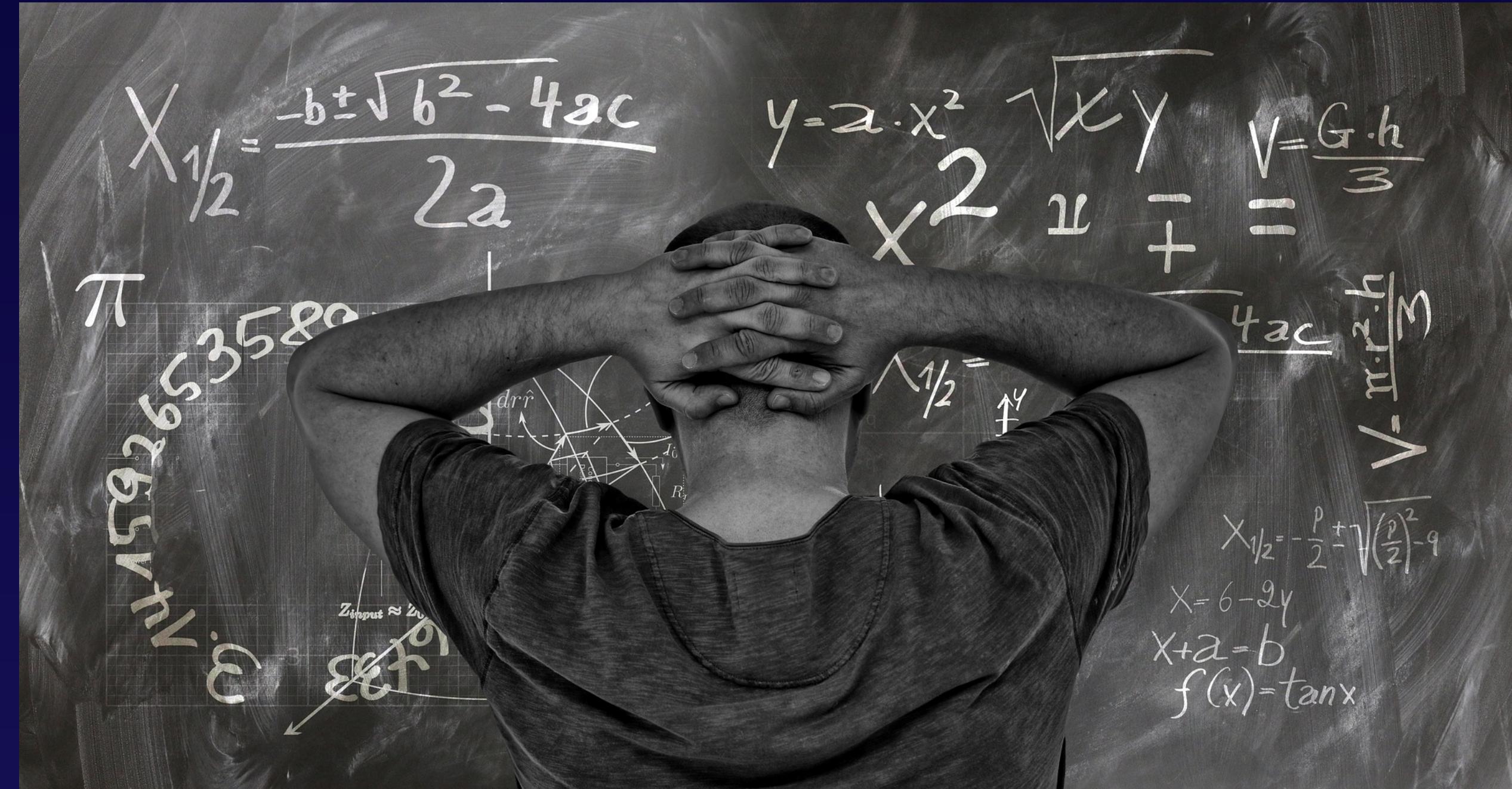


Copy.  
Paste.  
Race.

## Santa Clara Summit - Preliminary Results\*

POSITION	NAME	COUNTRY	TIME	POINTS
1st	CLOUD BRIGADE	US	00:10.4	989.57
2nd	RAHUL SHAH	US	00:11.1	988.9
3rd	ADRIAN SARNO	US	00:11.6	988.38
4th	A CLOUD GURU	US	00:11.7	988.25
5th	AAMIR	US	00:12.0	987.96
6th	CPRIME TIME	US	00:14.3	985.74
7th	DEEPAK	US	00:15.2	984.81
8th	NORAEK	US	00:15.5	984.54
9th	PRAVEEN KUMAR	US	00:15.7	984.29
10th	PATRICK	US	00:16.2	983.84

# DO SOMETHING DIFFERENT

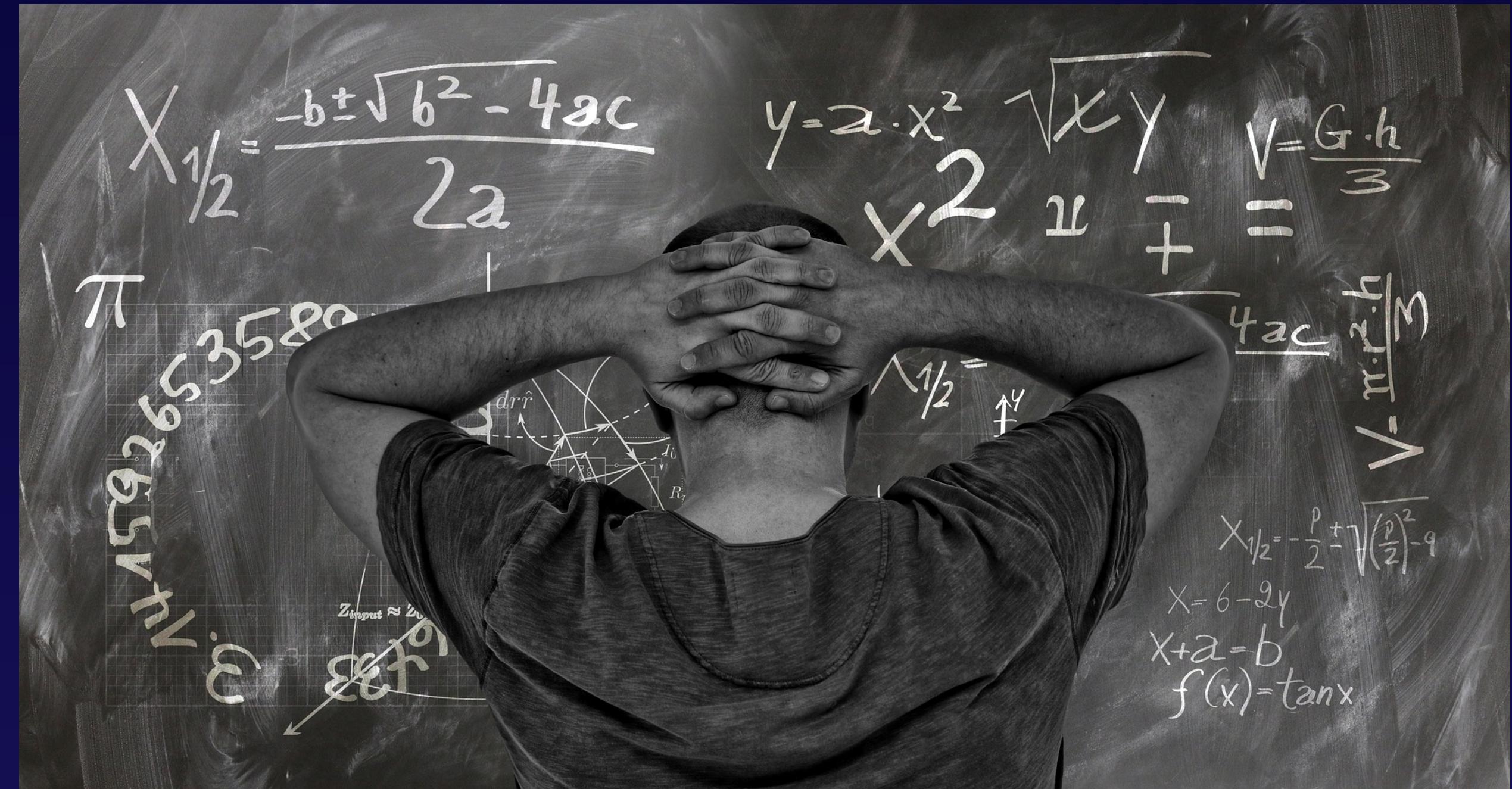


```
3 require File.expand_path('../..', __FILE__)
4 # Prevent database truncation if the migration fails
5 abort("The Rails environment is reading configuration from database")
6 require 'spec_helper'
7 require 'rspec/rails'

8 require 'capybara/rspec'
9 require 'capybara/rails'
10
11 Capybara.javascript_driver = :webkit
12 Category.delete_all; Category.create!
13 Shoulda::Matchers.configure do |config|
14   config.integrate do |with|
15     with.test_framework :rspec
16     with.library :rails
17   end
18 end
19
20 # Add additional requires below this line
21
22 # Requires supporting files within the same directory as this file if you want
23 # spec/support/ and its subdirectories to be loaded by your application
24 # run as spec files by default. This means that files in spec/support
25 # will be run with the same environment as your test, which is good for
26 # run twice. It is recommended that you do not name this file spec.rb,
27 # end with .spec.rb. You can configure this using the RSpec.configure
28 # option or the --tag command on the command line.
29
30 # option or the --tag command on the command line.
```



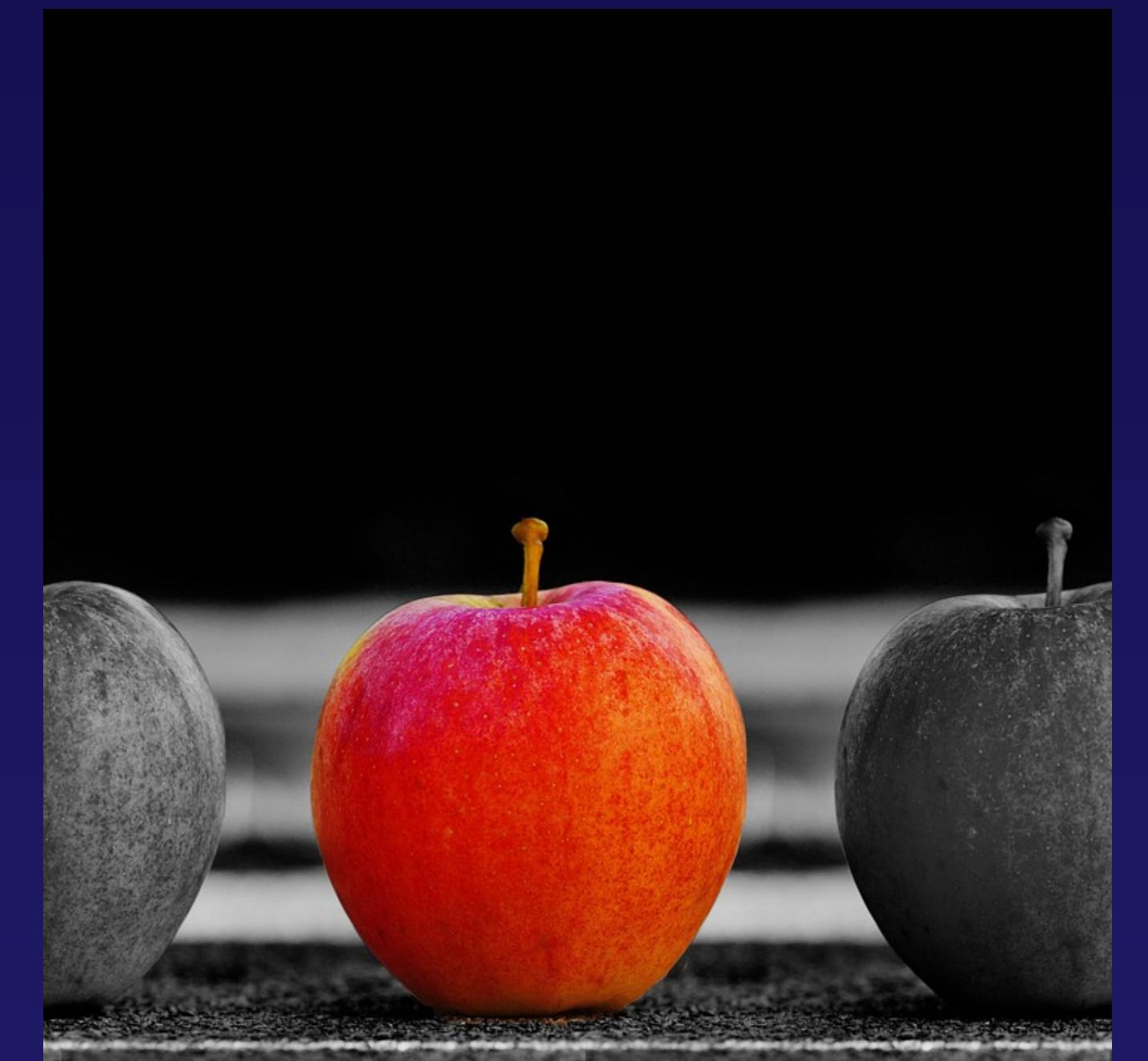
# DO SOMETHING DIFFERENT

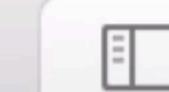


```
3 require File.expand_path('../..', __FILE__)
4 # Prevent database truncation if the migration fails
5 abort("The Rails environment is reading configuration from database.yml, but it's
6 require 'spec_helper'
7 require 'rspec/rails'

8 require 'capybara/rspec'
9 require 'capybara/rails'

10 Capybara.javascript_driver = :webkit
11 Category.delete_all; Category.create!
12 Shoulda::Matchers.configure do |config|
13   config.integrate do |with|
14     with.test_framework :rspec
15     with.library :rails
16   end
17 end
18 end
19
20 # Add additional requires below this line
21
22 # Requires supporting files within the same directory as this file if you don't
23 # want to change the search path for these files.
24 # spec/support/ and its subdirectories
25 # run as spec files by default. This means you can run 'rake spec' to
26 # in _spec.rb will both be run.
27 # run twice. It is recommended that you do not name this file
28 # end with _spec.rb. You can configure this using
29 # option or the --tag option on the command line. If you
30 # option is given, it will be passed to the buffer
```





google.com



# Google



Google Search

I'm Feeling Lucky

The above is an illustration for educational purposes.

Google™ is a trademark of Google, Inc. LMGTFY is not associated with Google in any way.



## Implementation of the Pure Pursuit Path Tracking Algorithm

R. Craig Coulter  
CMU-RI-TR-92-01

The Robotics Institute  
Carnegie Mellon University  
Pittsburgh, Pennsylvania 15213

January 1992

© 1990 Carnegie Mellon

```
rabbit = [0, 0]
pointing = [0, 0]

# Reward when yaw (car_orientation) is pointed to the next waypoint IN
FRONT.

# Find nearest waypoint coordinates in front of car
rabbit = waypoints[closest_waypoints+1]

radius = math.sqrt((x - rabbit[0])**2 + (y - rabbit[1])**2)

pointing[0] = x + (radius * math.cos(car_orientation))
pointing[1] = y + (radius * math.sin(car_orientation))

vector_delta = math.hypot(pointing[0] - rabbit[0], pointing[1] -
rabbit[1])

# Max distance for pointing away will be the radius * 2
# Min distance means we are pointing directly at the next waypoint
# We can setup a reward that is a ratio to this max.

if vector_delta == 0:
    reward = 1
else:
    reward = ( 1 - vector_delta / (radius * 2))
```



```
rabbit = [0,0]
pointing = [0,0]

# Reward when yaw (car_orientation) is pointed to the next waypoint IN
FRONT.

# Find nearest waypoint coordinates in front of car
rabbit = waypoints[closest_waypoints+1]

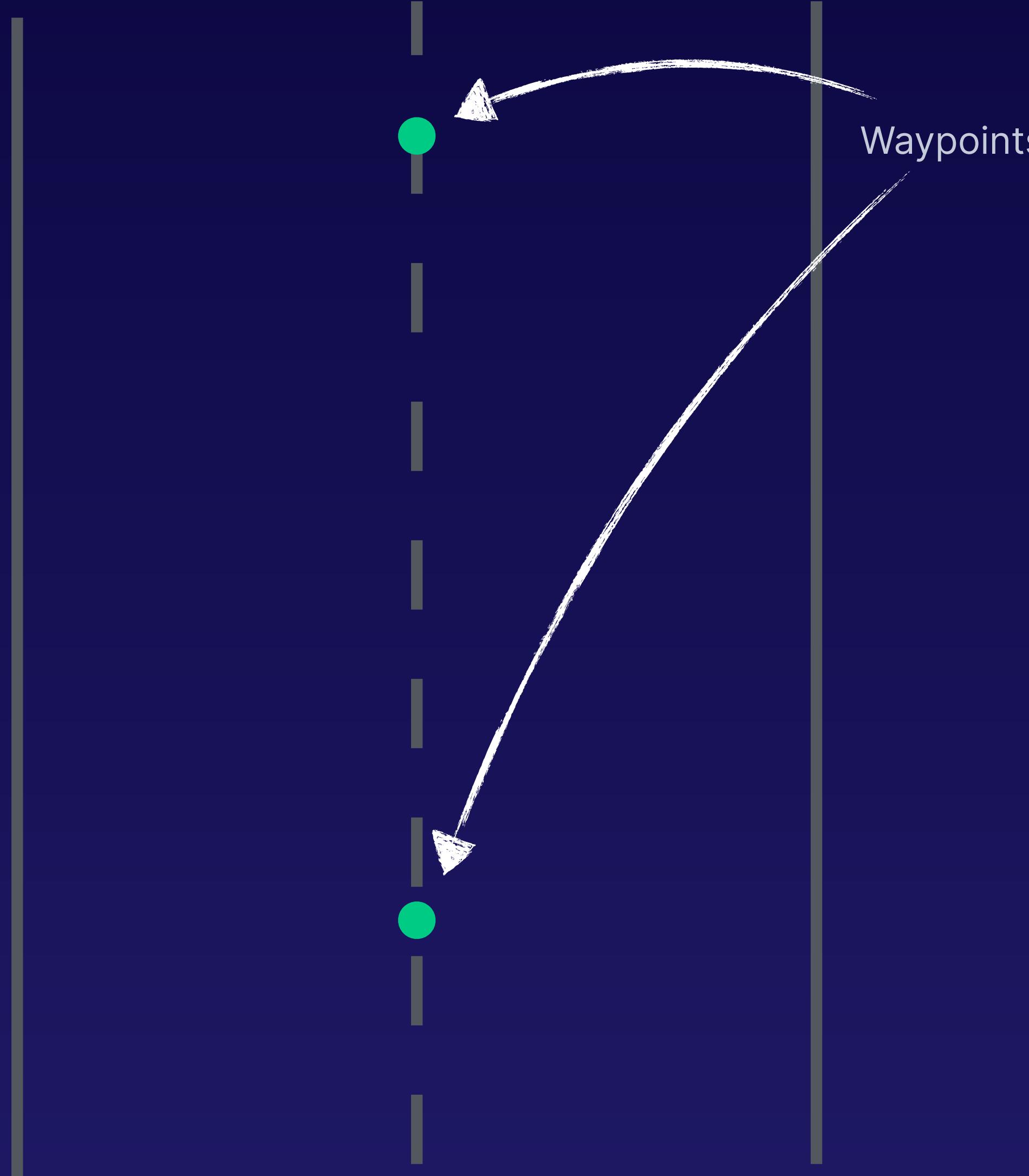
radius = math.sqrt((x - rabbit[0])**2 + (y - rabbit[1])**2)

pointing[0] = x + (radius * math.cos(car_orientation))
pointing[1] = y + (radius * math.sin(car_orientation))

vector_delta = math.hypot(pointing[0] - rabbit[0], pointing[1] -
rabbit[1])

# Max distance for pointing away will be the radius * 2
# Min distance means we are pointing directly at the next waypoint
# We can setup a reward that is a ratio to this max.

if vector_delta == 0:
    reward = 1
else:
    reward = ( 1 - vector_delta / (radius * 2))
```



```
rabbit = [0,0]
pointing = [0,0]

# Reward when yaw (car_orientation) is pointed to the next waypoint IN
FRONT.

# Find nearest waypoint coordinates in front of car
rabbit = waypoints[closest_waypoints+1]

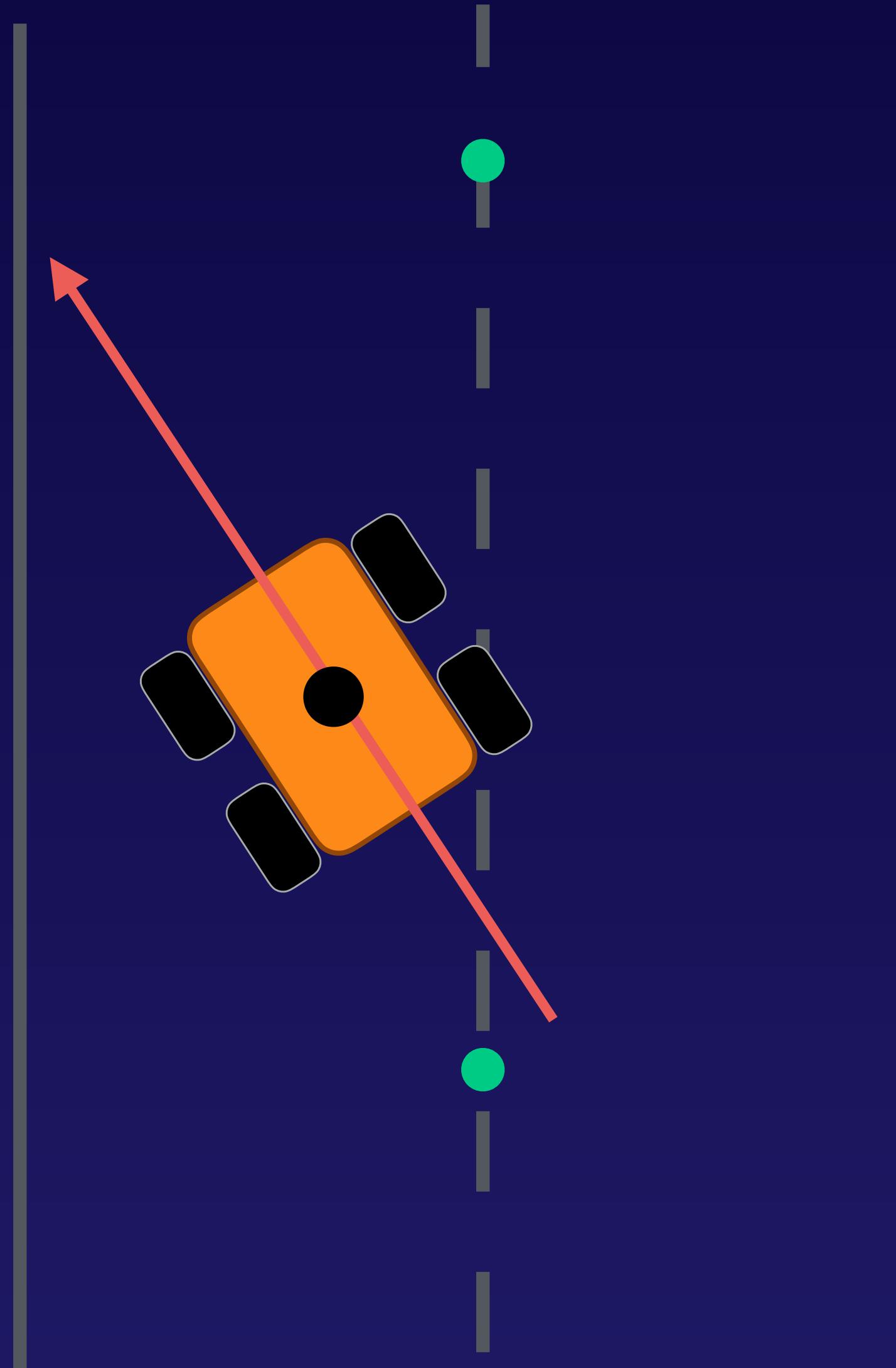
radius = math.sqrt((x - rabbit[0])**2 + (y - rabbit[1])**2)

pointing[0] = x + (radius * math.cos(car_orientation))
pointing[1] = y + (radius * math.sin(car_orientation))

vector_delta = math.hypot(pointing[0] - rabbit[0], pointing[1] -
rabbit[1])

# Max distance for pointing away will be the radius * 2
# Min distance means we are pointing directly at the next waypoint
# We can setup a reward that is a ratio to this max.

if vector_delta == 0:
    reward = 1
else:
    reward = ( 1 - vector_delta / (radius * 2))
```



```
rabbit = [0,0]
pointing = [0,0]

# Reward when yaw (car_orientation) is pointed to the next waypoint IN FRONT.

# Find nearest waypoint coordinates in front of car
rabbit = waypoints[closest_waypoints+1]

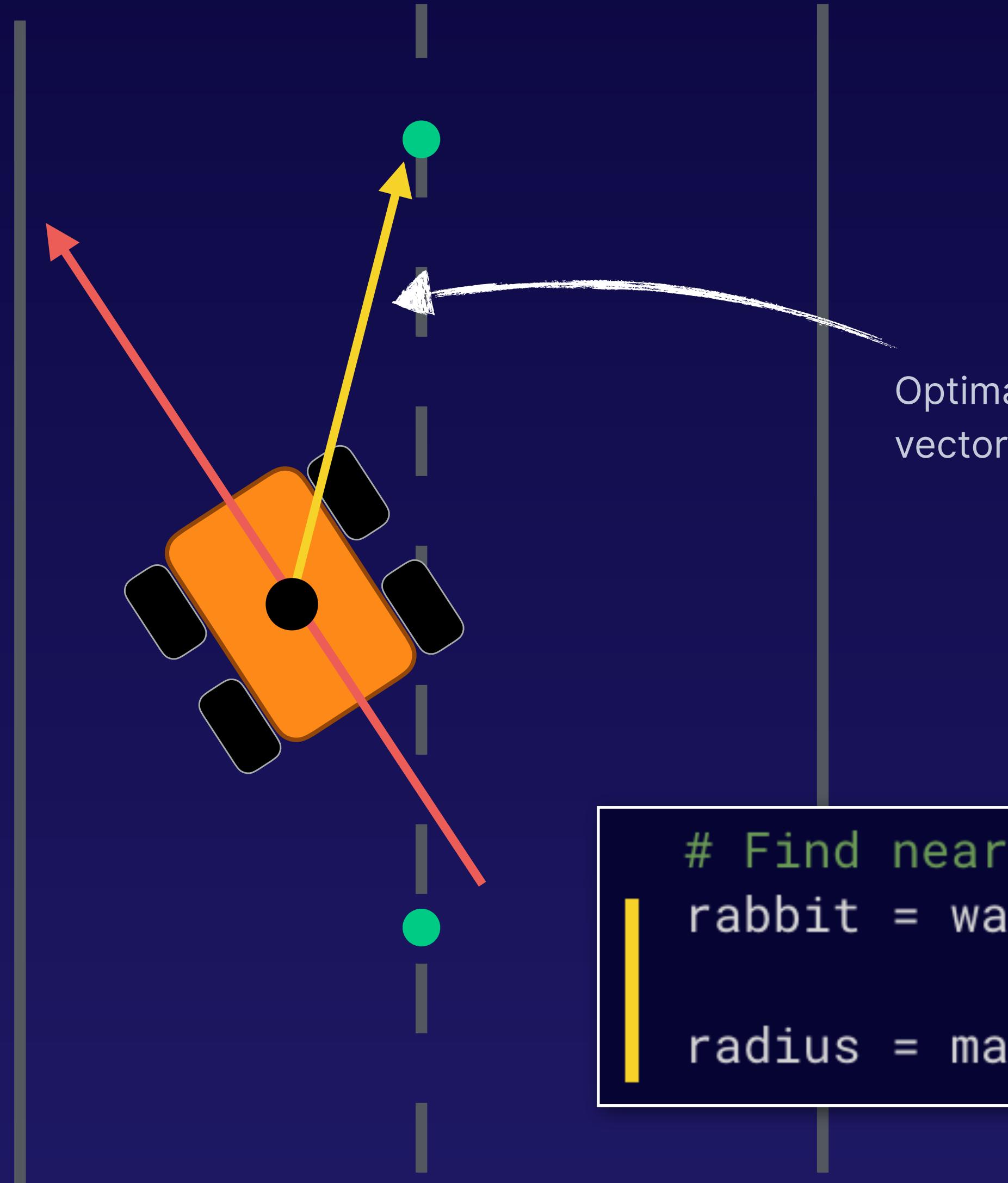
radius = math.sqrt((x - rabbit[0])**2 + (y - rabbit[1])**2 )

pointing[0] = x + (radius * math.cos(car_orientation))
pointing[1] = y + (radius * math.sin(car_orientation))

vector_delta = math.hypot(pointing[0] - rabbit[0], pointing[1] - rabbit[1])

# Max distance for pointing away will be the radius * 2
# Min distance means we are pointing directly at the next waypoint
# We can setup a reward that is a ratio to this max.

if vector_delta == 0:
    reward = 1
else:
    reward = ( 1 - vector_delta / (radius * 2))
```



```
rabbit = [0, 0]
pointing = [0, 0]

# Reward when yaw (car_orientation) is pointed to the next waypoint IN FRONT.

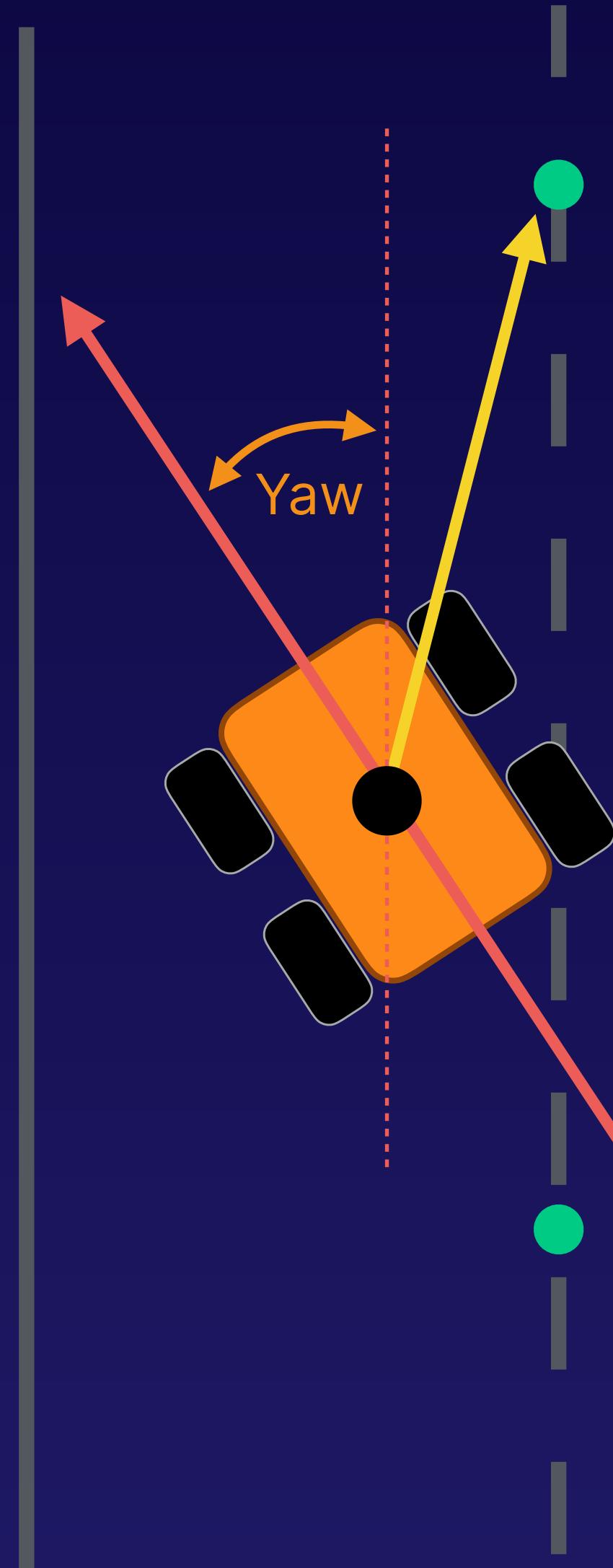
# Find nearest waypoint coordinates in front of car
rabbit = waypoints[closest_waypoints+1]

radius = math.sqrt((x - rabbit[0])**2 + (y - rabbit[1])**2 )

pointing[0] = x + (radius * math.cos(car_orientation))
pointing[1] = y + (radius * math.sin(car_orientation))

vector_delta = math.hypot(pointing[0] - rabbit[0], pointing[1] - rabbit[1])

# Max distance for pointing away will be the radius * 2
# Min distance means we are pointing directly at the next waypoint
```



```
rabbit = [0,0]
pointing = [0,0]

# Reward when yaw (car_orientation) is pointed to the next waypoint IN
FRONT.

# Find nearest waypoint coordinates in front of car
rabbit = waypoints[closest_waypoints+1]

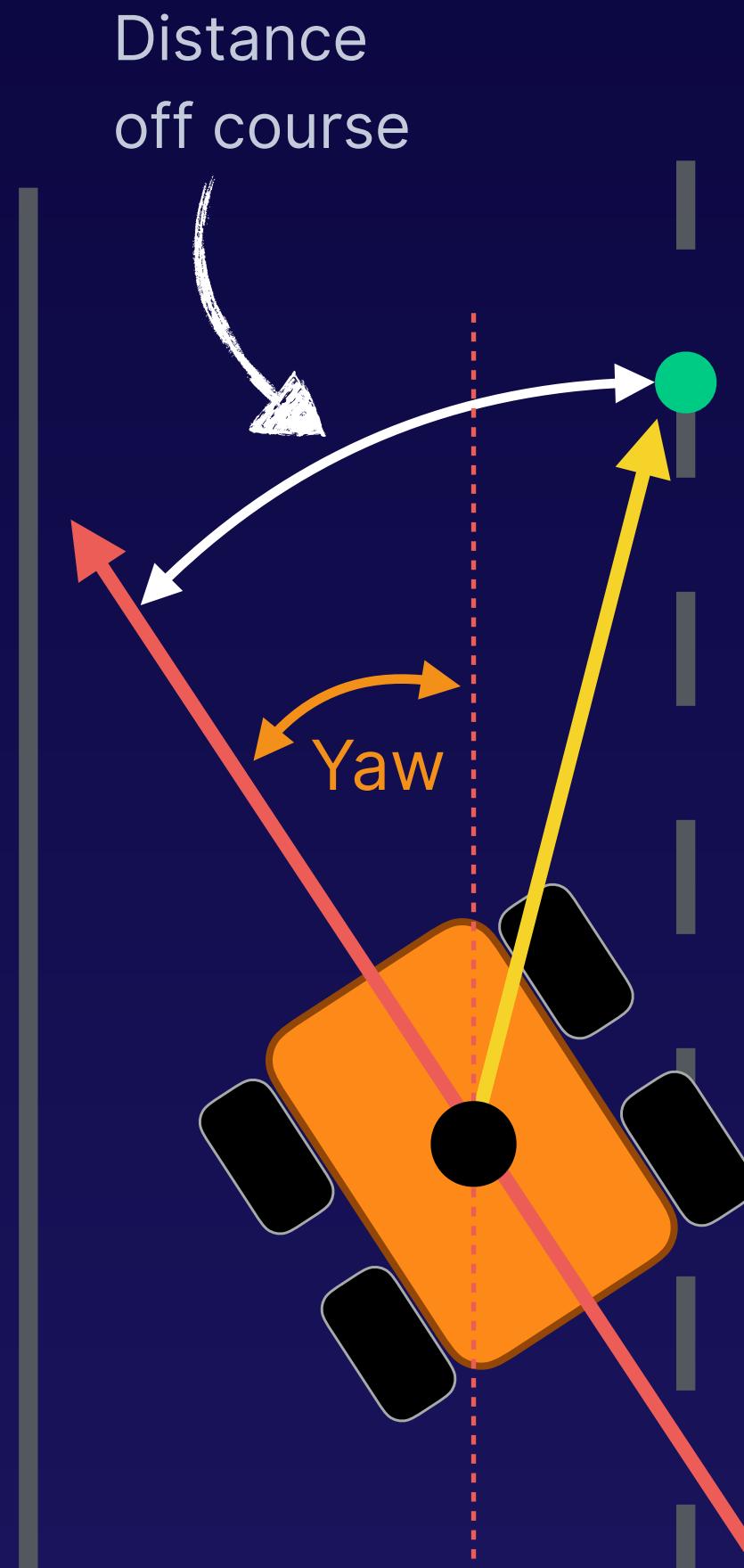
radius = math.sqrt((x - rabbit[0])**2 + (y - rabbit[1])**2 )

pointing[0] = x + (radius * math.cos(car_orientation))
pointing[1] = y + (radius * math.sin(car_orientation))

vector_delta = math.hypot(pointing[0] - rabbit[0], pointing[1] -
rabbit[1])

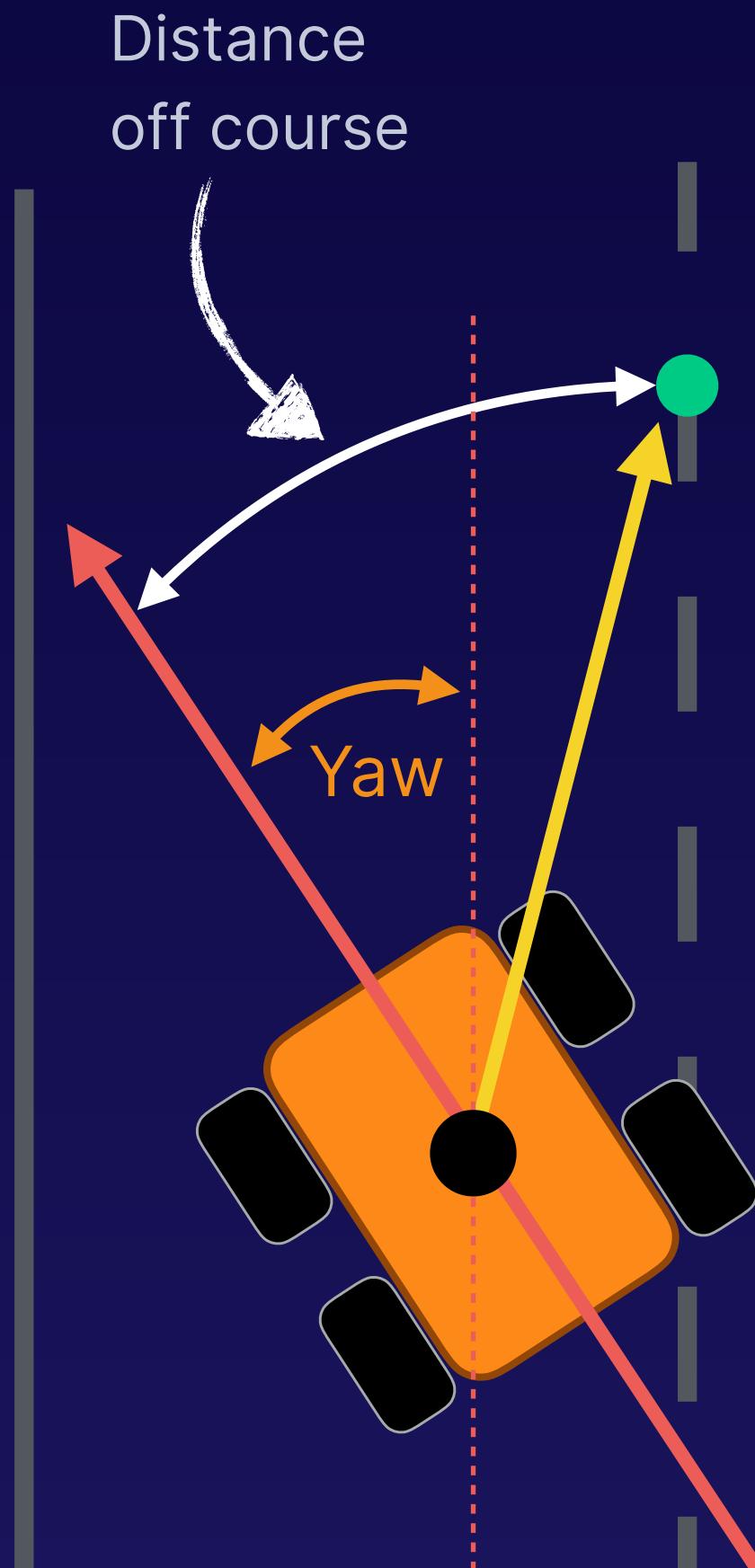
# Max distance for pointing away will be the radius * 2
# Min distance means we are pointing directly at the next waypoint
# We can setup a reward that is a ratio to this max.
```

```
pointing[0] = x + (radius * math.cos(car_orientation))
pointing[1] = y + (radius * math.sin(car_orientation))
```



```
vector_delta = math.hypot(pointing[0] - rabbit[0], pointing[1] -  
                           rabbit[1])
```

```
rabbit = [0, 0]  
pointing = [0, 0]  
  
# Reward when yaw (car_orientation) is pointed to the next waypoint IN  
FRONT.  
  
# Find nearest waypoint coordinates in front of car  
rabbit = waypoints[closest_waypoints+1]  
  
radius = math.sqrt((x - rabbit[0])**2 + (y - rabbit[1])**2 )  
  
pointing[0] = x + (radius * math.cos(car_orientation))  
pointing[1] = y + (radius * math.sin(car_orientation))  
  
vector_delta = math.hypot(pointing[0] - rabbit[0], pointing[1] -  
                           rabbit[1])  
  
# Max distance for pointing away will be the radius * 2  
# Min distance means we are pointing directly at the next waypoint  
# We can setup a reward that is a ratio to this max.
```



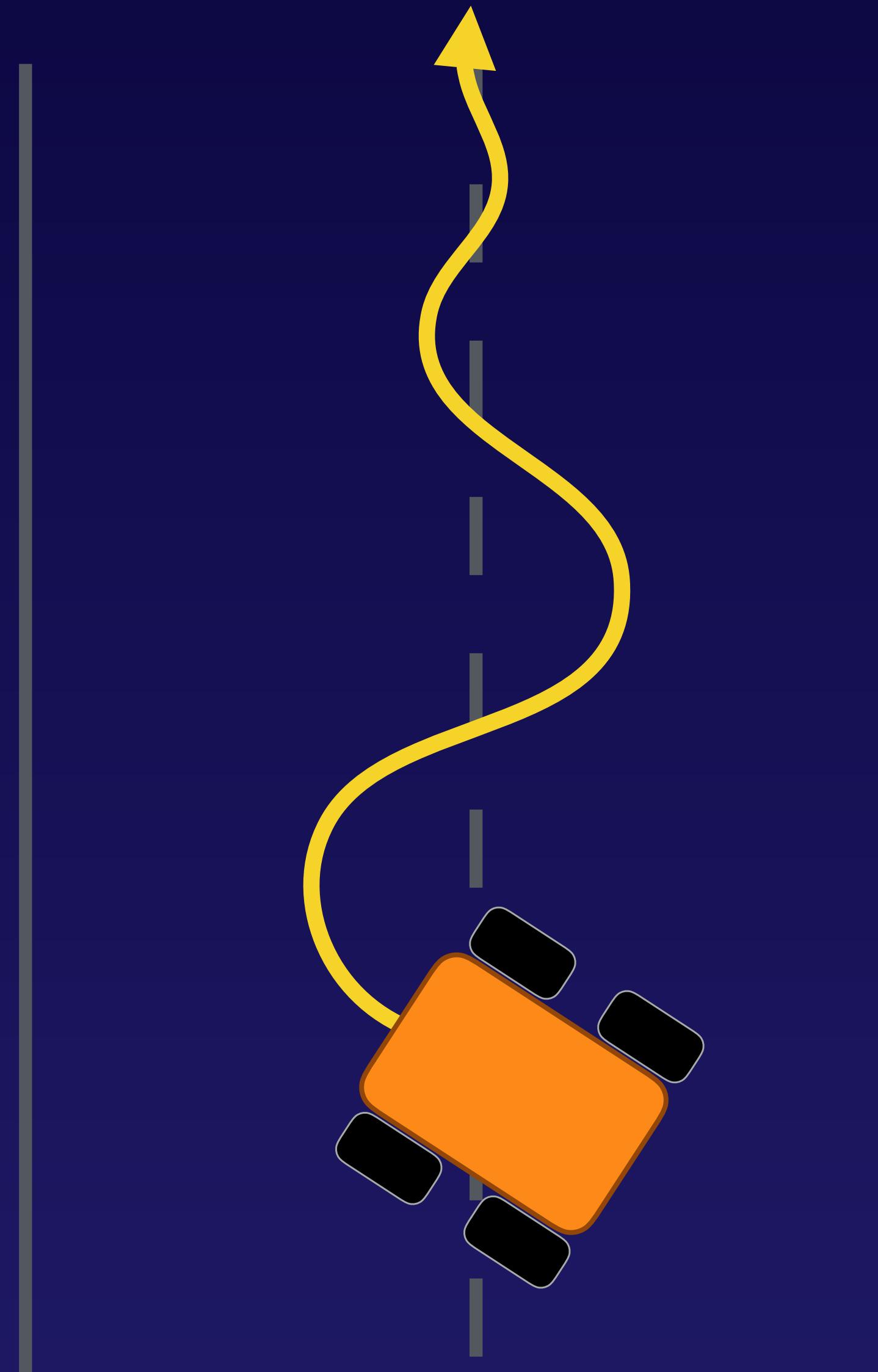
```
rabbit = [0, 0]
pointing = [0, 0]

# Reward when yaw (car_orientation) is pointed to the next waypoint IN FRONT.
```

```
if vector_delta == 0:
    reward = 1
else:
    reward = ( 1 - vector_delta / (radius * 2))
```

```
# Max distance for pointing away will be the radius * 2
# Min distance means we are pointing directly at the next waypoint
# We can setup a reward that is a ratio to this max.
```

```
if vector_delta == 0:
    reward = 1
else:
    reward = ( 1 - vector_delta / (radius * 2))
```



```
rabbit = [0,0]
pointing = [0,0]

# Reward when yaw (car_orientation) is pointed to the next waypoint IN
FRONT.

# Find nearest waypoint coordinates in front of car
rabbit = waypoints[closest_waypoints+1]

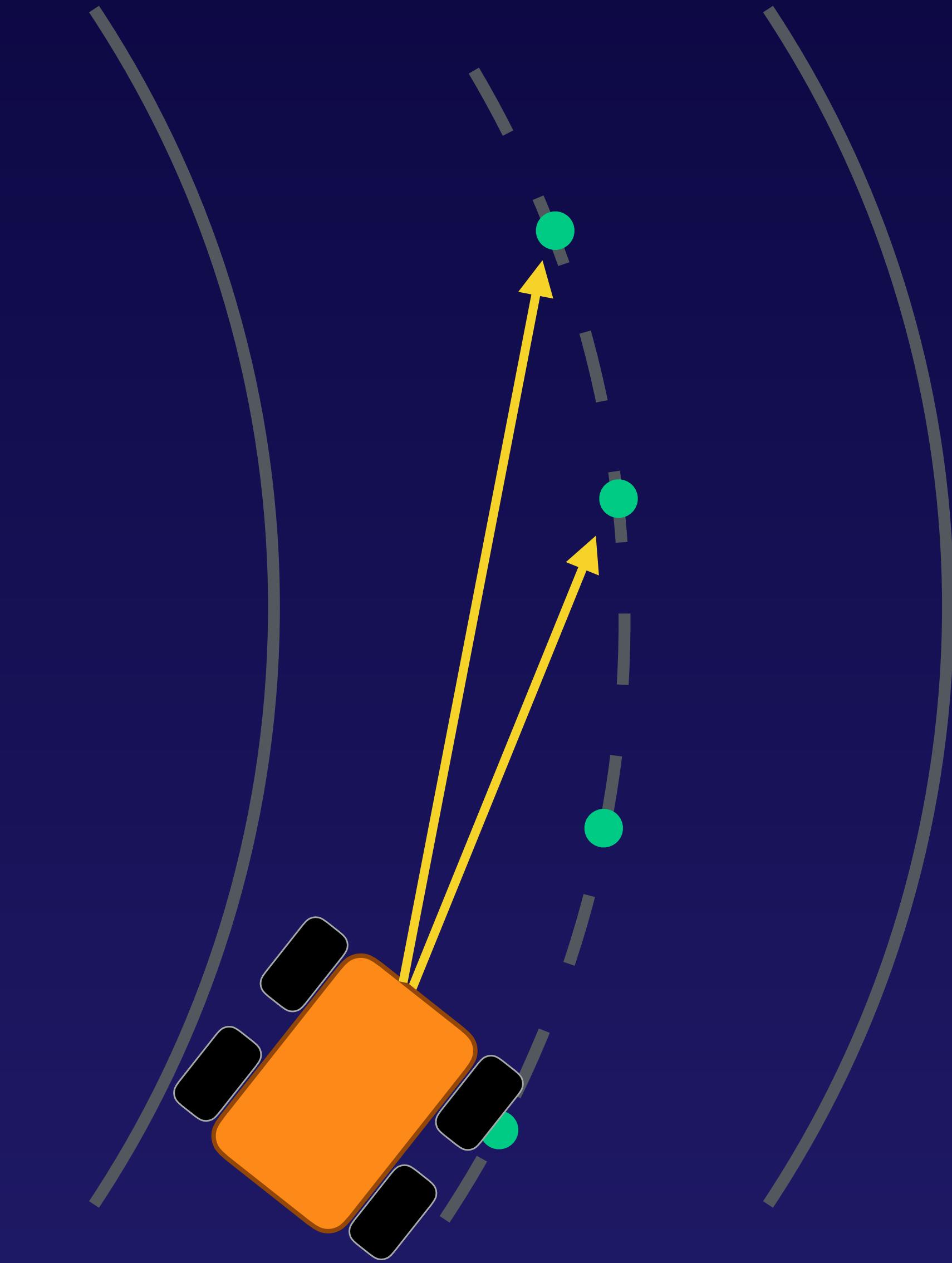
radius = math.sqrt((x - rabbit[0])**2 + (y - rabbit[1])**2)

pointing[0] = x + (radius * math.cos(car_orientation))
pointing[1] = y + (radius * math.sin(car_orientation))

vector_delta = math.hypot(pointing[0] - rabbit[0], pointing[1] -
rabbit[1])

# Max distance for pointing away will be the radius * 2
# Min distance means we are pointing directly at the next waypoint
# We can setup a reward that is a ratio to this max.

if vector_delta == 0:
    reward = 1
else:
    reward = ( 1 - vector_delta / (radius * 2))
```



```
rabbit = [0,0]
pointing = [0,0]

# Reward when yaw (car_orientation) is pointed to the next waypoint IN
FRONT.

# Find nearest waypoint coordinates in front of car
rabbit = waypoints[closest_waypoints+1]

radius = math.sqrt((x - rabbit[0])**2 + (y - rabbit[1])**2 )

pointing[0] = x + (radius * math.cos(car_orientation))
pointing[1] = y + (radius * math.sin(car_orientation))

vector_delta = math.hypot(pointing[0] - rabbit[0], pointing[1] -
rabbit[1])

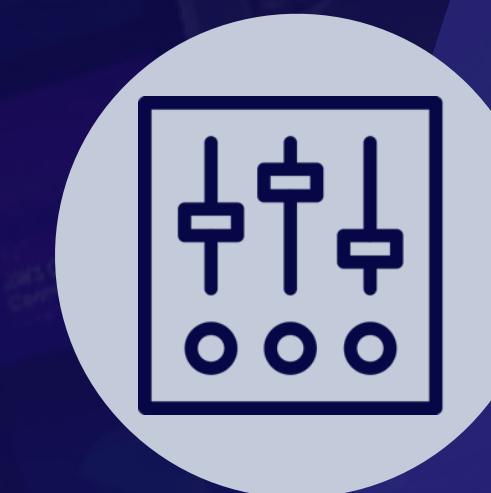
# Max distance for pointing away will be the radius * 2
# Min distance means we are pointing directly at the next waypoint
# We can setup a reward that is a ratio to this max.

if vector_delta == 0:
    reward = 1
else:
    reward = ( 1 - vector_delta / (radius * 2))
```

# *Scott's Slightly Credible* Tips For DeepRacer Glory™



Dare to be Different



Turn the Knobs



▼ **Hyperparameters** Info

Specify hyperparameters to tune your training performance.

Batch size

32  
 64  
 128  
 256  
 512

Number of epochs

Spin

Integer between 3 and 10.

Learning rate

Spin

Real number between 0.00000001 (1e-8) and 0.001 (1e-3).

Exploration

Categorical Parameters  
 Epsilon Greedy

Entropy

Spin

Real number between 0 and 1.

Discount factor

Spin

Real number between 0 and 1.

Loss type

Mean square error  
 Huber

Number of episodes between each training

Spin

Integer between 5 and 100.





▼ **Hyperparameters** [Info](#)

Specify hyperparameters to tune your training performance.

**Batch size**

32  
 64  
 128  
 256  
 512

**Number of epochs**

10

Integer between 3 and 10.

**Learning rate**

0.0003

Real number between 0.00000001 (1e-8) and 0.001 (1e-3).

**Exploration**

Categorical Parameters  
 Epsilon Greedy

**Entropy**

0.01

Real number between 0 and 1.

**Discount factor**

0.999

Real number between 0 and 1.

**Loss type**

Mean square error  
 Huber

**Number of episodes between each training**

20

Integer between 5 and 100.



▼ **Hyperparameters** [Info](#)

Specify hyperparameters to tune your training performance.

Batch size

32  
 64  
 128  
 256  
 512

Number of epochs

**Learning rate**

0.0003 (Up/Down)

Real number between 0.00000001 (1e-8) and 0.001 (1e-3).

Epsilon Greedy

Entropy

0.01 (Up/Down)

Real number between 0 and 1.

Discount factor

0.999 (Up/Down)

Real number between 0 and 1.

Loss type

Mean square error  
 Huber

Number of episodes between each training

20 (Up/Down)

Integer between 5 and 100.



Given a cost function  $J$  defined as:

$$J(W, b) = \frac{1}{m} \sum_{z=0}^m J(W, b, x^{(z)}, y^{(z)})$$

And a normal batch gradient decent taking the mean squared error of all the samples:

$$W = W - \alpha \nabla J(W, b)$$

The learning rate is defined as  $\alpha$ .

▼ **Hyperparameters** [Info](#)  
Specify hyperparameters to tune your training performance.

Batch size  
 32  
 64  
 128  
 256  
 512

Number of epochs

**Learning rate**  
0.0003

Real number between 0.00000001 (1e-8) and 0.001 (1e-3).

Epsilon Greedy

Entropy  
0.01   
Real number between 0 and 1.

Discount factor  
0.999   
Real number between 0 and 1.

Loss type  
 Mean square error  
 Huber

Number of episodes between each training  
20   
Integer between 5 and 100.



Given a cost function  $J$  defined as:

$$J(W, b) = \frac{1}{m} \sum_{z=0}^m J(W, b, x^{(z)}, y^{(z)})$$

And a normal batch gradient decent taking the mean squared error of all the samples:

$$W = W - \alpha \nabla J(W, b)$$

The learning rate is defined as  $\alpha$ .



▼ **Hyperparameters** Info  
Specify hyperparameters to tune your training performance.

Batch size  
 32  
 64  
 128  
 256  
 512

Number of epochs

**Learning rate**  
0.0003  
Real number between 0.00000001 (1e-8) and 0.001 (1e-3).

Epsilon Greedy

Entropy  
0.01  
Real number between 0 and 1.

Discount factor  
0.999  
Real number between 0 and 1.

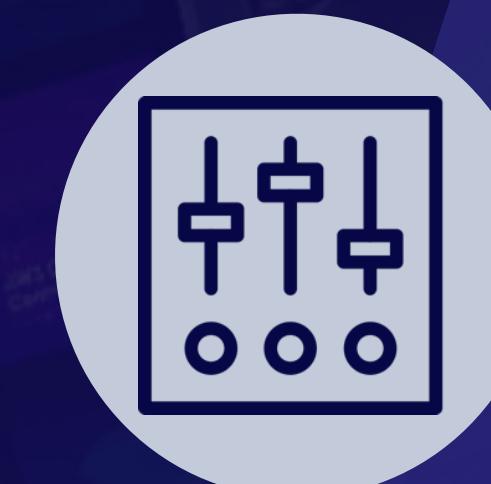
Loss type  
 Mean square error  
 Huber

Number of episodes between each training  
20  
Integer between 5 and 100.

# *Scott's Slightly Credible* Tips For DeepRacer Glory™



Dare to be Different



Turn the Knobs



Don't Fear the Greek



▼ **Hyperparameters** [Info](#)  
Specify hyperparameters to tune your training performance.

Batch size

32  
 64  
 128  
 256  
 512

Number of epochs

**Learning rate**

0.0003 (Up/Down)

Real number between 0.00000001 (1e-8) and 0.001 (1e-3).

Epsilon Greedy

Entropy

0.01 (Up/Down)

Real number between 0 and 1.

Discount factor

0.999 (Up/Down)

Real number between 0 and 1.

Loss type

Mean square error  
 Huber

Number of episodes between each training

20 (Up/Down)

Integer between 5 and 100.







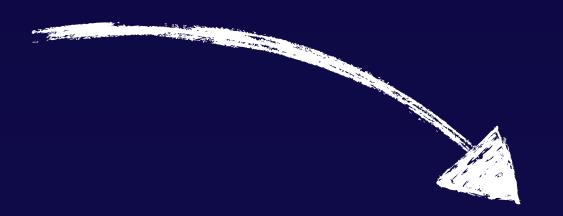








Larger Learning Rate



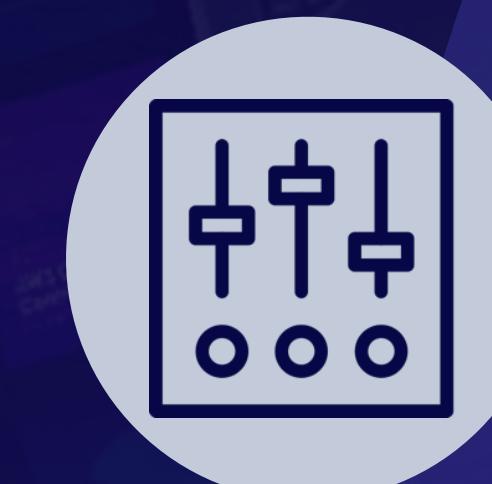
Smaller Learning Rate



# *Scott's Slightly Credible* Tips For DeepRacer Glory™



Dare to be Different



Turn the Knobs



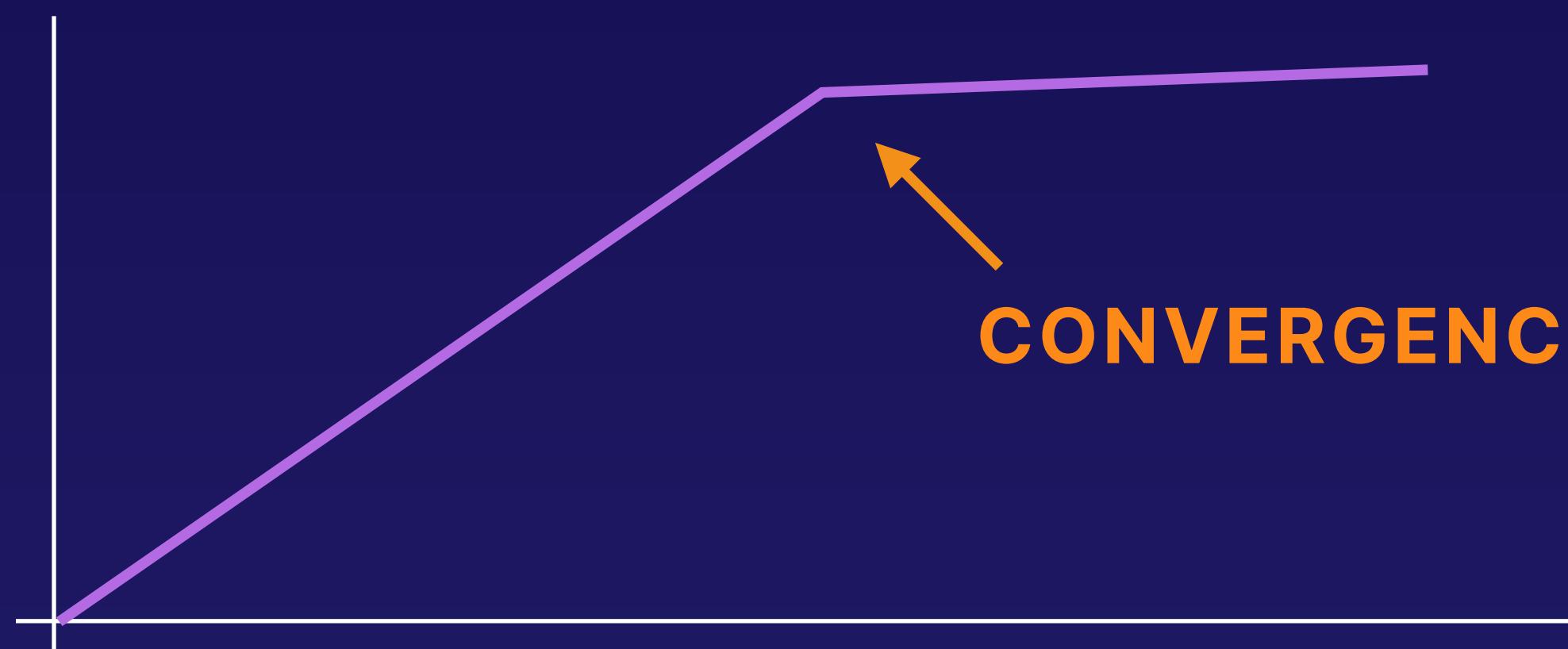
Don't Fear the Greek



Iterate. Iterate. Iterate.



Round	Learning Rate	Coverged At
1	0.0010	50 points
2	0.0005	75 points
3	0.0003	100 points
4	0.0001	104 points



▼ Hyperparameters [Info](#)  
Specify hyperparameters to tune your training performance.

Batch size  
 32  
 64  
 128  
 256  
 512

Number of epochs

Learning rate  
0.0003

Real number between 0.00000001 (1e-8) and 0.001 (1e-3).

Epsilon Greedy

Entropy  
0.01   
Real number between 0 and 1.

Discount factor  
0.999   
Real number between 0 and 1.

Loss type  
 Mean square error  
 Huber

Number of episodes between each training  
20   
Integer between 5 and 100.



# SCOTT

Models (27)

Search models

Name
StayLeft-GetFast-10degree-5ms-4h-clone
StartFresh-PP-4h-v2
StartFresh-PP-4h
StayLeft-GetFast-10degree-5ms-4h
PurePursuit-RabbitAhead-2h
PurePursuit-RabbitAhead
PurePursuit-GetFast-ProgressIncentive-WithoutPure
PurePursuit-GetFast-ProgressIncentive
GetFast-PurePursuit-Bowtie-3h-clone
GetFast-PurePursuit-Bowtie-3h

# NICK K.

Name
CLV2
CLV1
TRLV5-SM
RLV9-C-GEN
RLV7-C-SM
RLV6-C-SM
RLV5-C-SM
RLV8-C-GEN
RLV8-B-SM
RLV7-B-SM
RLV7-B-Gen
RLV6-B
RLV4-B-clone
RLV4-B
RLV3-B
RLV3
RLV2
Rabbit-clone-clone
RLV1
Rabbit-clone
Newbe-clone-clone
OldboyV7
MRBasicV3

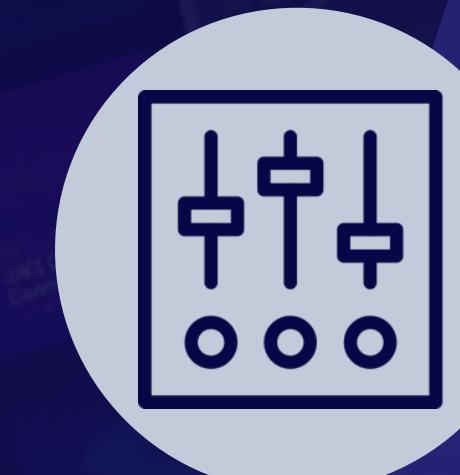
# NICK T.

Name
Winner-1h-reinvent-pure-clone
Winner-1h-reinvent-pure
nick-nicksgoodone-clone-clone-clone-clone-Pure-clone-clone
nick-nicksgoodone-clone-clone-clone-clone-Pure-clone
nick-nicksgoodone-clone-clone-clone-clone-Pure
nick-nicksgoodone-clone-clone-clone
nick-nicksgoodone-clone-clone
nick-nicksgoodone-clone
nick-nicksgoodone
Nick

# *Scott's Slightly Credible* Tips For DeepRacer Glory™



Dare to be Different



Turn the Knobs



Don't Fear the Greek



Iterate. Iterate. Iterate.



# Supervised Learning

## TRAINING PROCESS

You feed in data that you know to be correct (ground truth) so the training process has something to model.

## OUTCOME

Your models can only be as good as the examples provided.





# Reinforcement Learning

**TRAINING PROCESS** You provide a reward system to encourage desired behavior.

**OUTCOME** Your model can become better than anyone that has ever done that thing before.



# Use the Reinforce!





A CLOUD GURU

# Thank You & Keep Being Awesome!



@scottpletcher



[linkedin.com/in/scottpletcher](https://linkedin.com/in/scottpletcher)