SUMMIT

ATLANTA

AWS DeepRacer ~~101~~ 102: Lessons from Victory Lane

Scott Pletcher
Instructor
A Cloud Guru

aws SUMMIT

A CLOUD GURU

Continuum of AWS Bad-Assery

Amazon
DeepRacer

AFTER THIS
SESSION

ZERO
EXPERIENCE

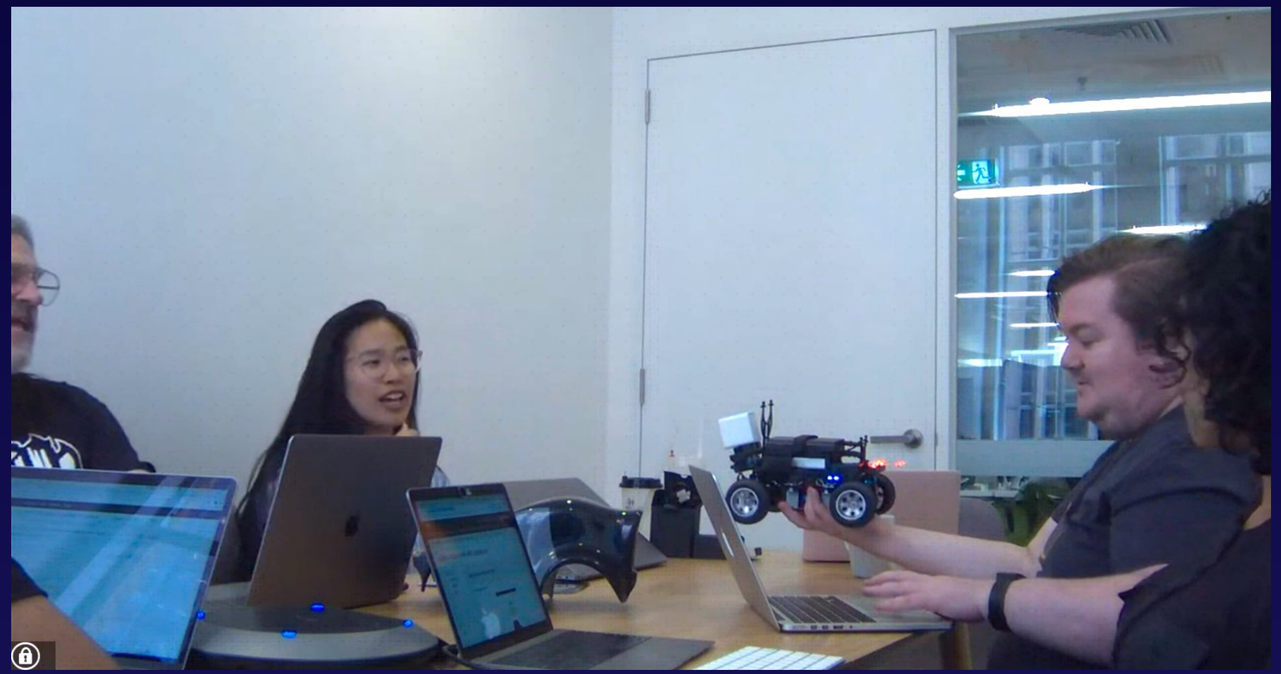
AFTER
WORKSHOP
OR EXPLORING

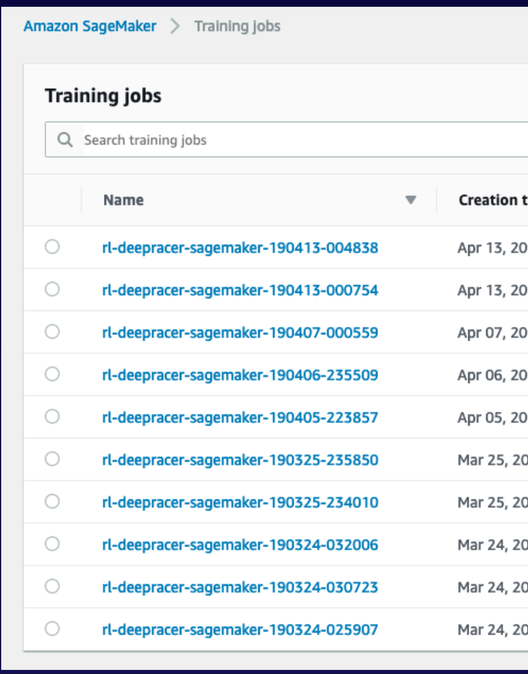
A CLOUD GURU

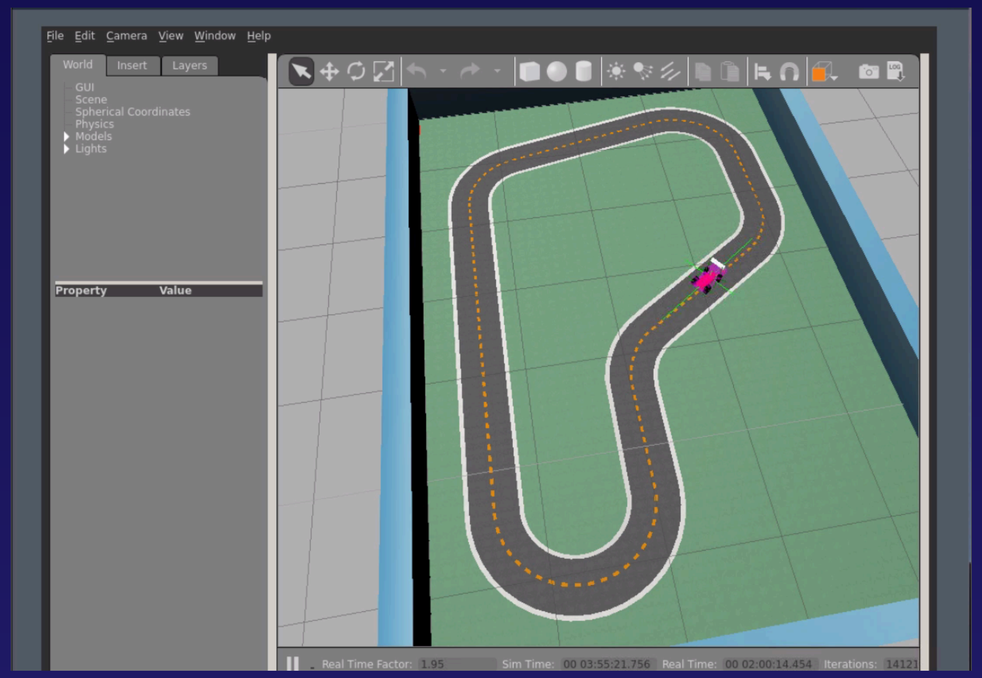
“AWS DeepRacer is a fully autonomous 1/18th scale race car designed to get you rolling with reinforcement learning.”

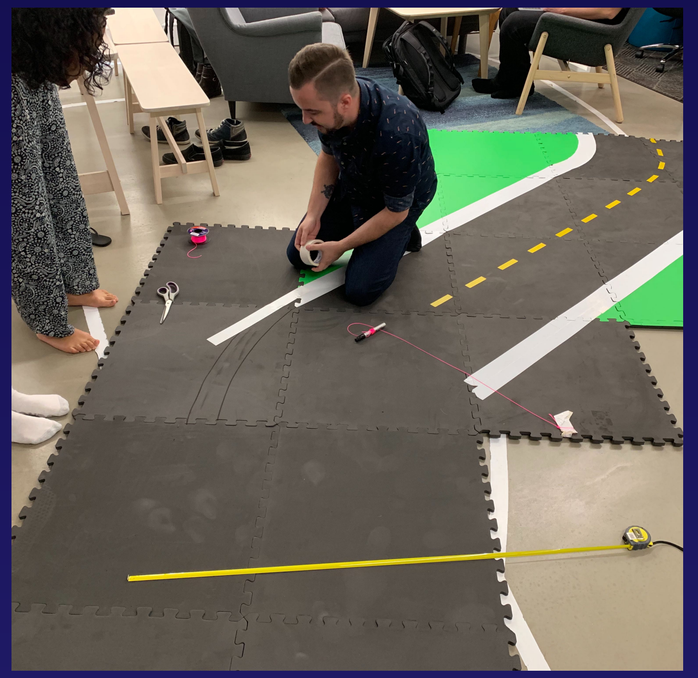
Dr. Matt Wood

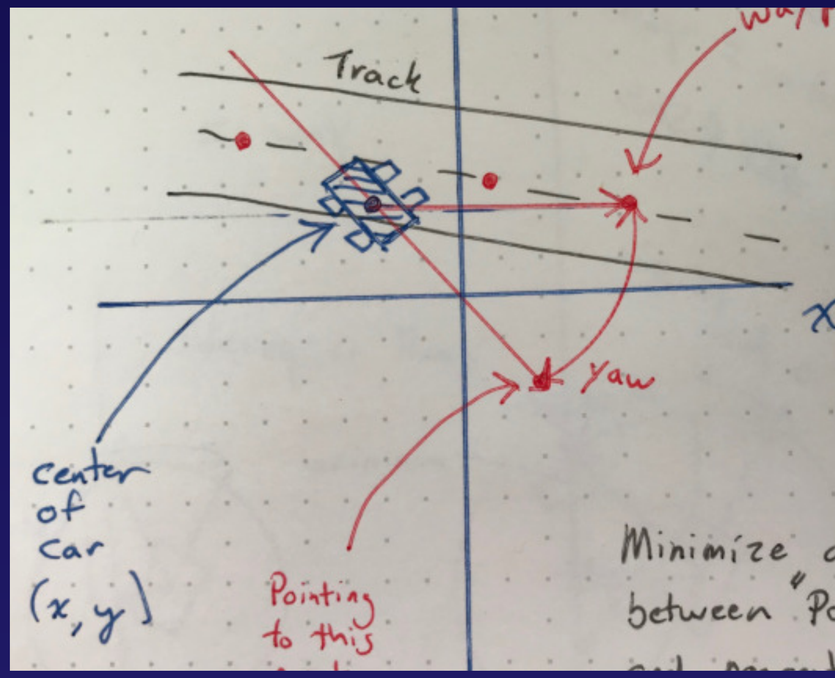
GM, Deep Learning and AI
Amazon Web Services




Amazon SageMaker
Training jobs
Training jobs
Search training jobs
Name
Creation t
rl-deepracer-sagemaker-190413-004838
Apr 13, 20
rl-deepracer-sagemaker-190413-000754
Apr 13, 20
rl-deepracer-sagemaker-190407-000559
Apr 07, 20
rl-deepracer-sagemaker-190406-235509
Apr 06, 20
rl-deepracer-sagemaker-190405-223857
Apr 05, 20
rl-deepracer-sagemaker-190325-235850
Mar 25, 20
rl-deepracer-sagemaker-190325-234010
Mar 25, 20
rl-deepracer-sagemaker-190324-032006
Mar 24, 20
rl-deepracer-sagemaker-190324-030723
Mar 24, 20
rl-deepracer-sagemaker-190324-025907
Mar 24, 20


File Edit Camera View Window Help
World Insert Layers
GUI
Scene
Spherical Coordinates
Physics
Models
Lights
Property Value
Real Time Factor: 1.95 Sim Time: 00 03:55:21.756 Real Time: 00 02:00:14.454 Iterations: 14121

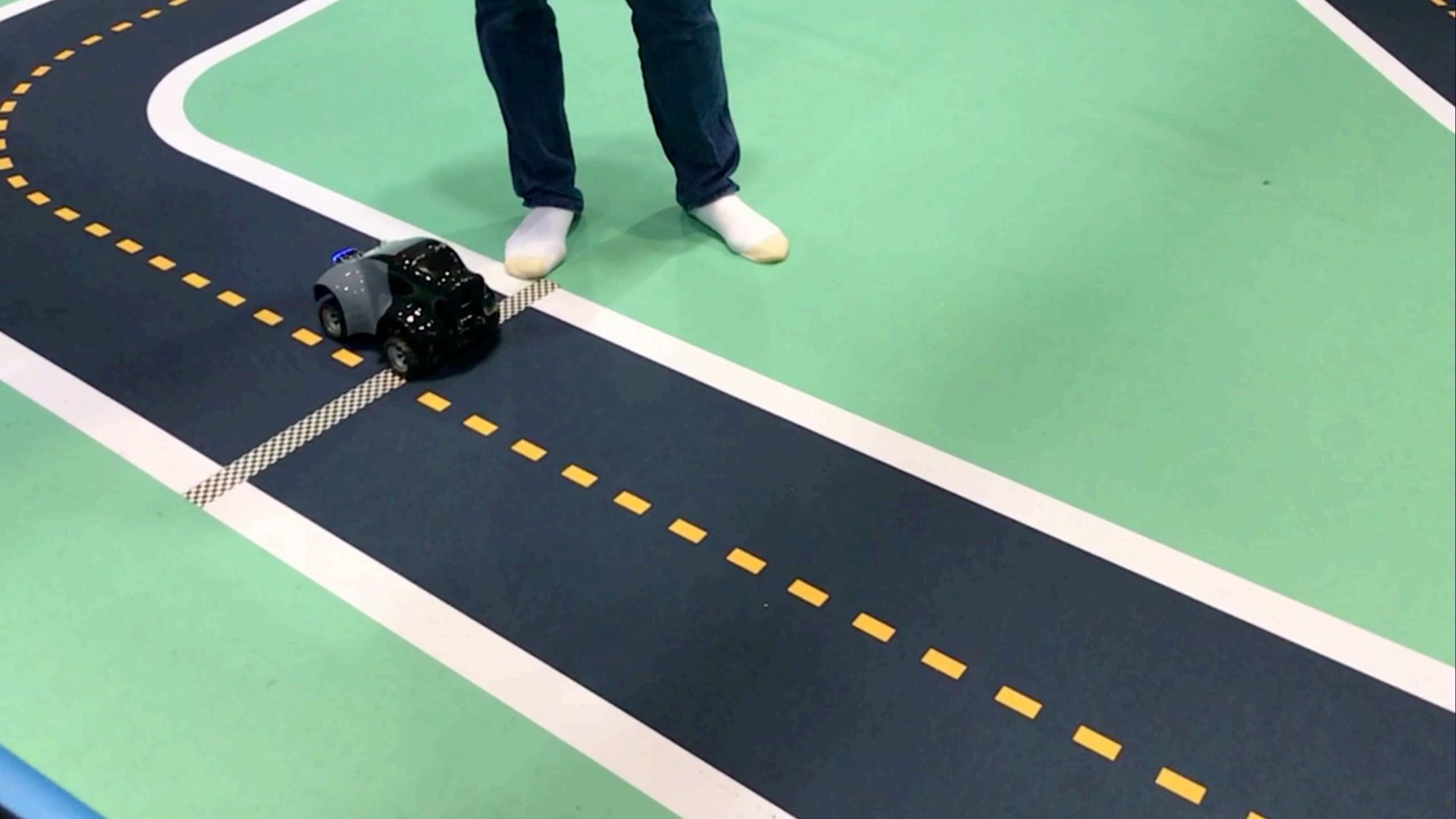



Track
Yaw
center of car (x, y)
Pointing to this
Minimize
between

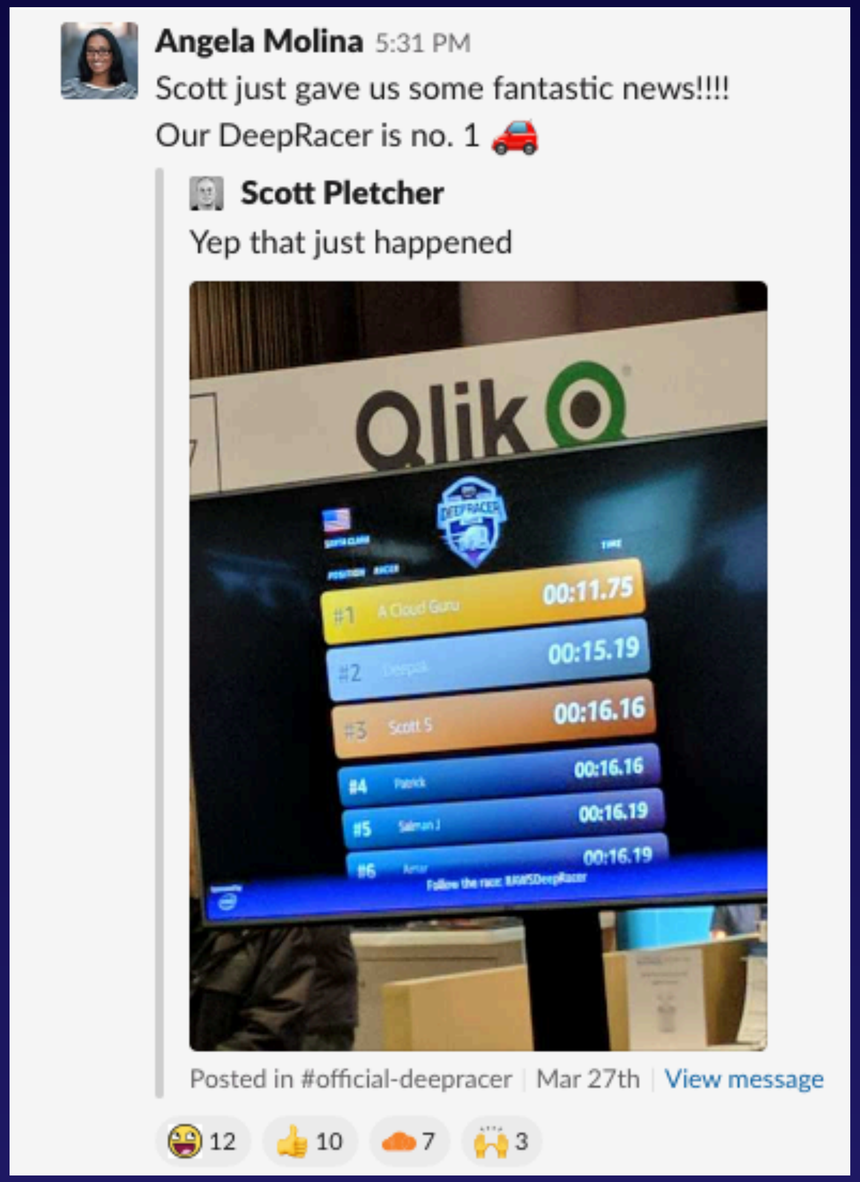
A CLOUD GURU

AWS DeepRacer League

Santa Clara

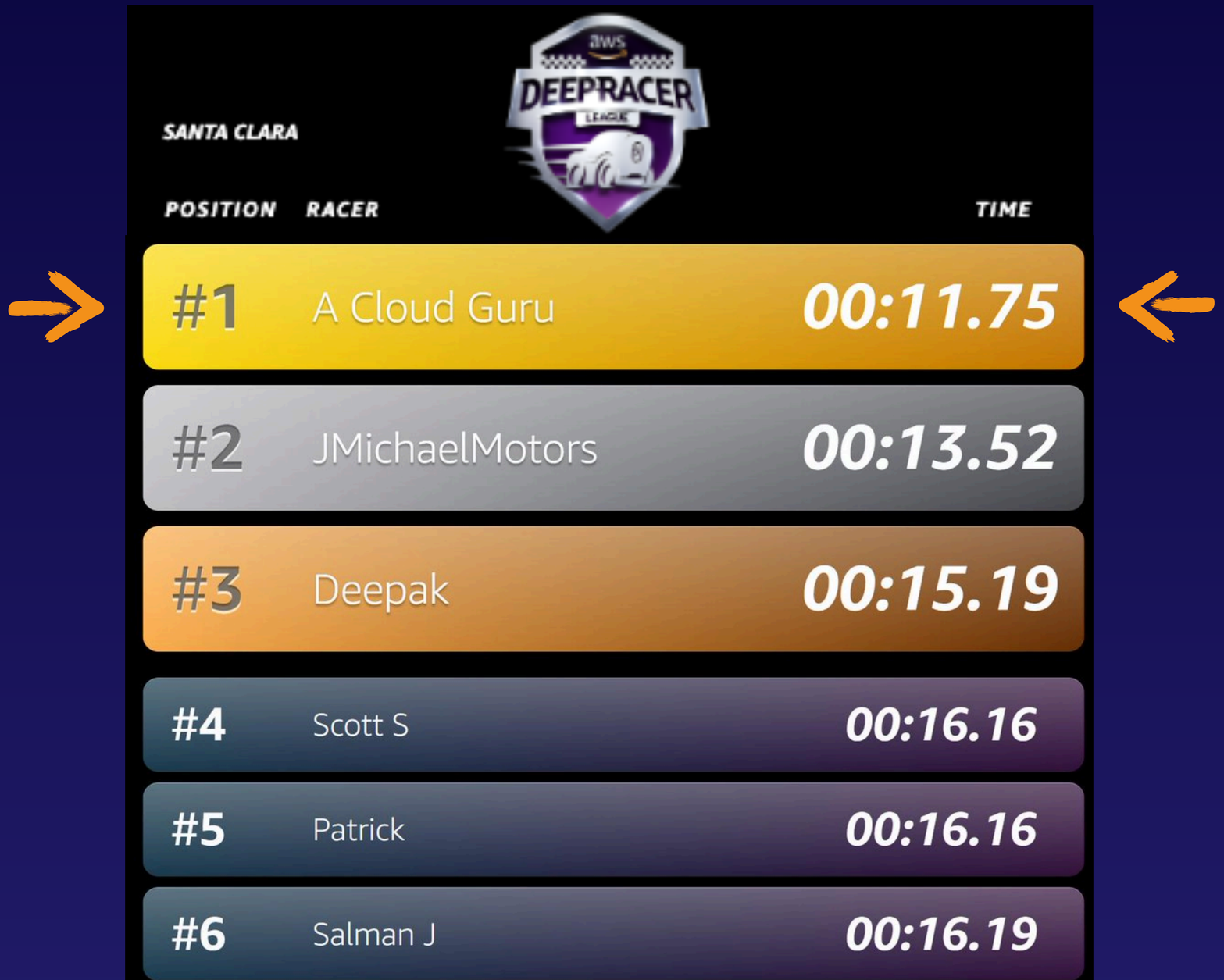


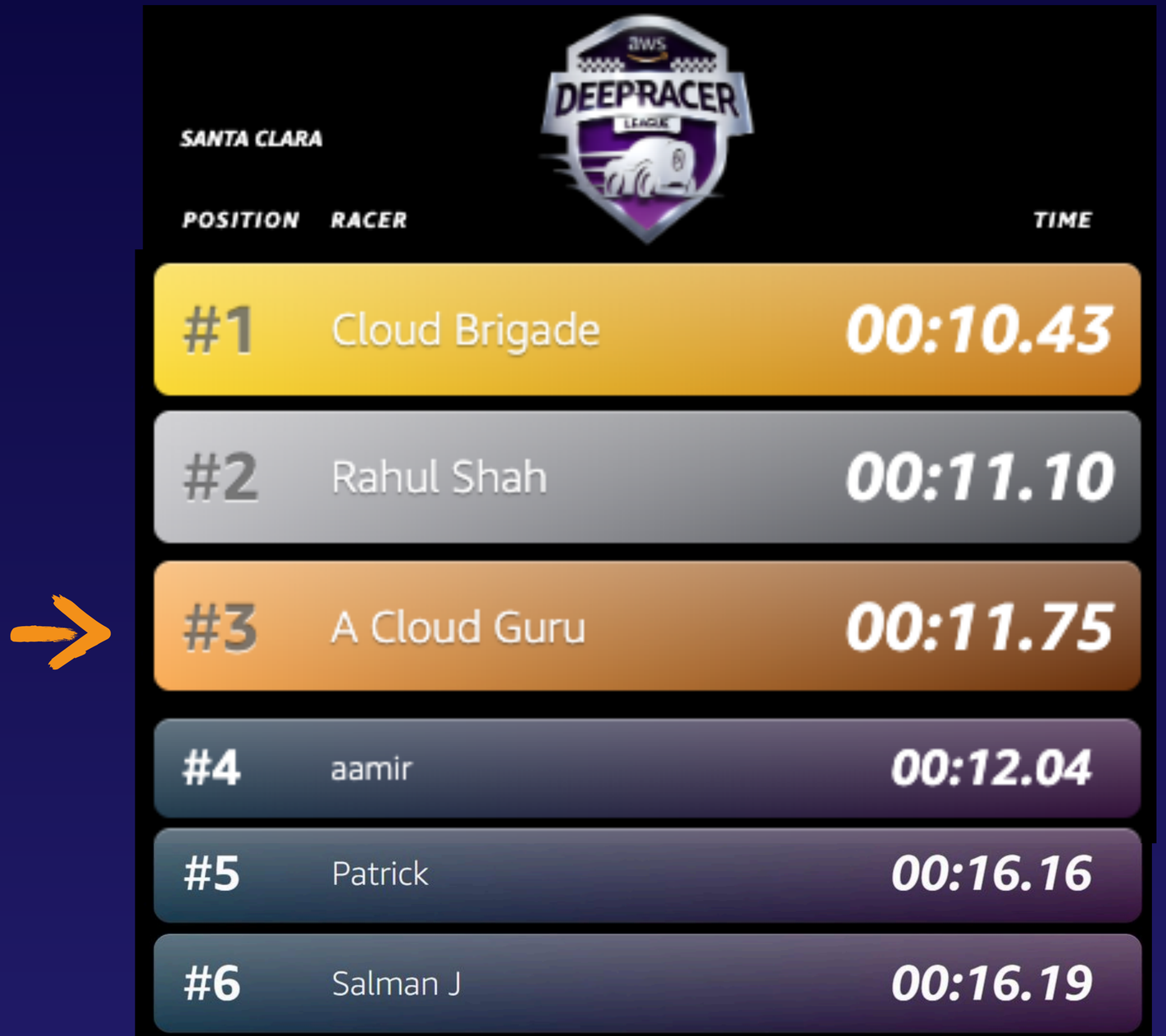

SANTA CLARA
aws
DEEPRACER
LEAGUE
POSITION
RACER
TIME
#1 A Cloud Guru 00:11.75
#2 JMichaelMotors 00:13.52
#3 Deepak 00:15.19
#4 Scott S 00:16.16
#5 Patrick 00:16.16
#6 Salman J 00:16.19
#7 Amar 00:16.19


Angela Molina 5:31 PM
Scott just gave us some fantastic news!!!!
Our DeepRacer is no. 1 🚗
Scott Pletcher
Yep that just happened
Qlik
#1 A Cloud Guru 00:11.75
#2 Deepak 00:15.19
#3 Scott S 00:16.16
#4 Patrick 00:16.16
#5 Salman J 00:16.19
#6 Amar 00:16.19
Posted in #official-deepracer | Mar 27th | View message
12
10
7
3

😲 OMG 😲 !!!!


apps associates
Qlik
COMCAST BUSINESS
MATILLION


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

A CLOUD GURU

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

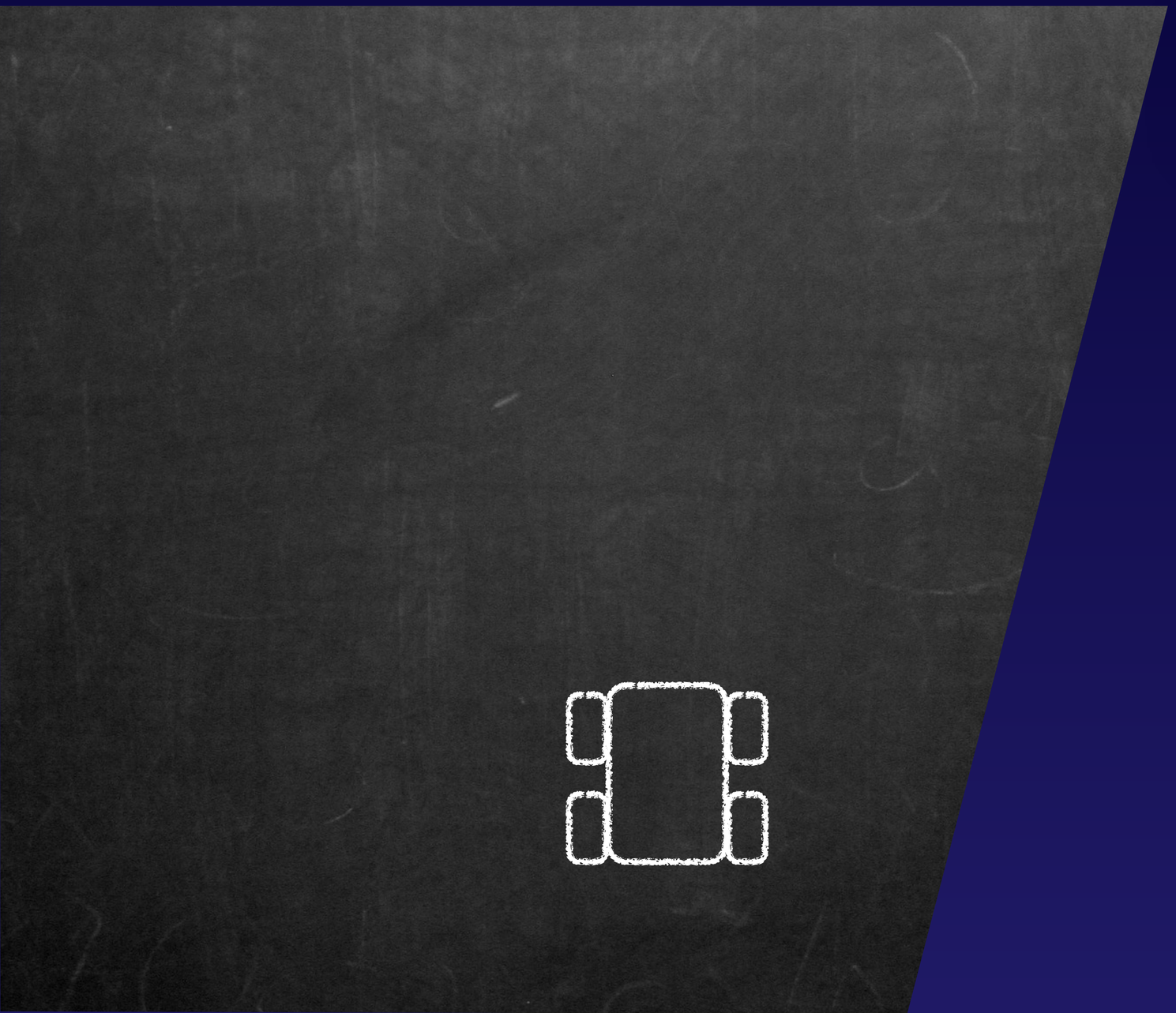
A CLOUD GURU

Scott's Slightly Credible

Tips For DeepRacer Glory™

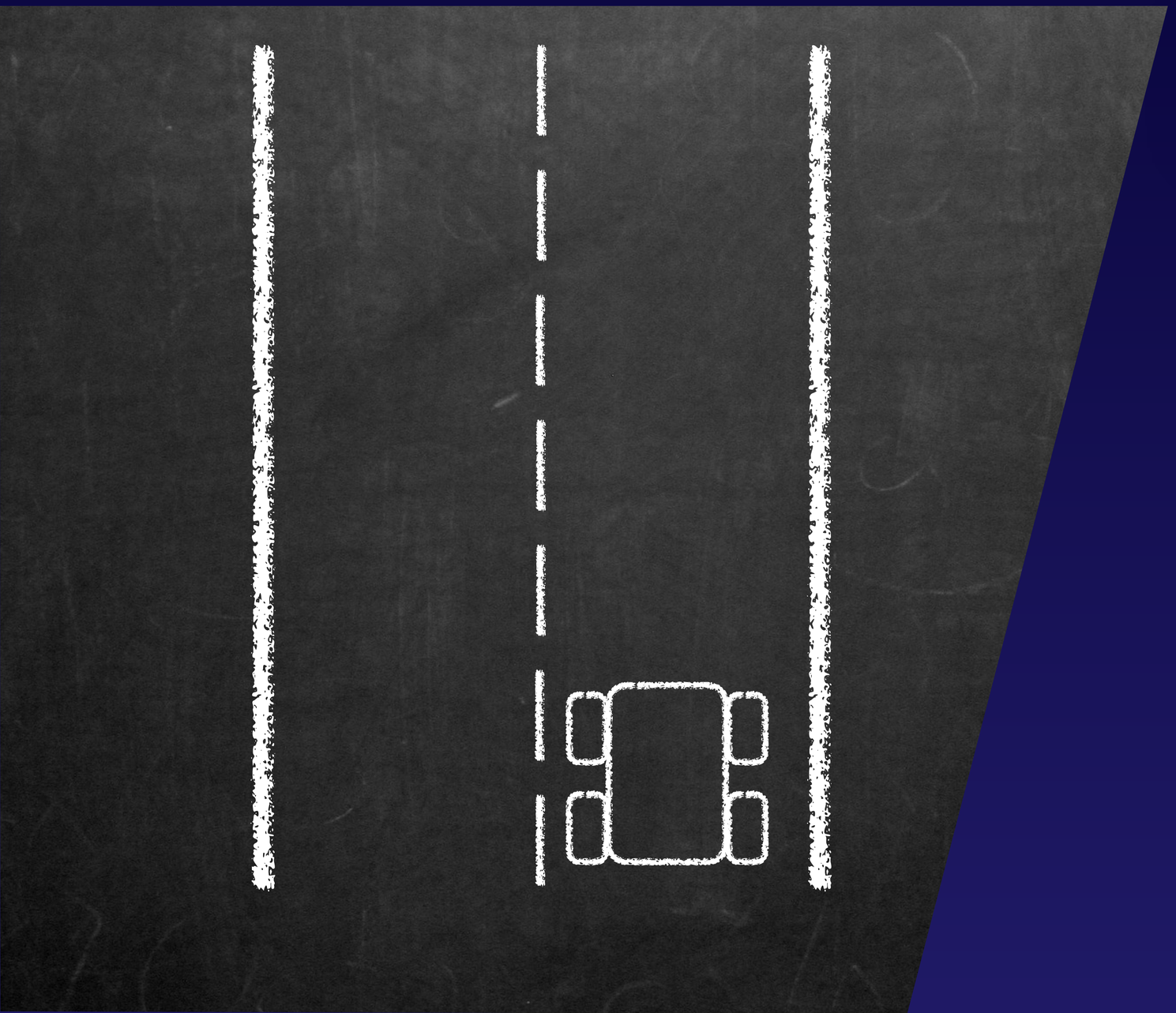
Please consult physician before DeepRacing. Common side effects of DeepRacer include insomnia, headaches and junk food consumption. Scott is not responsible for obscene AWS bills incurred by hours of training DeepRacer.

A CLOUD GURU



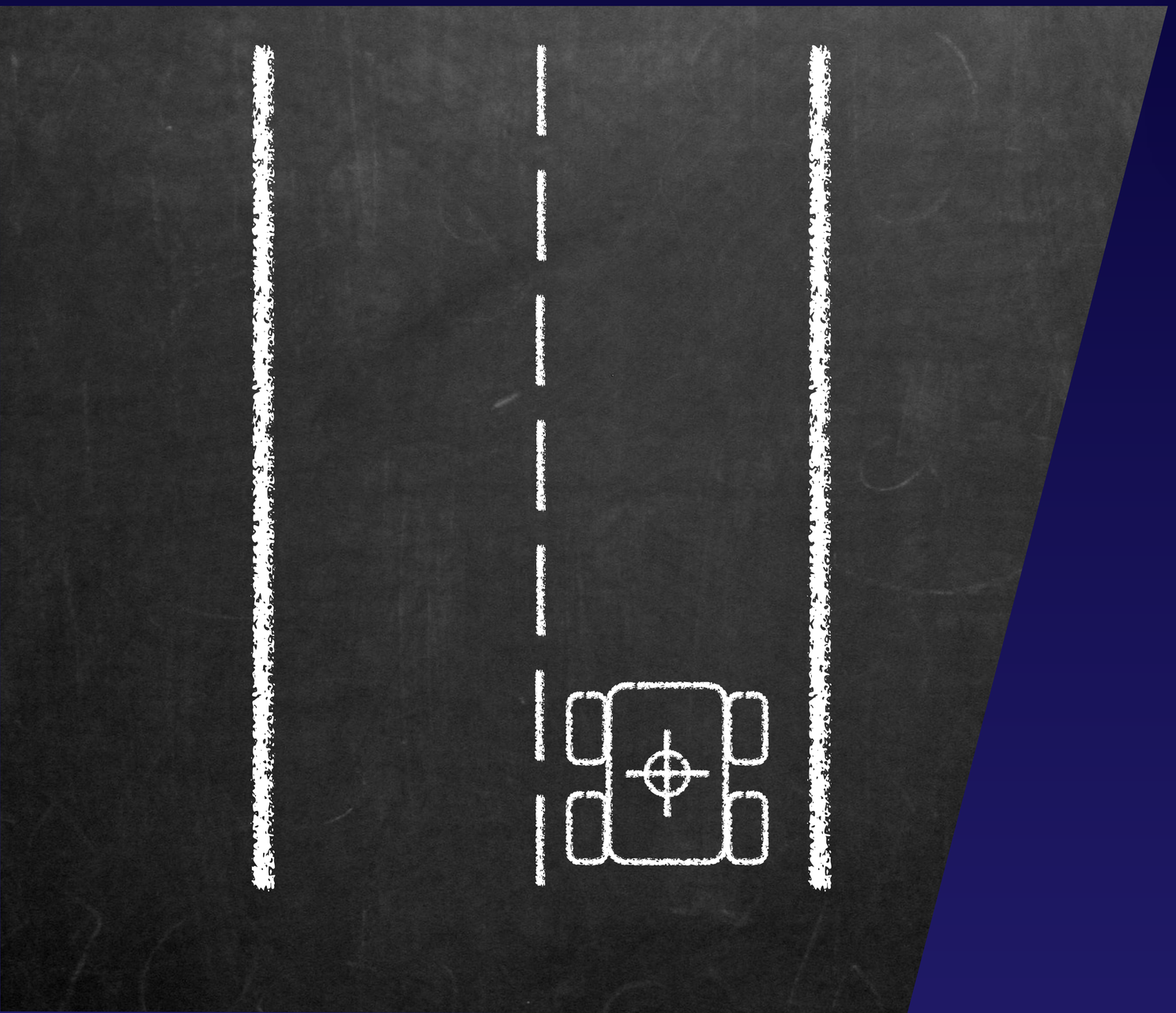
Markov Decision Process

AGENT



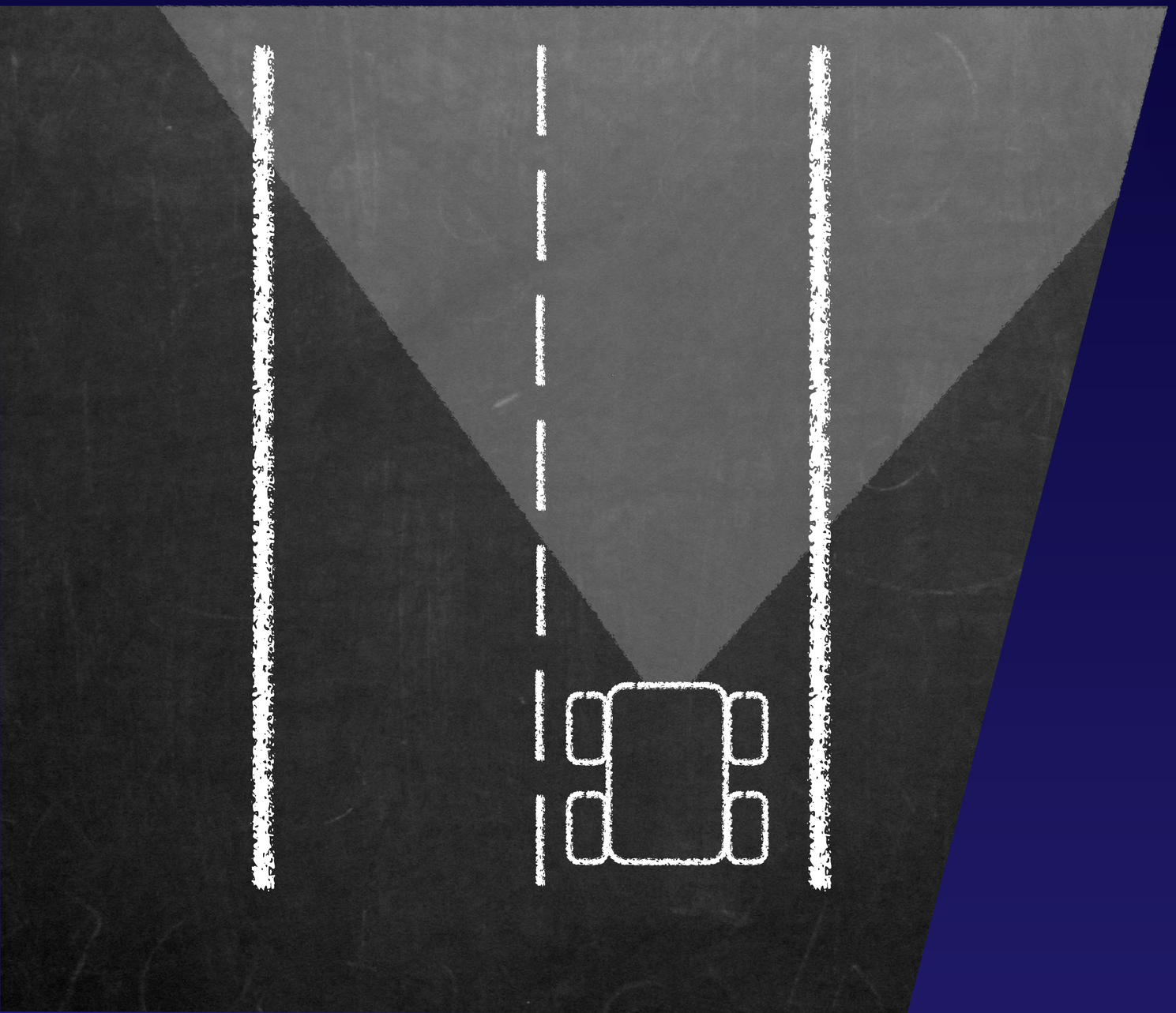
Markov Decision Process

- AGENT
- ENVIRONMENT



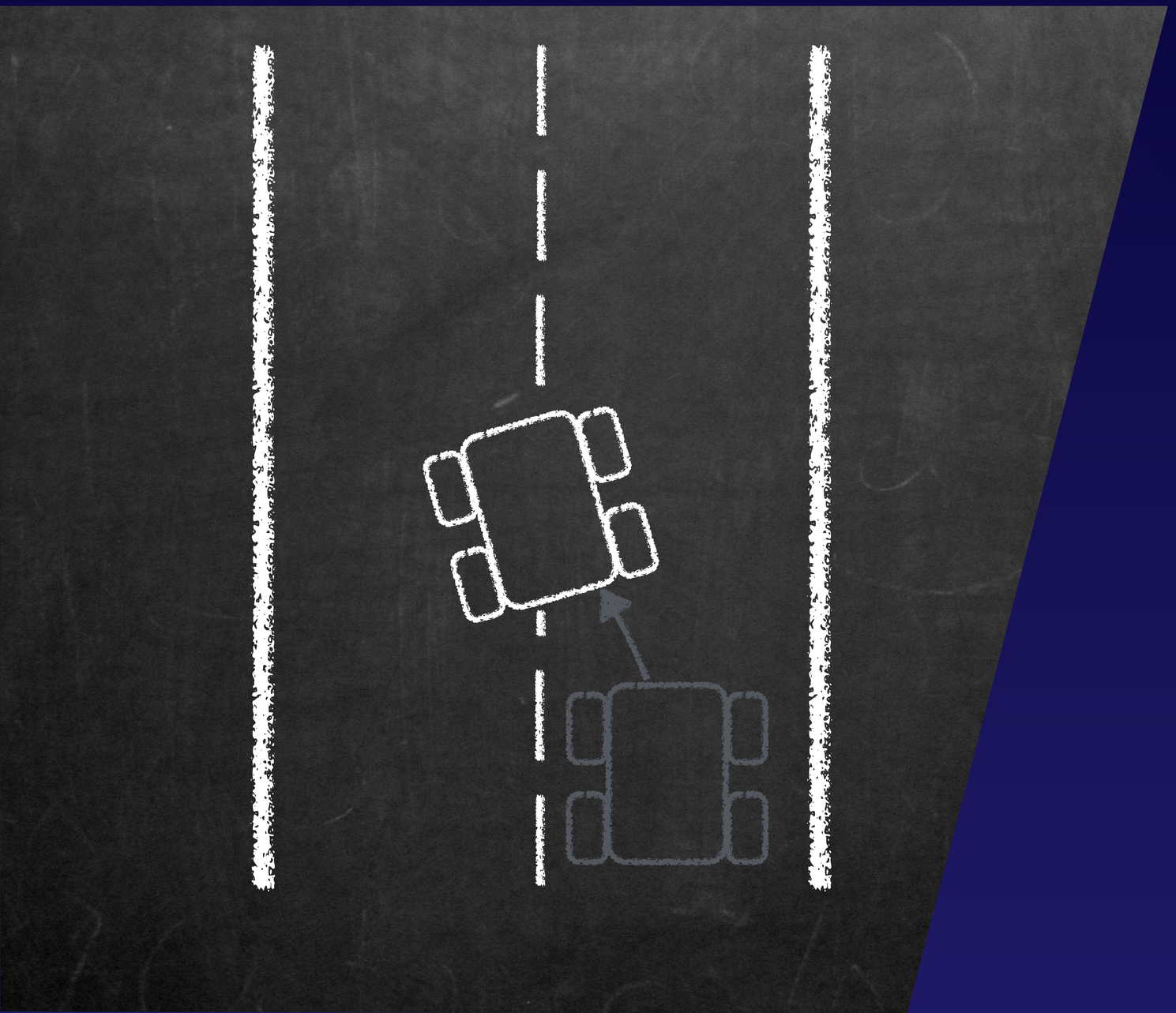
Markov Decision Process

- AGENT
- ENVIRONMENT
- STATE



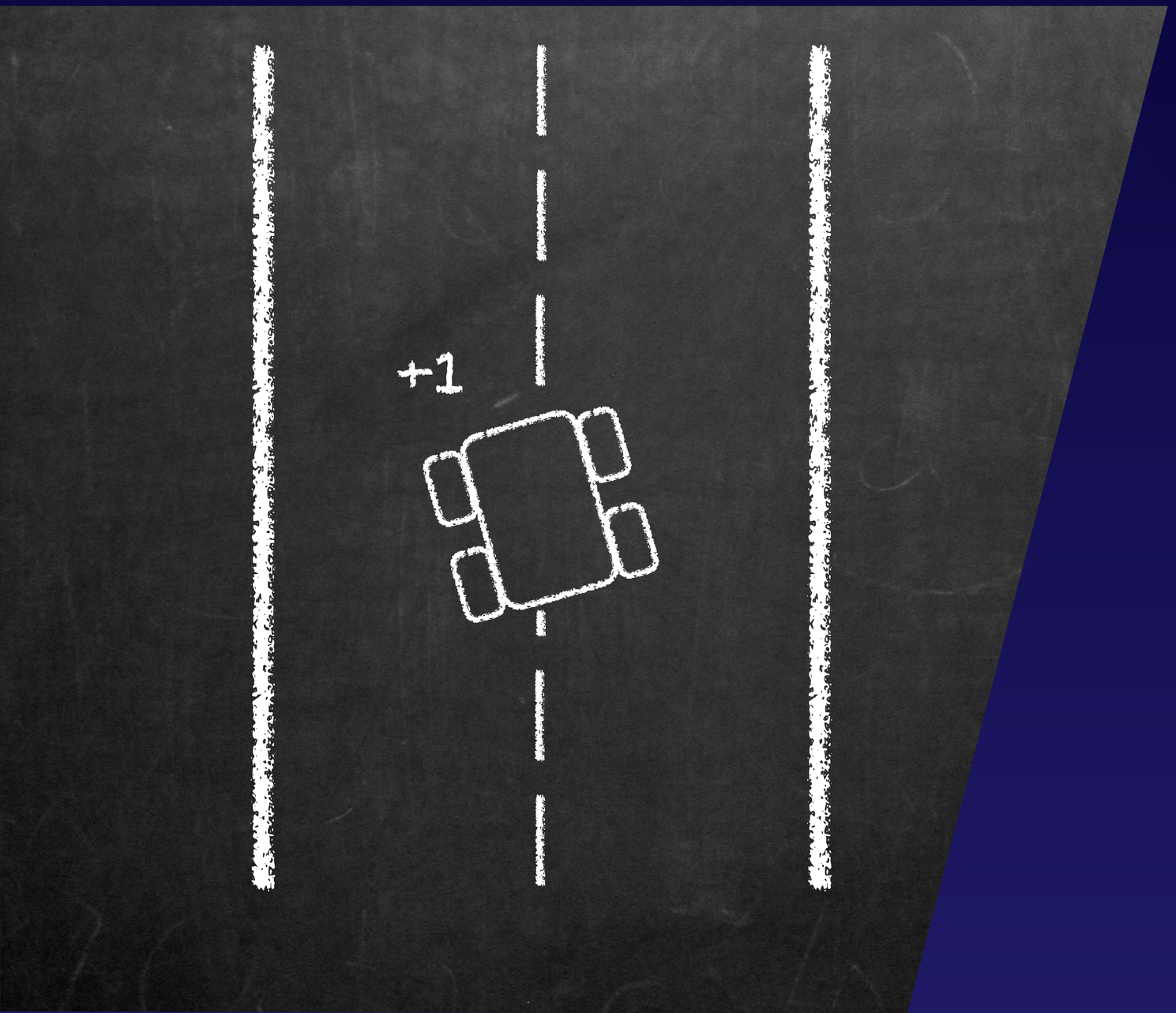
Markov Decision Process

- AGENT
- ENVIRONMENT
- STATE
- OBSERVATION



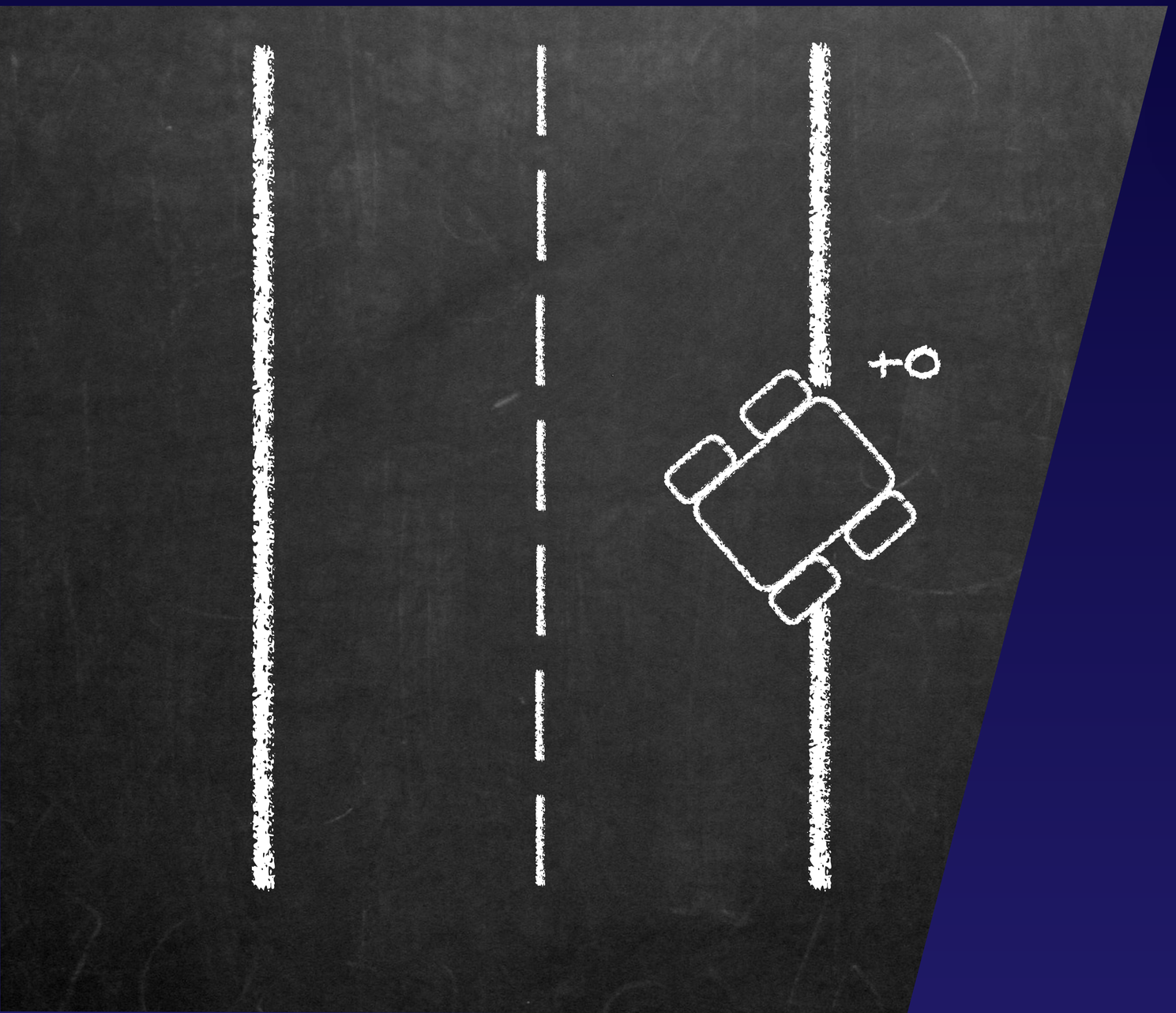
Markov Decision Process

- AGENT
- ENVIRONMENT
- STATE
- OBSERVATION
- ACTION



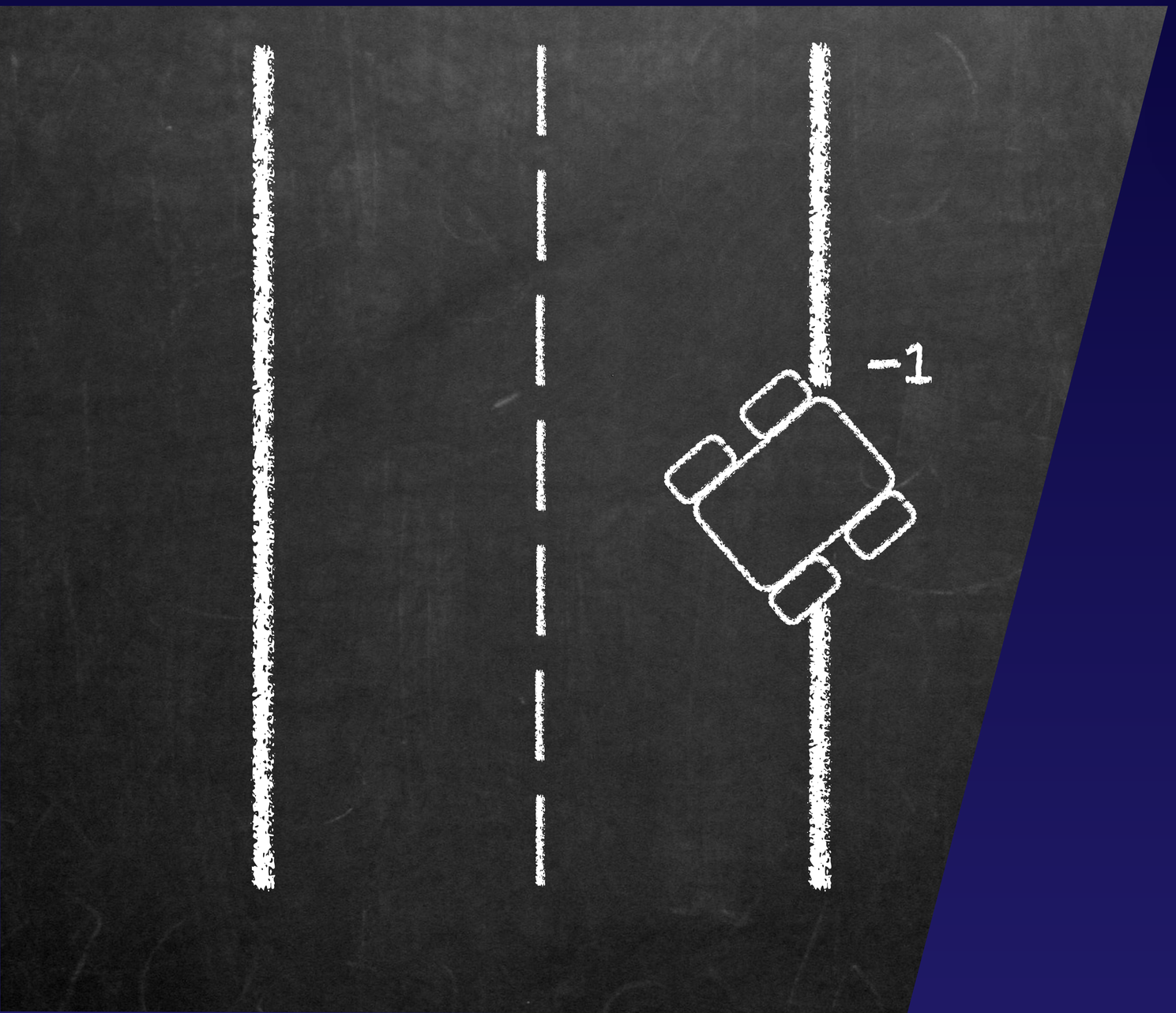
Markov Decision Process

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



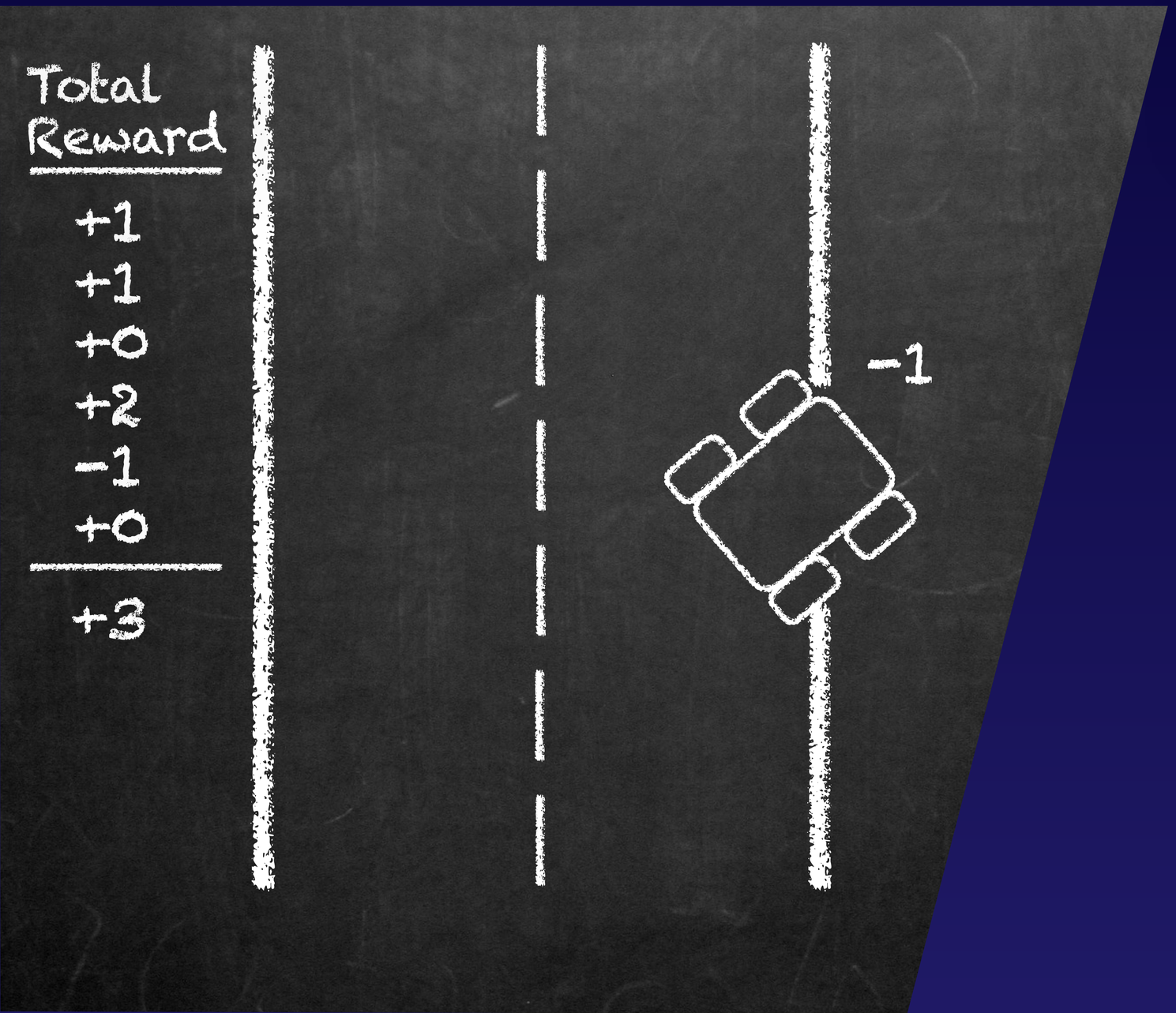
Markov Decision Process

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



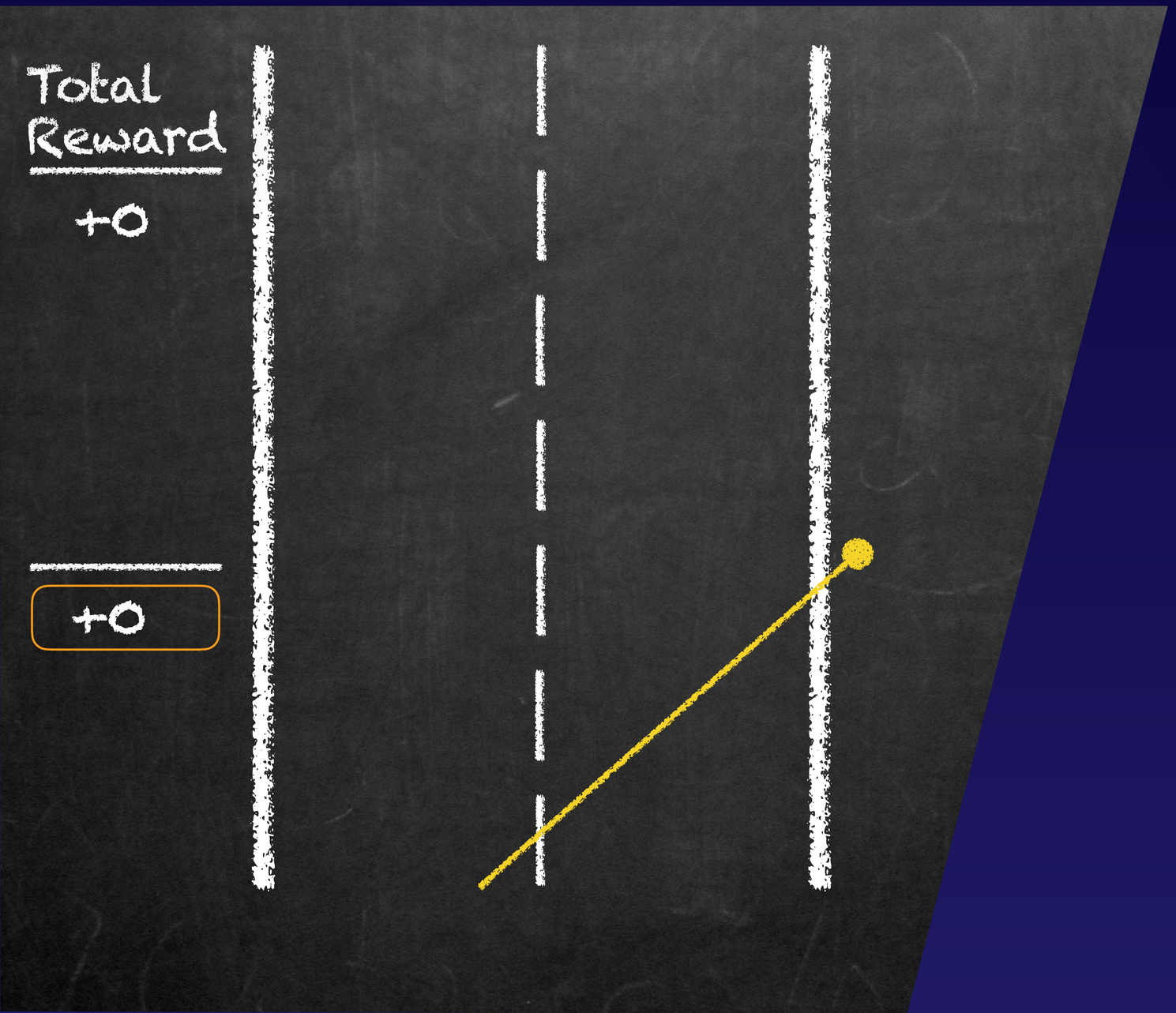
Markov Decision Process

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



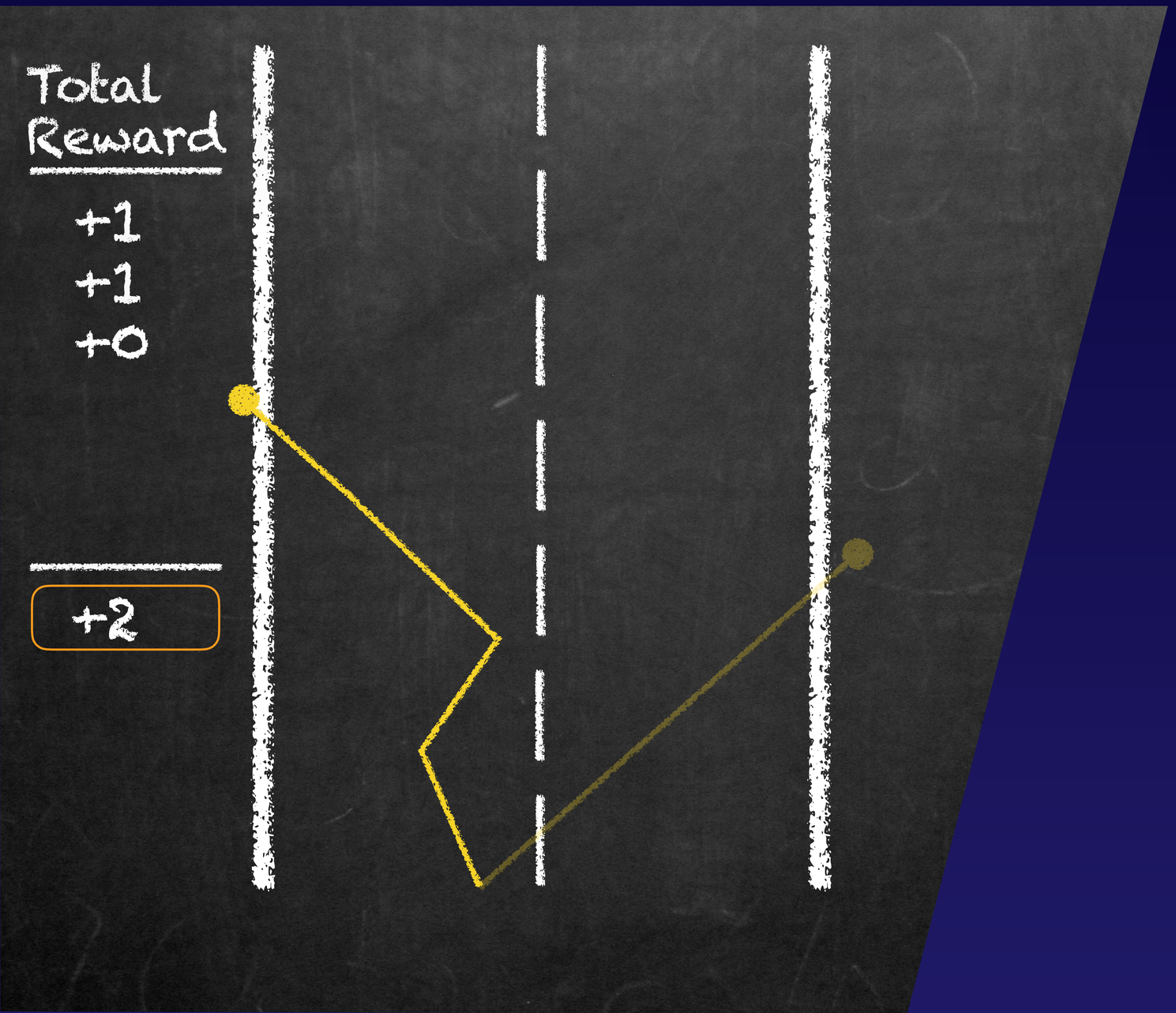
Markov Decision Process

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



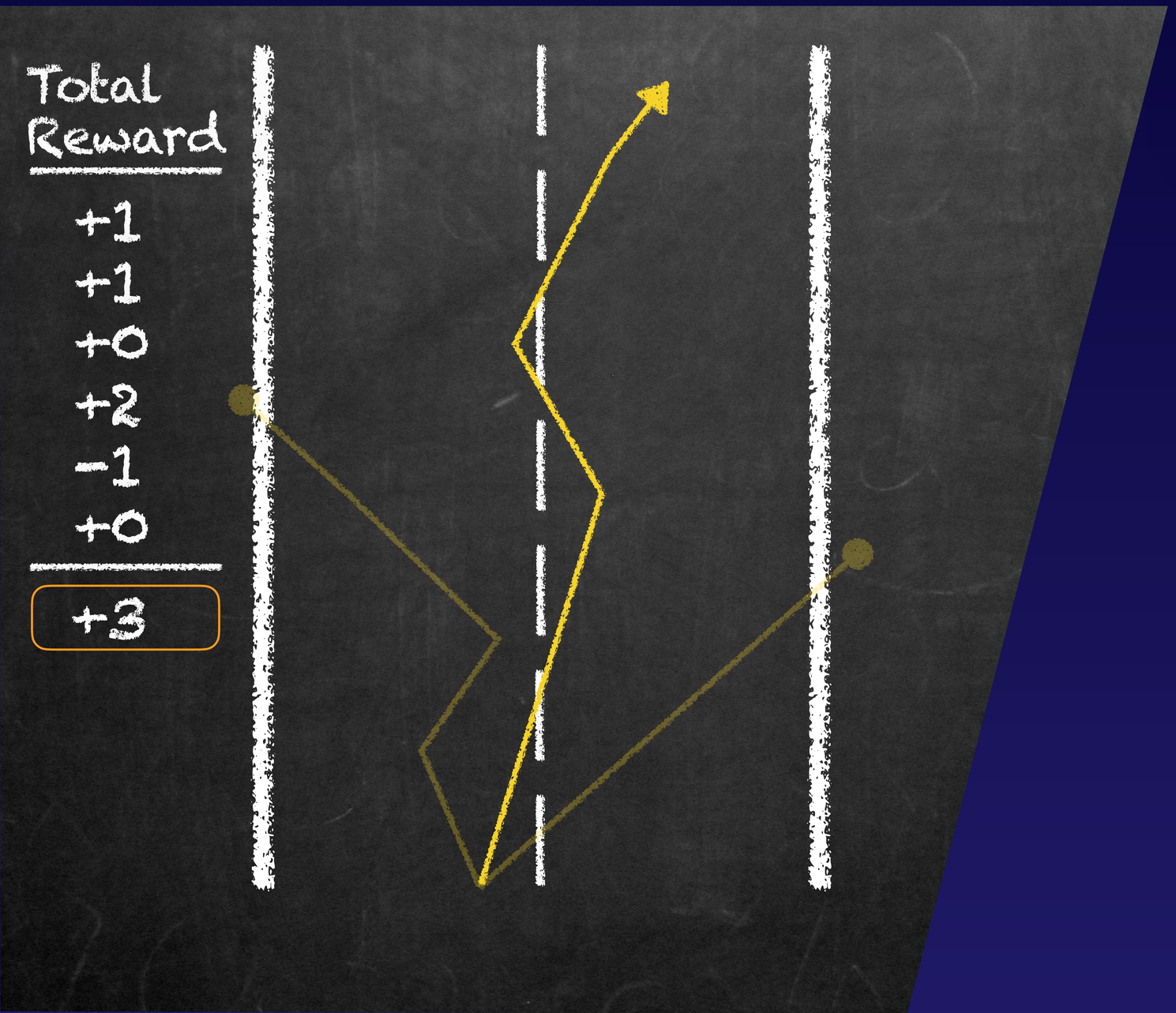
Markov Decision Process

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



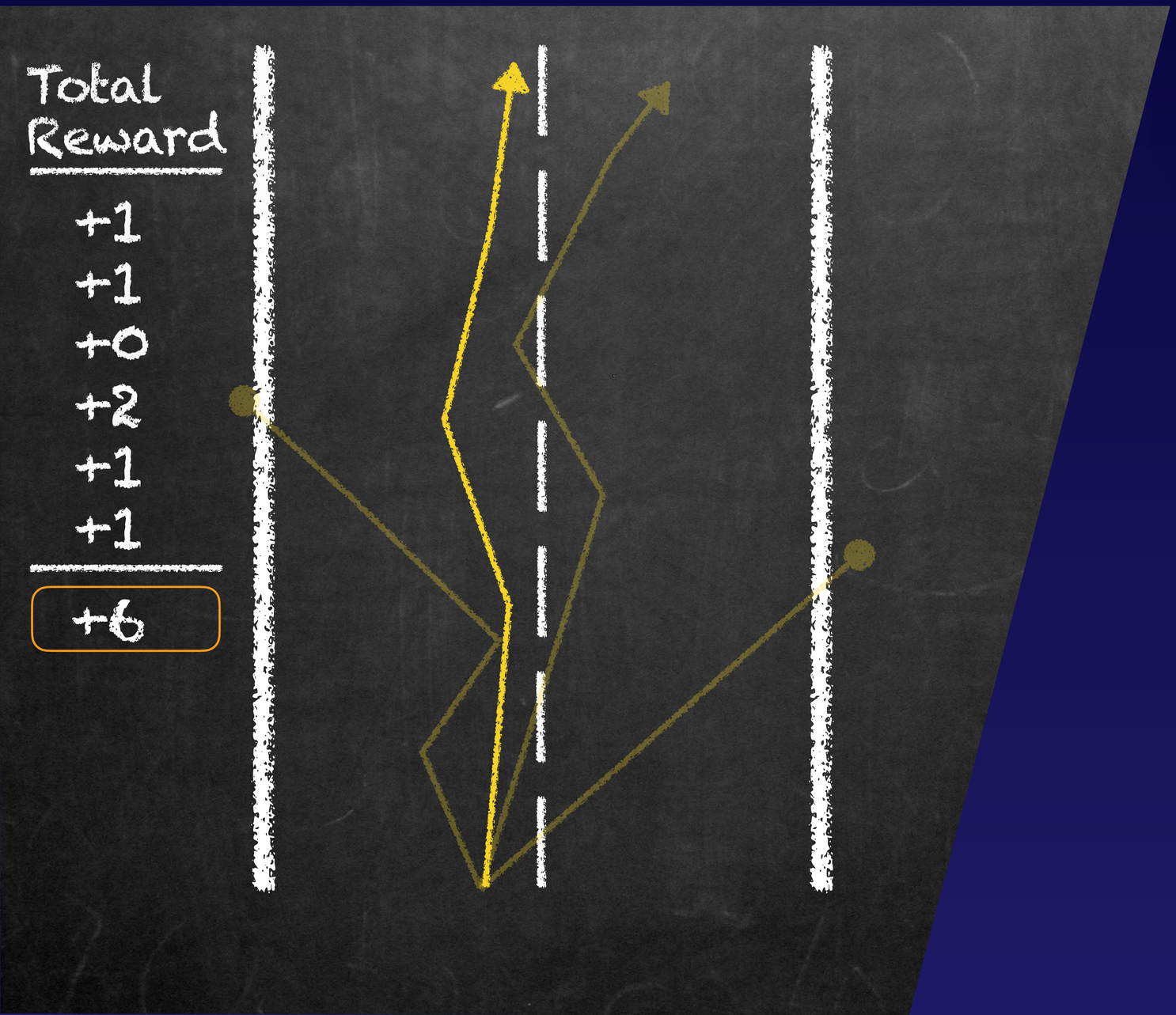
Markov Decision Process

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



Markov Decision Process

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



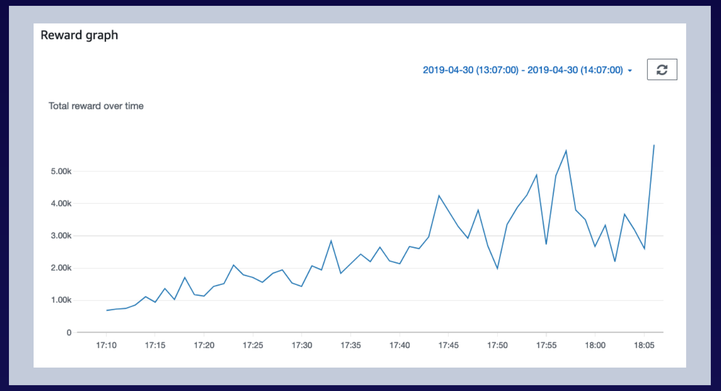
Markov Decision Process

- 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
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

A CLOUD GURU



POLICY

A CLOUD GURU

Dare to be Different

Scott's Slightly Credible

Tips For DeepRacer Glory™

Copy.

Paste.

Race.

A CLOUD GURU

Copy.

Paste.

Race.

A CLOUD GURU

~~Copy.~~

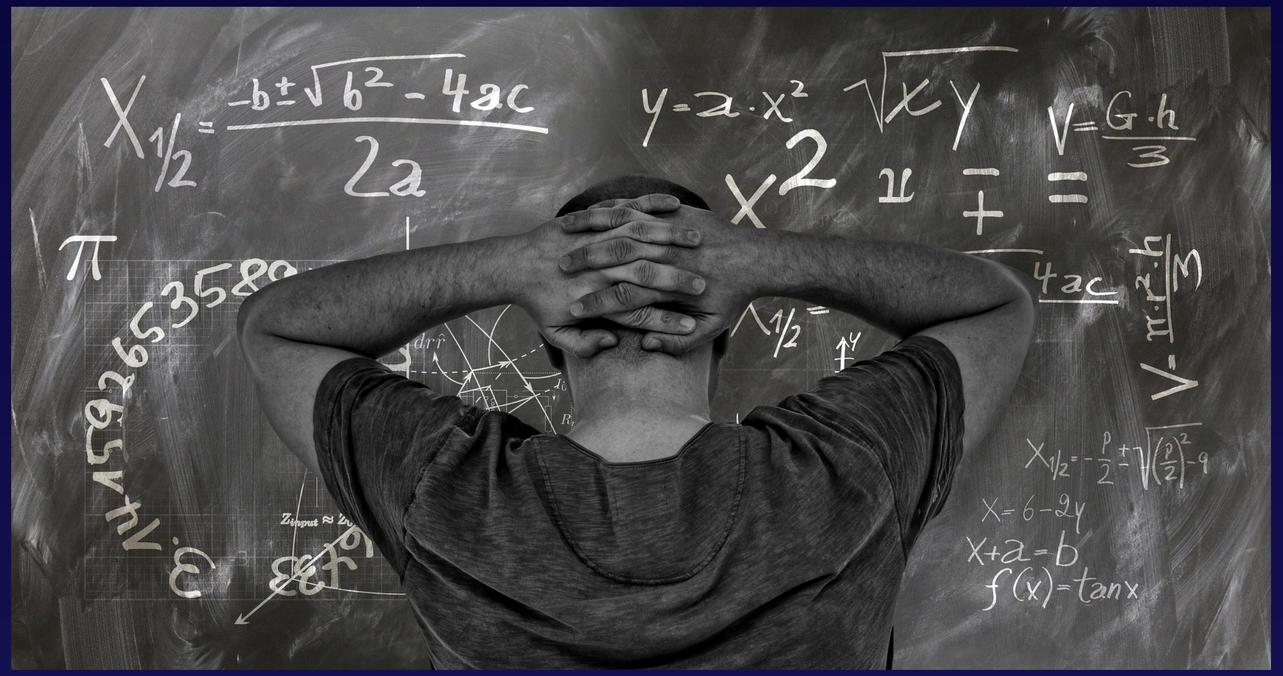
~~Paste.~~

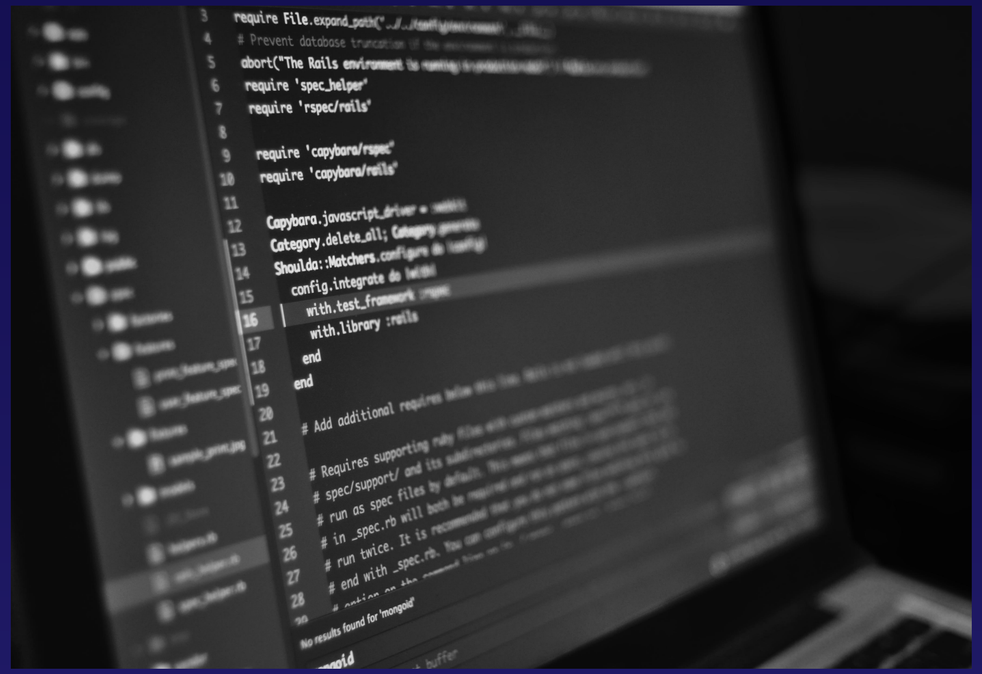
~~Race.~~

Santa Clara Summit - Preliminary Results*

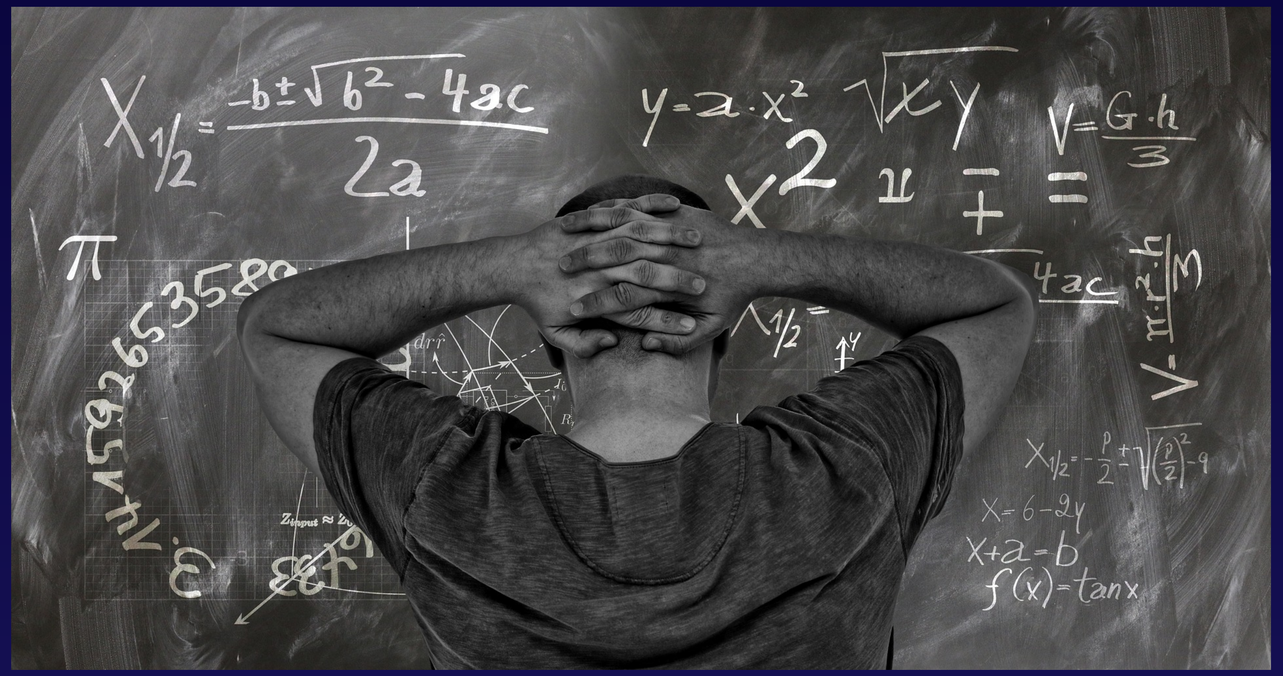
POSITION	NAME	COUNTRY	TIME	POINTS
1st	CLOUD BRIGADE	US	00:10.4	989.57
2nd	RAHUL SHAH	US	00:11.1	988.9
3rd	ADRIAN SARNO	US	00:11.6	988.38
4th	A CLOUD GURU	US	00:11.7	988.25
5th	AAMIR	US	00:12.0	987.96
6th	CPRIME TIME	US	00:14.3	985.74
7th	DEEPAK	US	00:15.2	984.81
8th	NORAEK	US	00:15.5	984.54
9th	PRAVEEN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

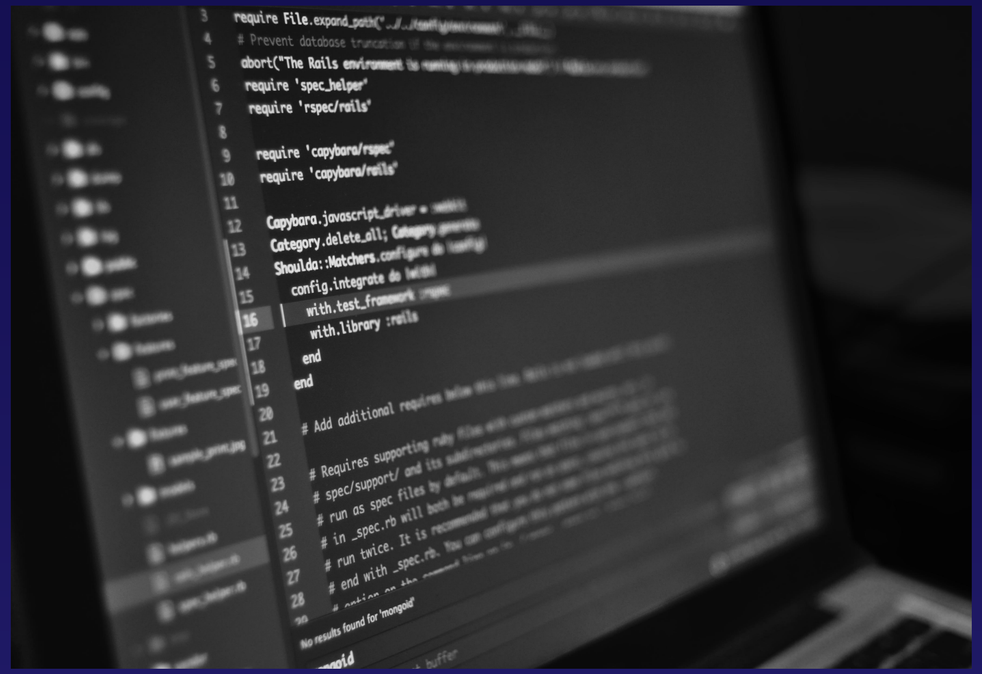
DO SOMETHING DIFFERENT

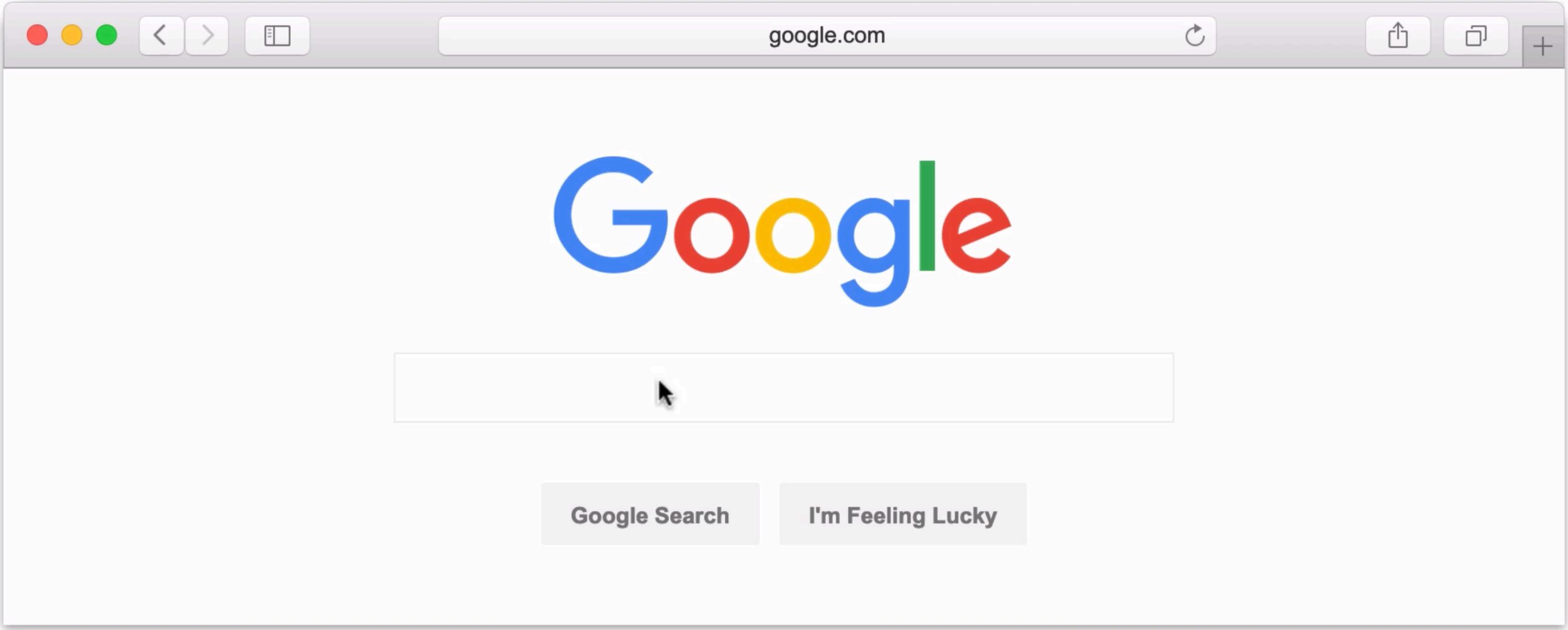


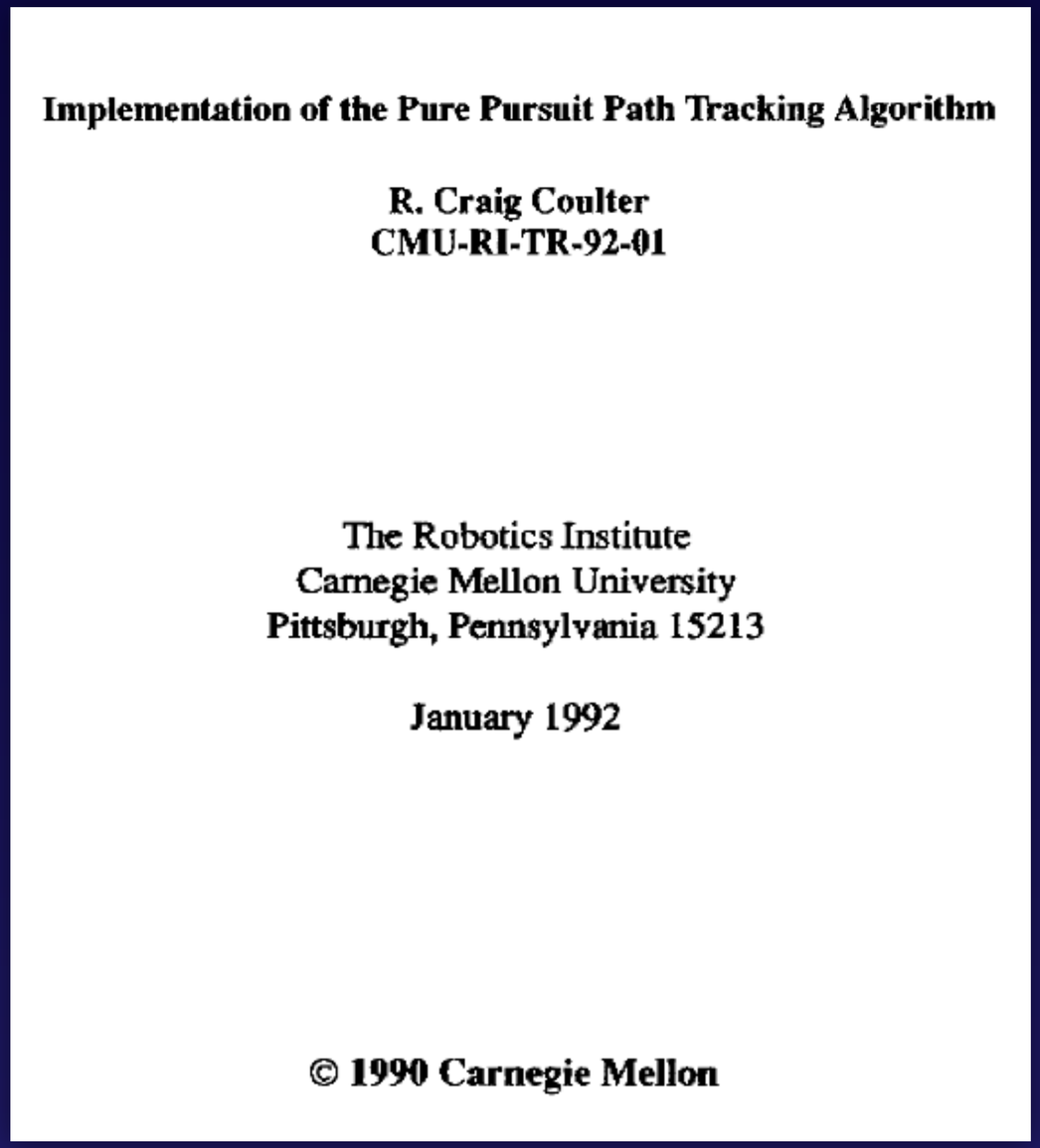


DO SOMETHING DIFFERENT









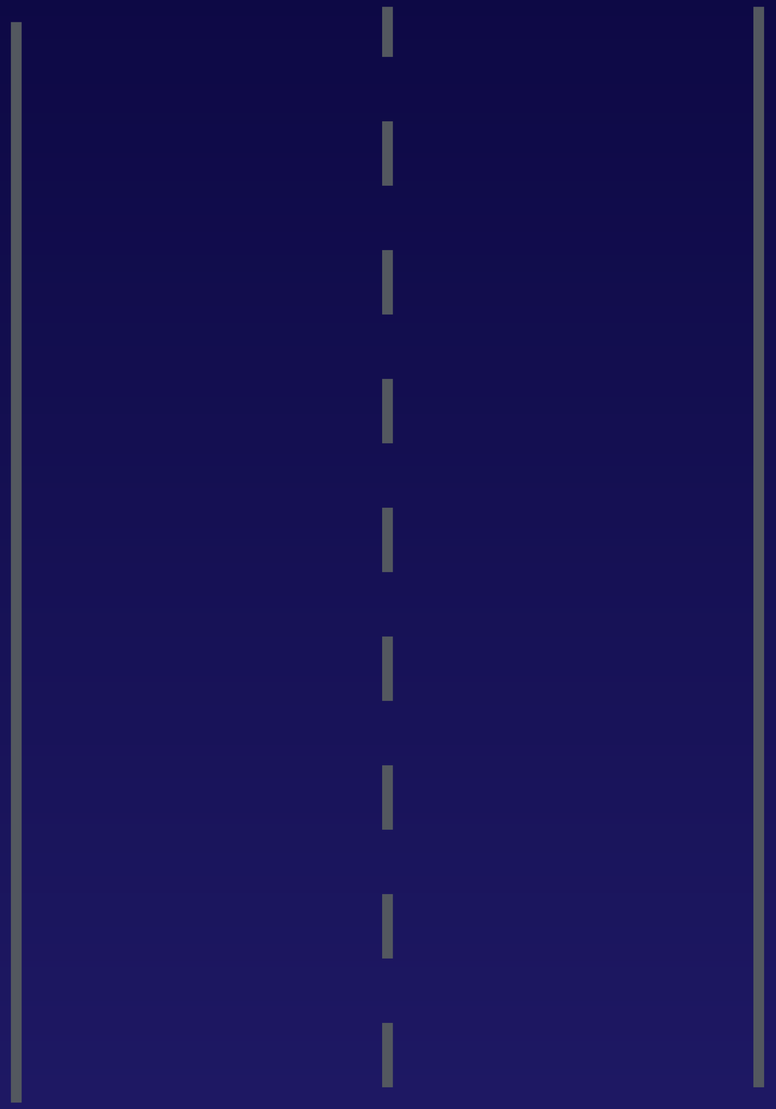
Implementation of the Pure Pursuit Path Tracking Algorithm

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

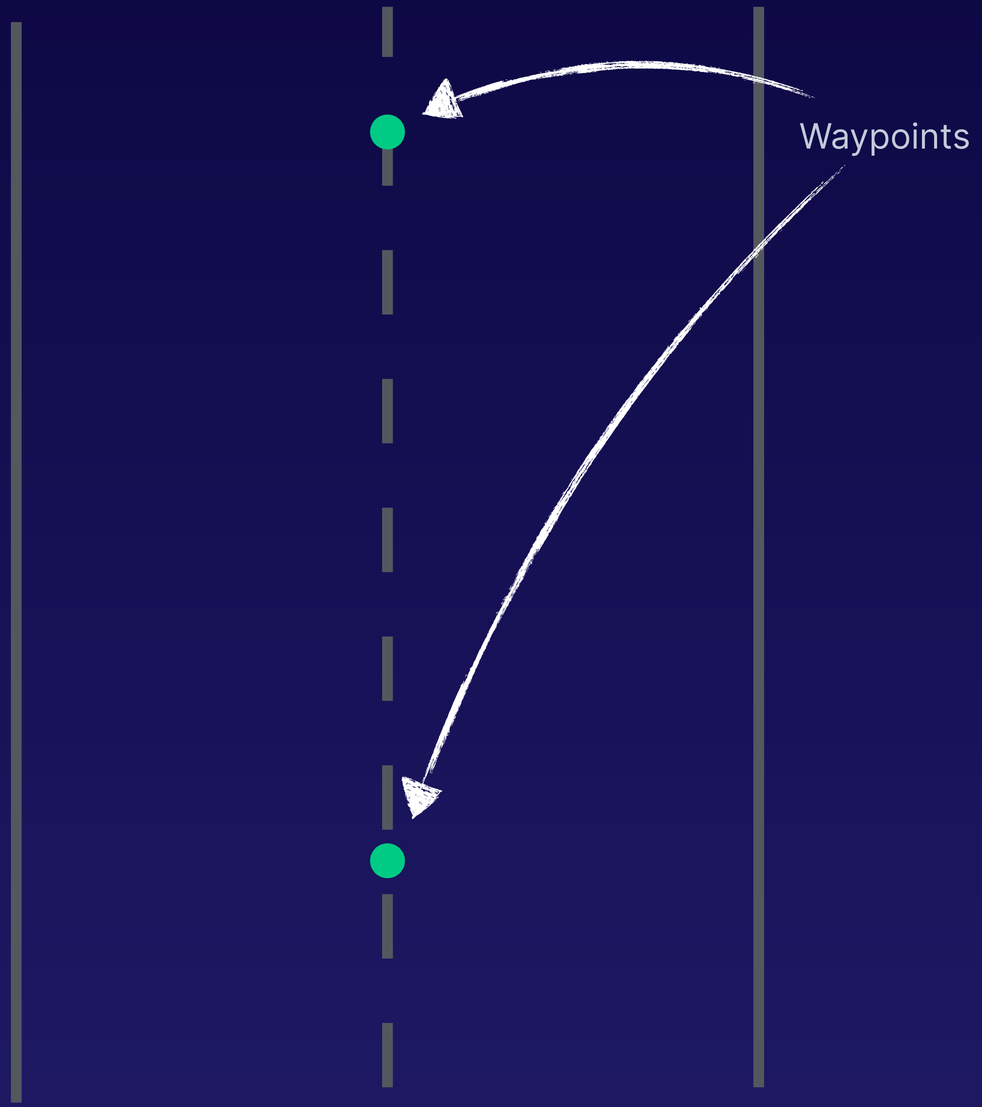
The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

January 1992

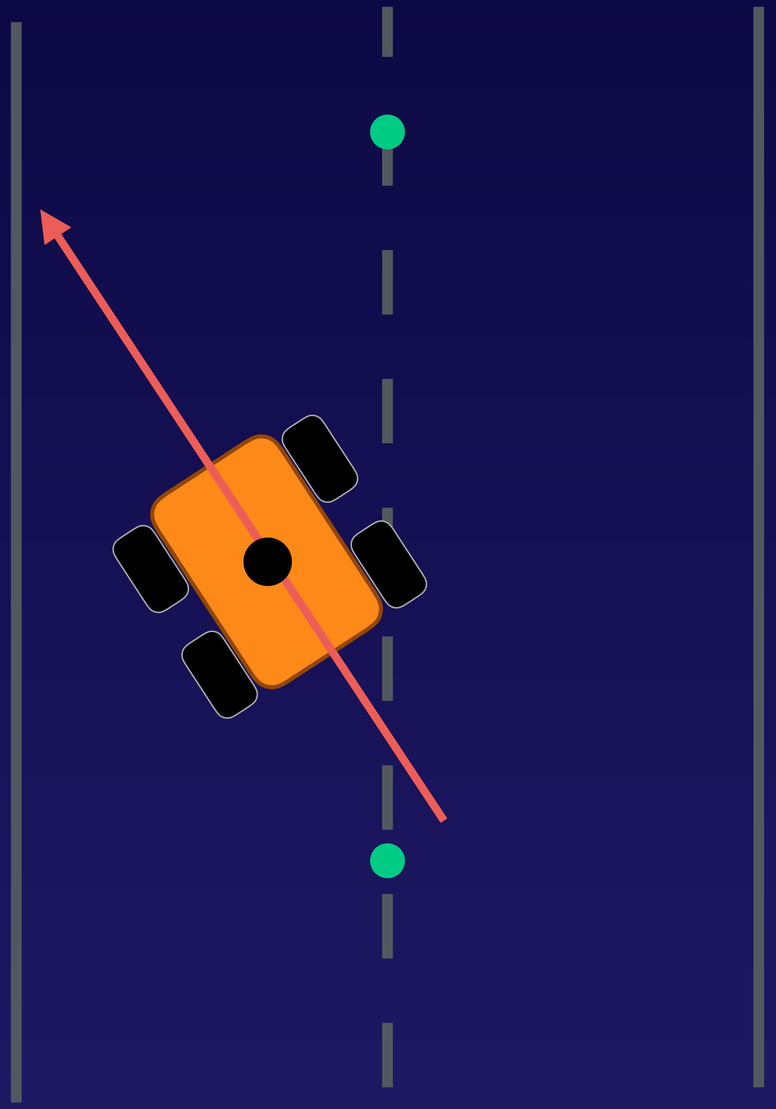

```
rabbit = [0,0]
pointing = [0,0]

# 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))
```



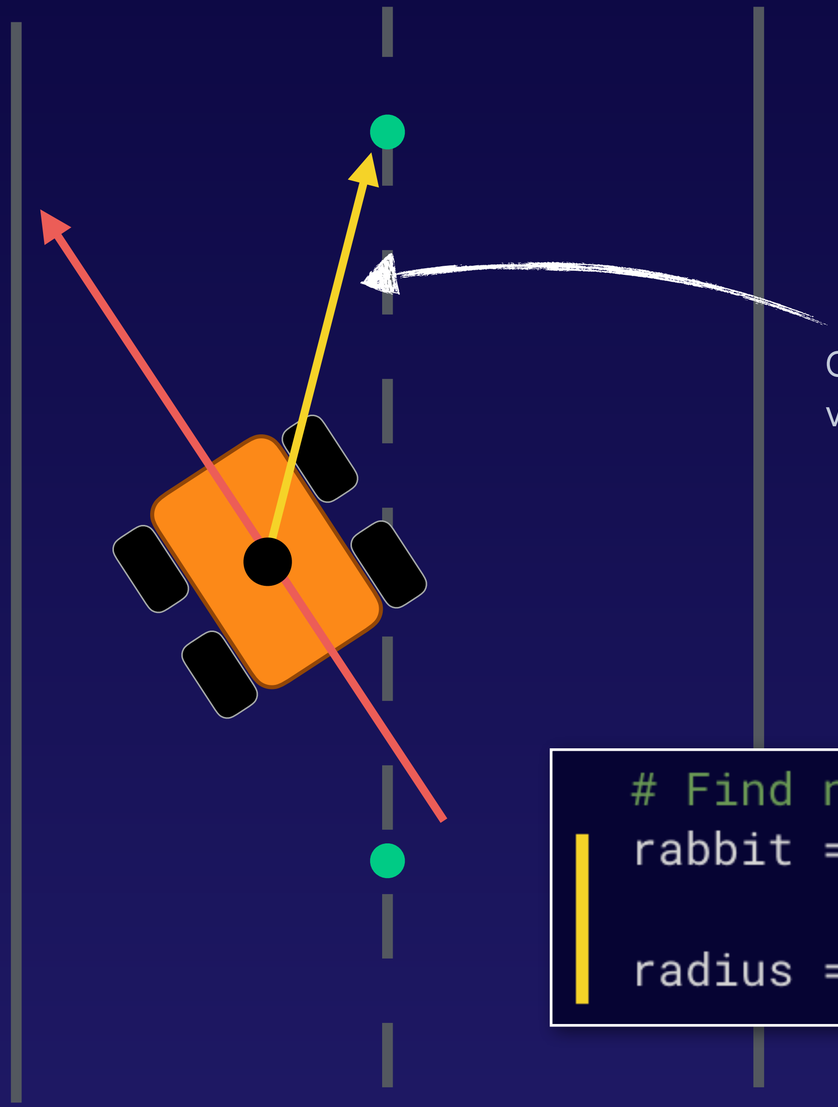
```
rabbit = [0,0]
pointing = [0,0]

# 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))
```

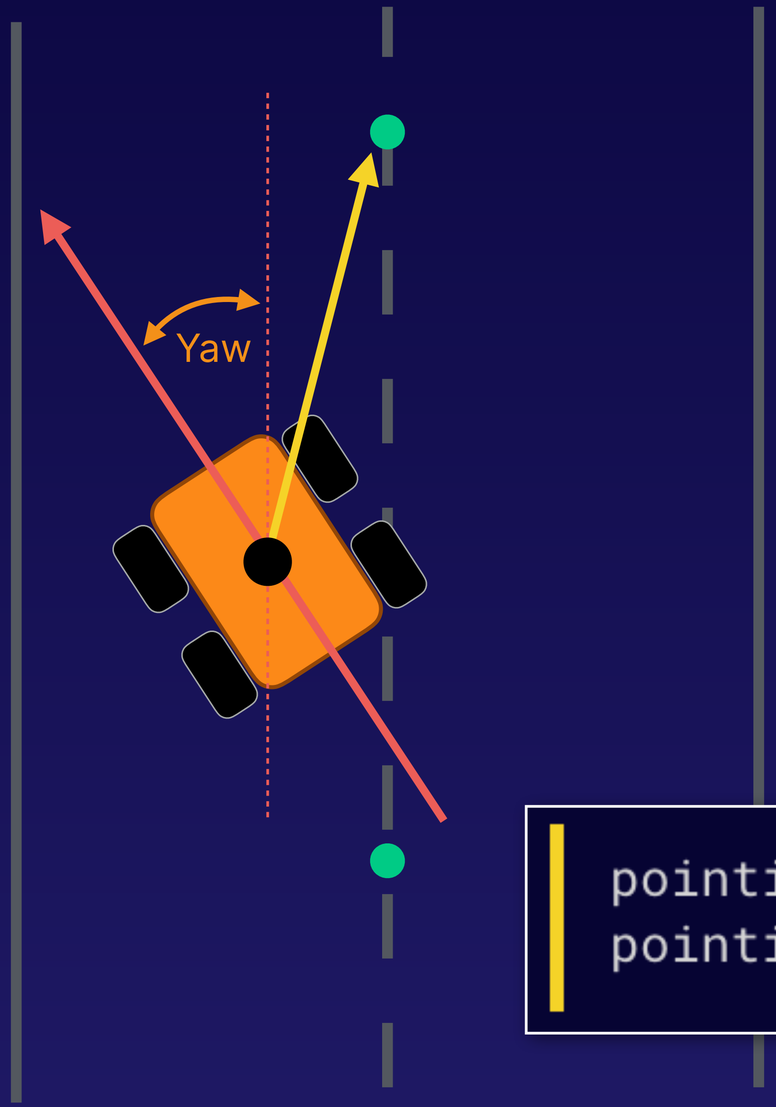


```
rabbit = [0,0]
pointing = [0,0]

# 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))
```



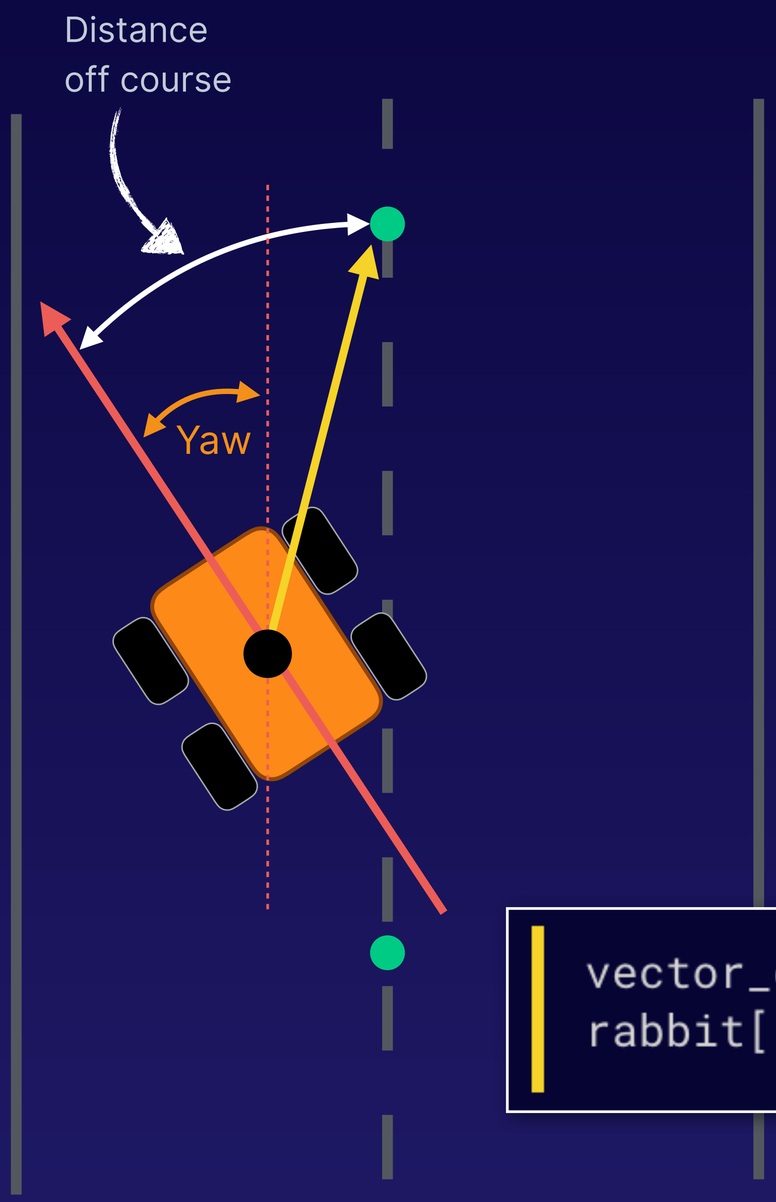

```
rabbit = [0,0]
pointing = [0,0]

# 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
```


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

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

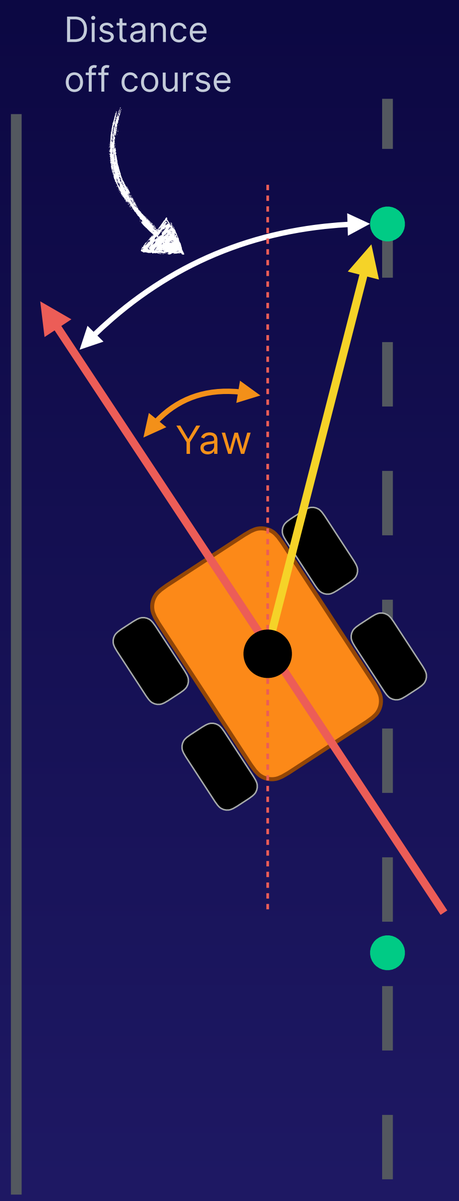

```
pointing[0] = x + (radius * math.cos(car_orientation))
pointing[1] = y + (radius * math.sin(car_orientation))
```




```
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.
```


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

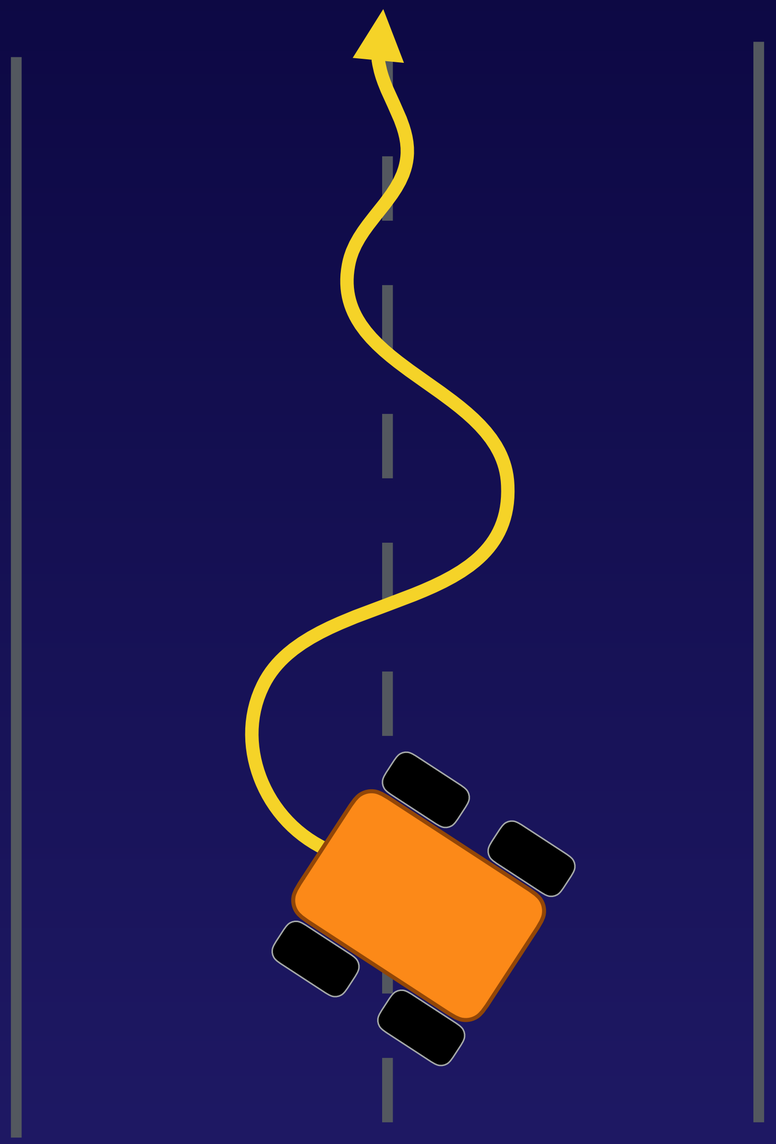
A CLOUD GURU



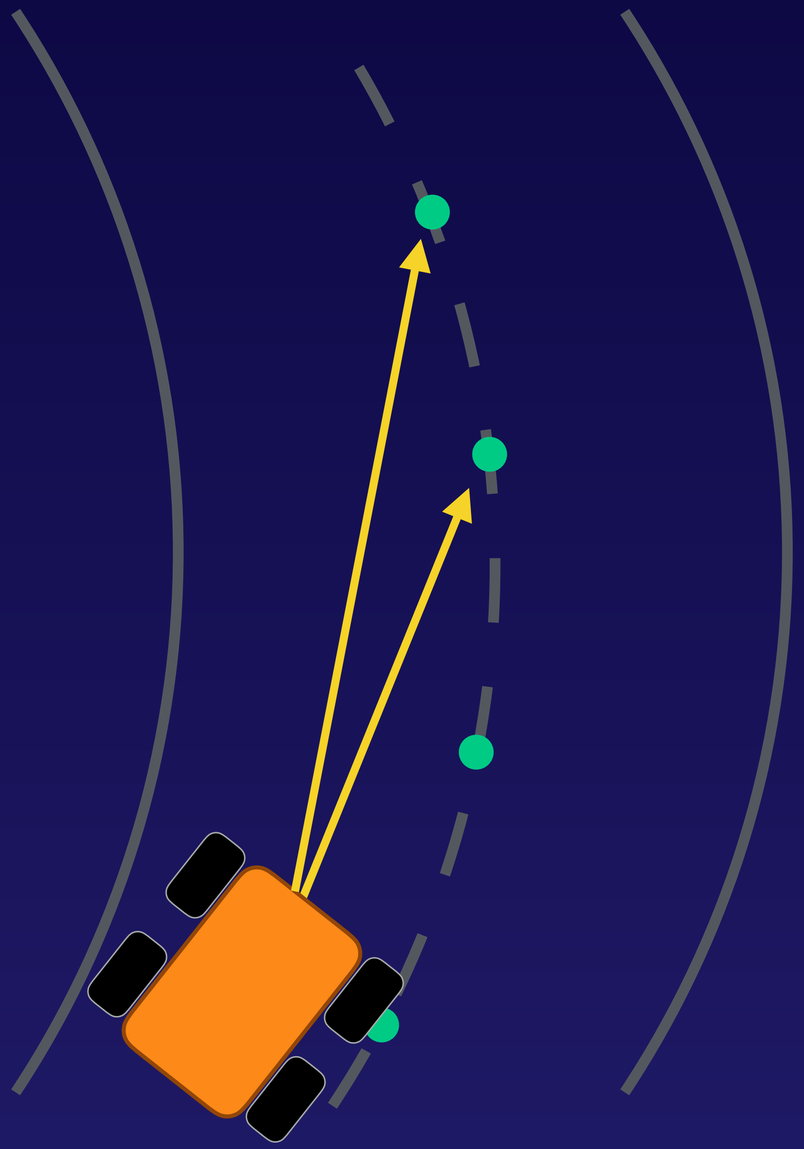

```
rabbit = [0,0]
pointing = [0,0]

# 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))
```

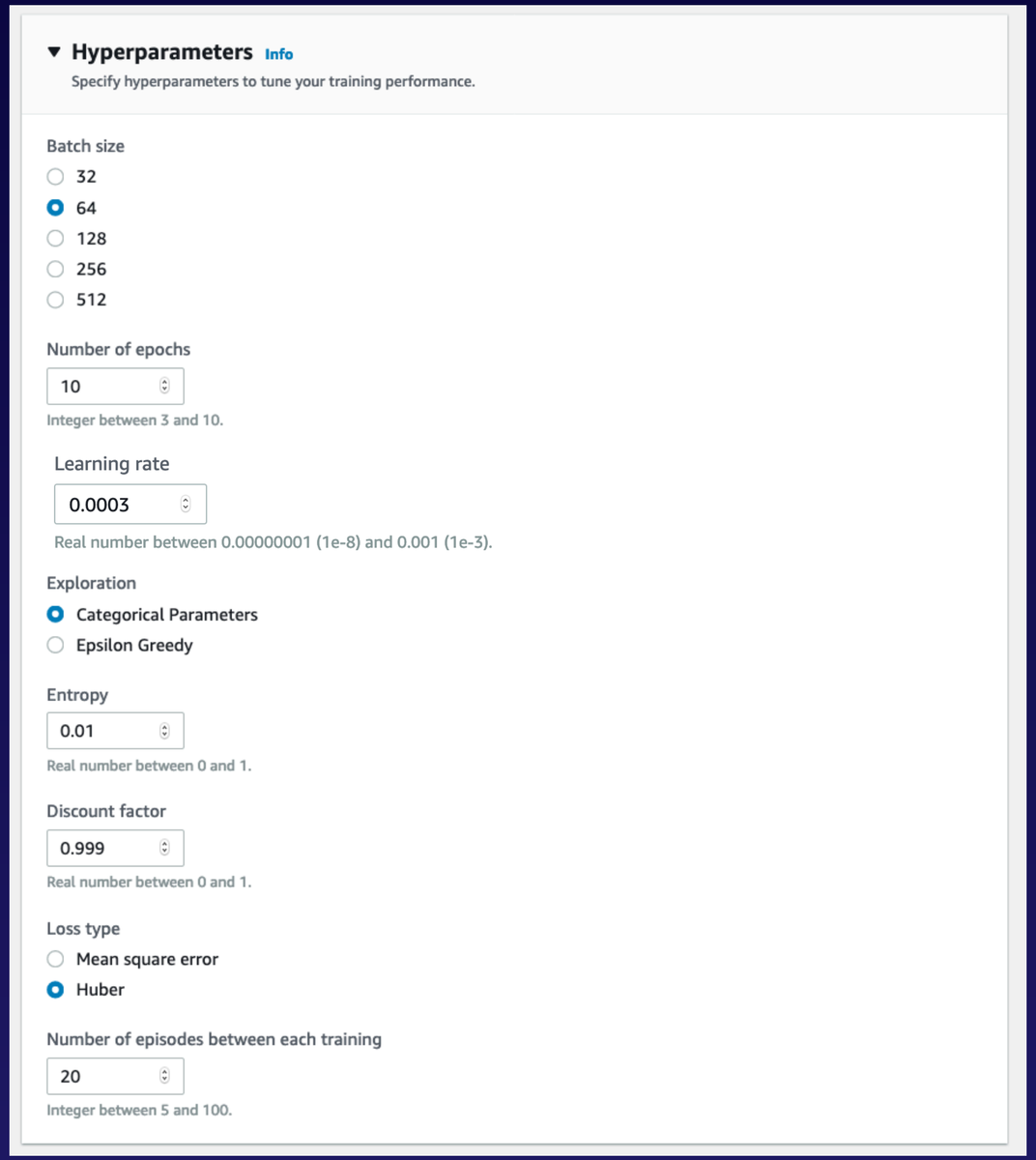
A CLOUD GURU

Dare to be Different

Scott's Slightly Credible

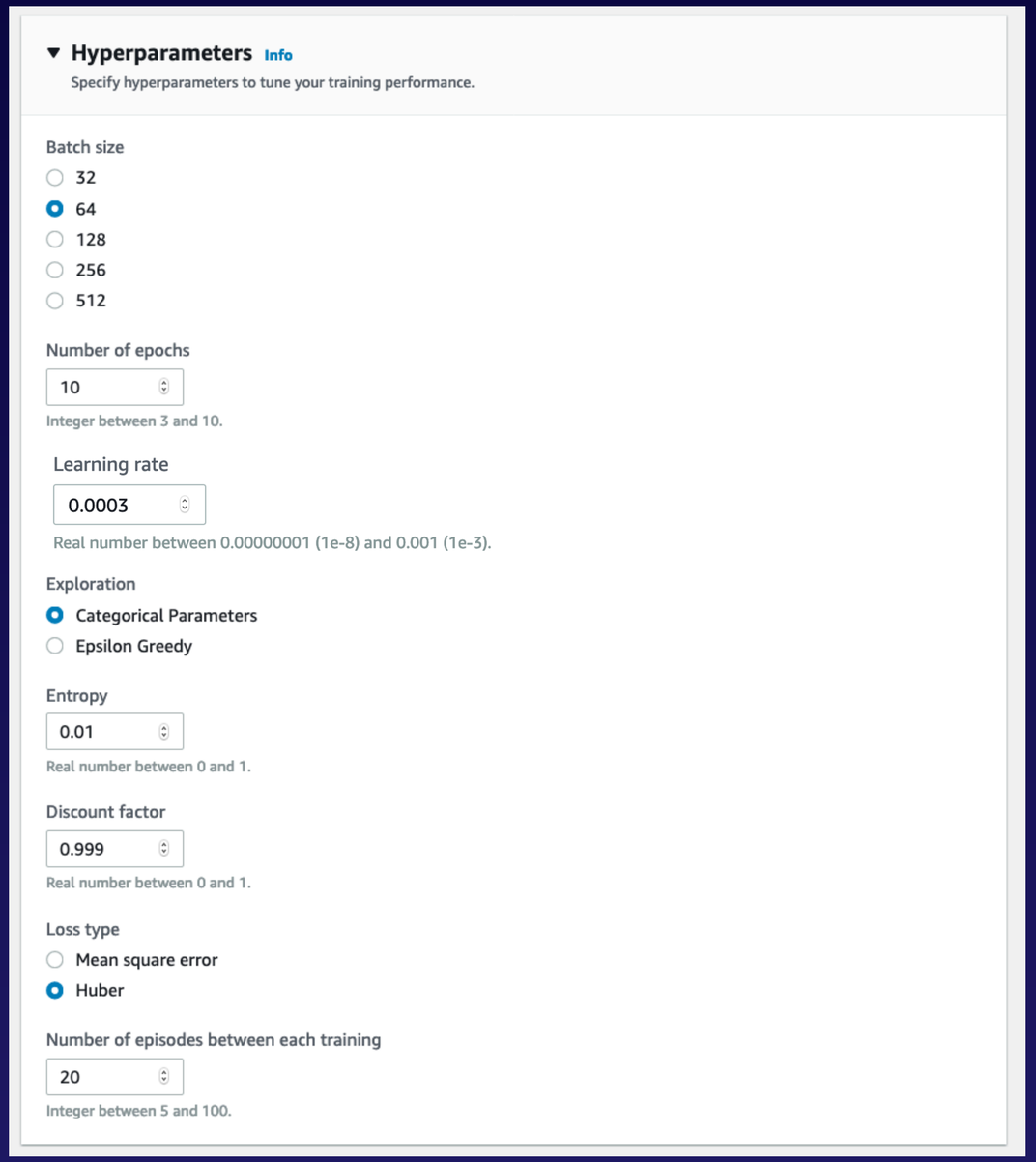
Tips For DeepRacer Glory™

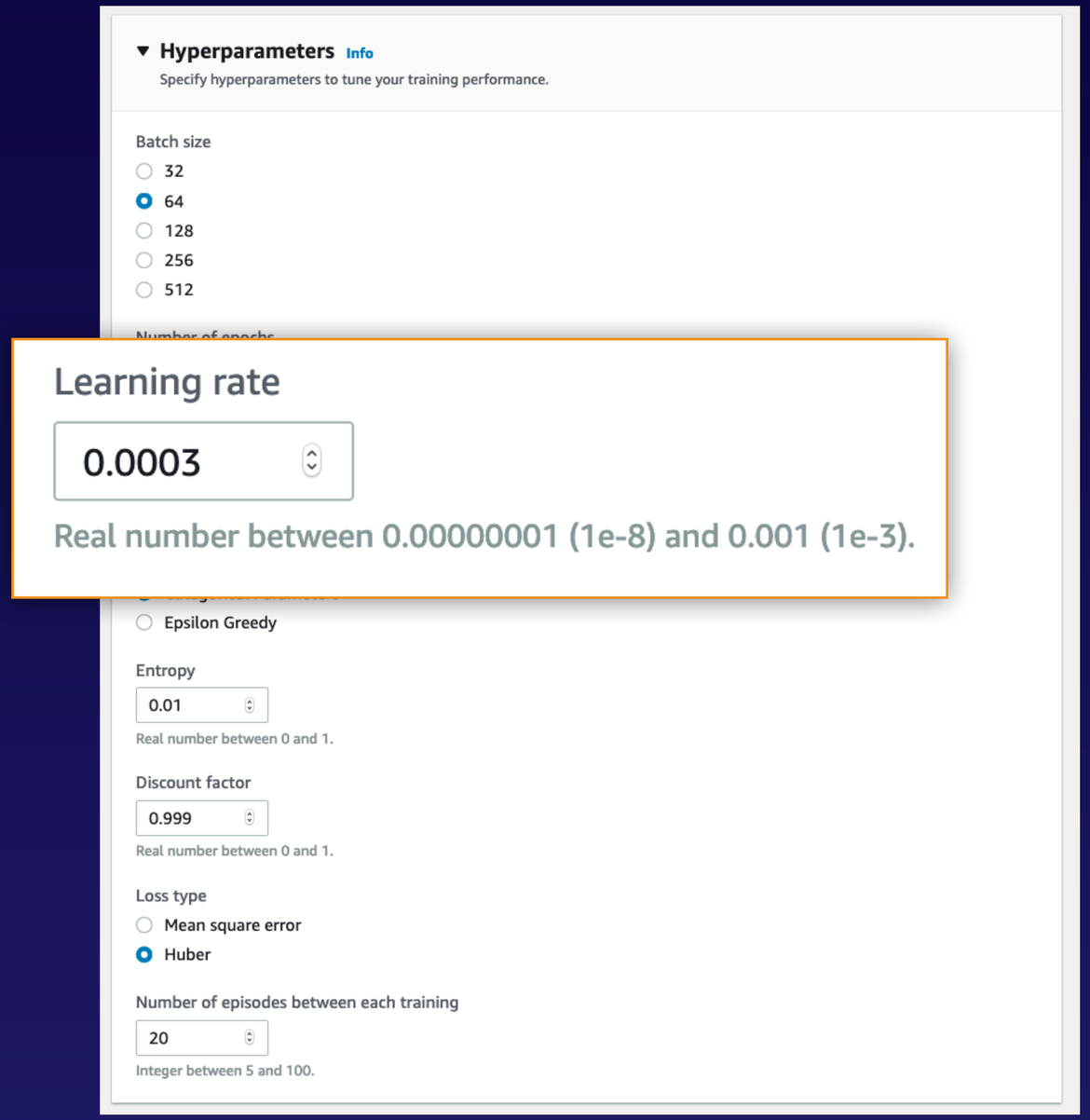
Turn the Knobs


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.


NORMAL CHANNEL
GAIN
BASS
TREBLE
BOOST SELECT
ON
BOOST CHANNEL
GAIN
CONTOUR
LOW
HIGH
PRESENCE
-15
+15
VOLUME
MASTER REVERB
FOOT SWITCH
HEADPHONE OUTPUT
LINE OUT
Marshall

A CLOUD GURU


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
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.

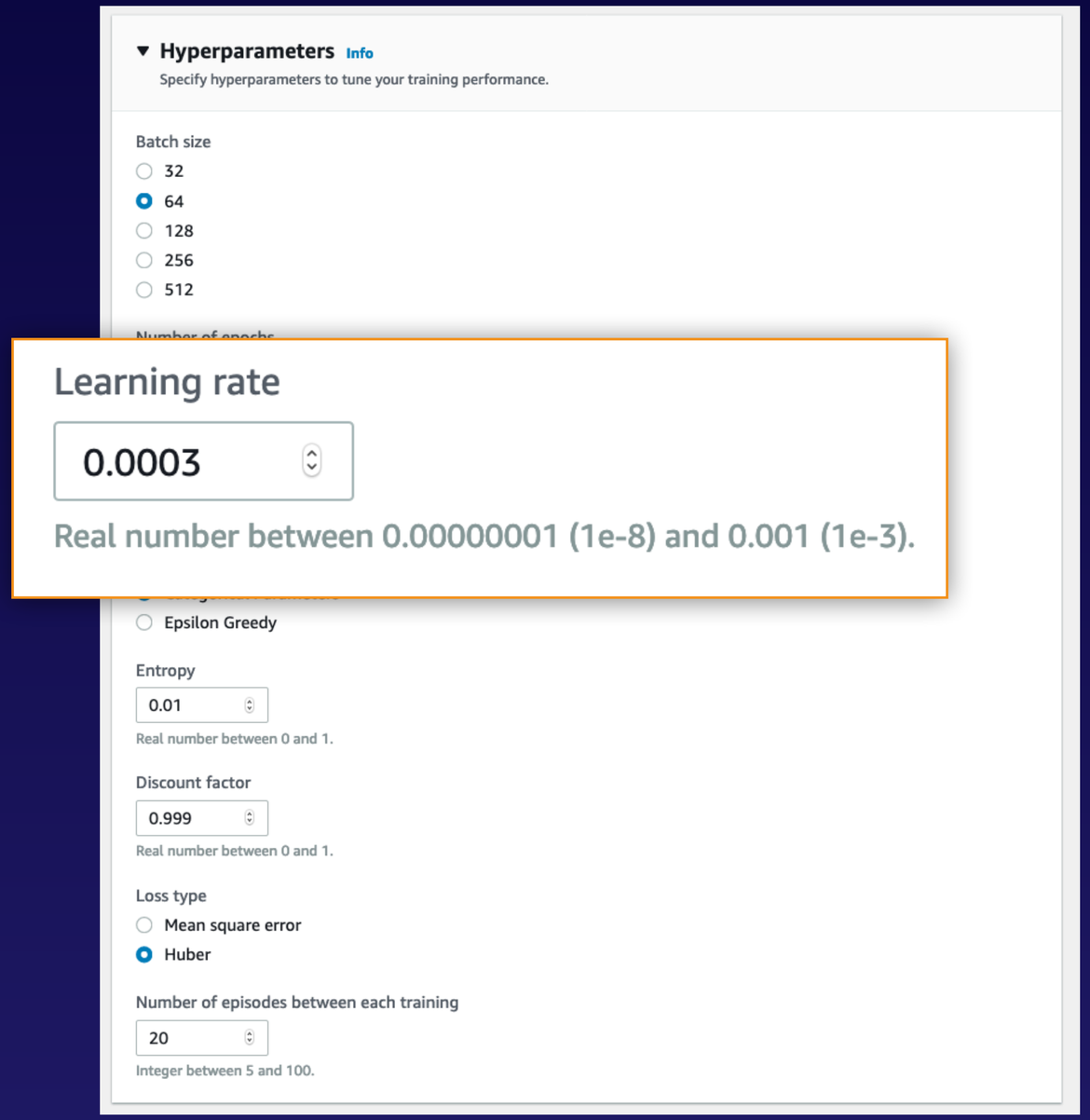
Given a cost function J defined as:

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

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 α.



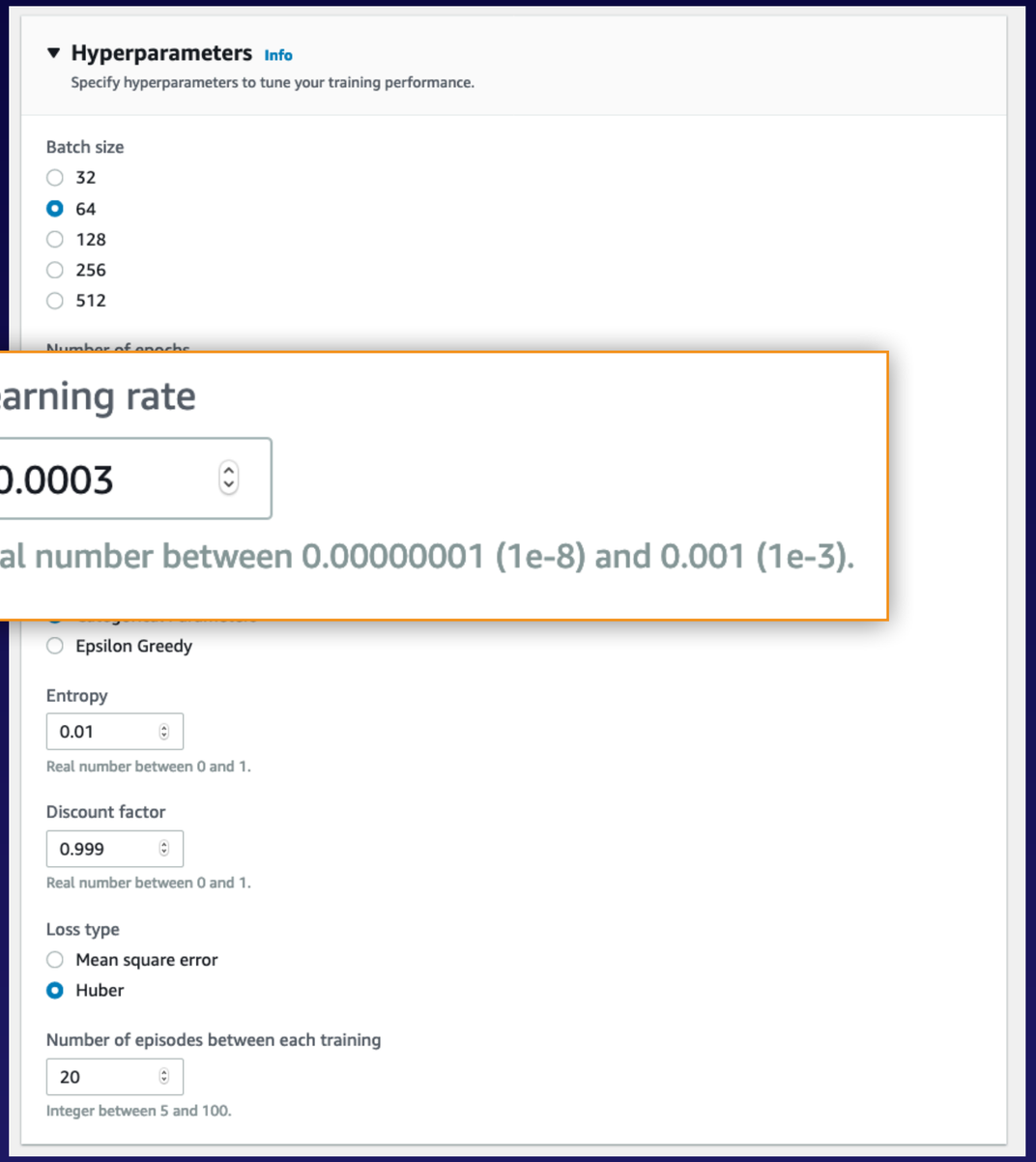
Given a cost function J defined as:

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

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 α.



A CLOUD GURU

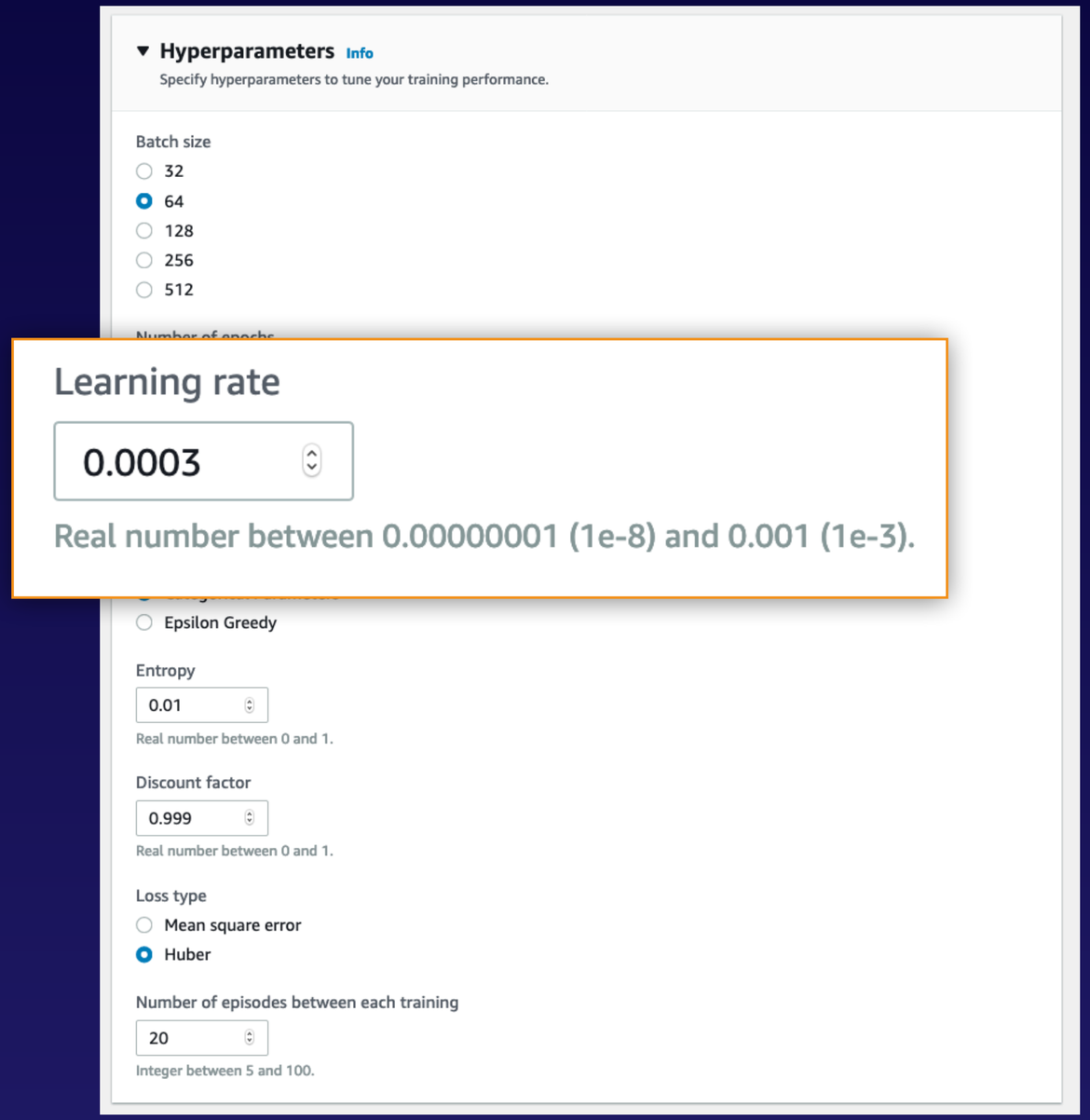
Scott's Slightly Credible

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
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.


CLEVELAND
BROWNS

A CLOUD GURU

Larger Learning Rate

Smaller Learning Rate

A CLOUD GURU

Scott's Slightly Credible

Tips For DeepRacer Glory™

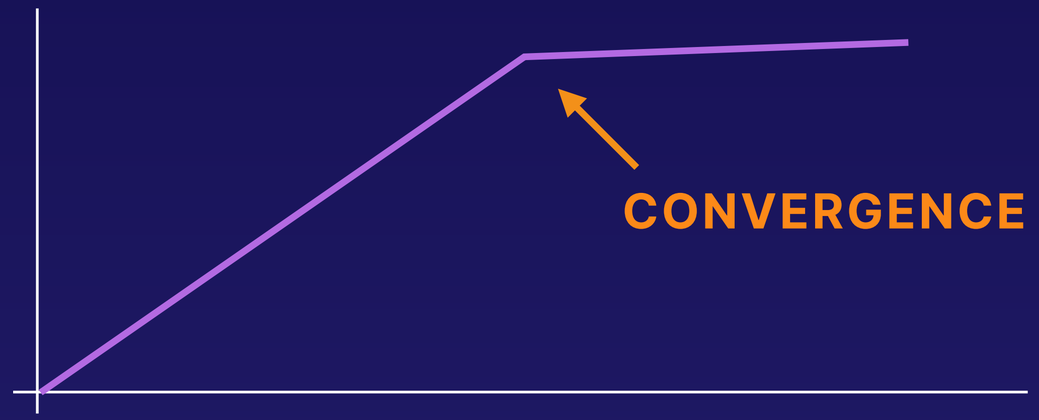
Dare to be Different

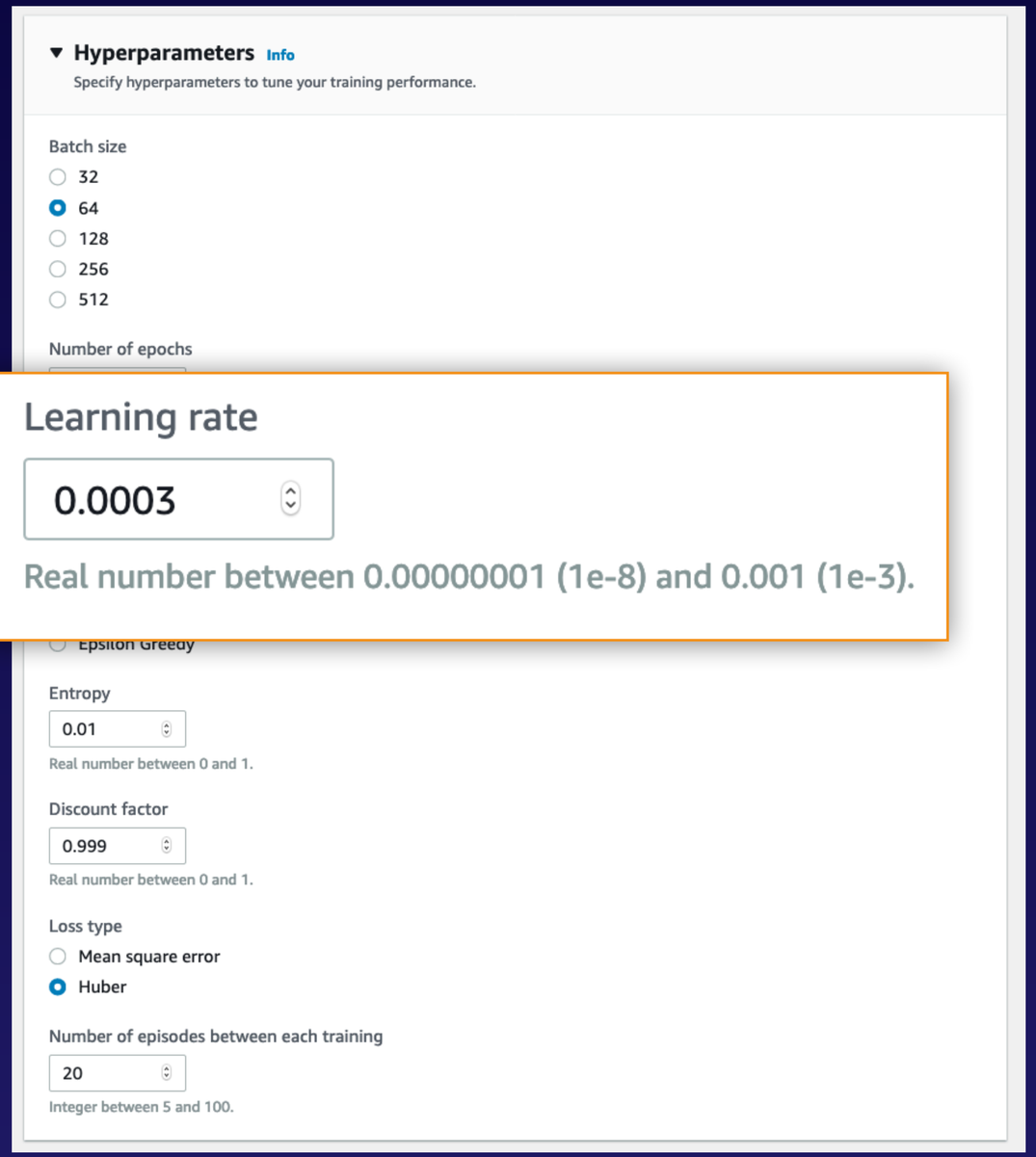
Turn the Knobs

Don't Fear the Greek

Iterate. Iterate. Iterate.

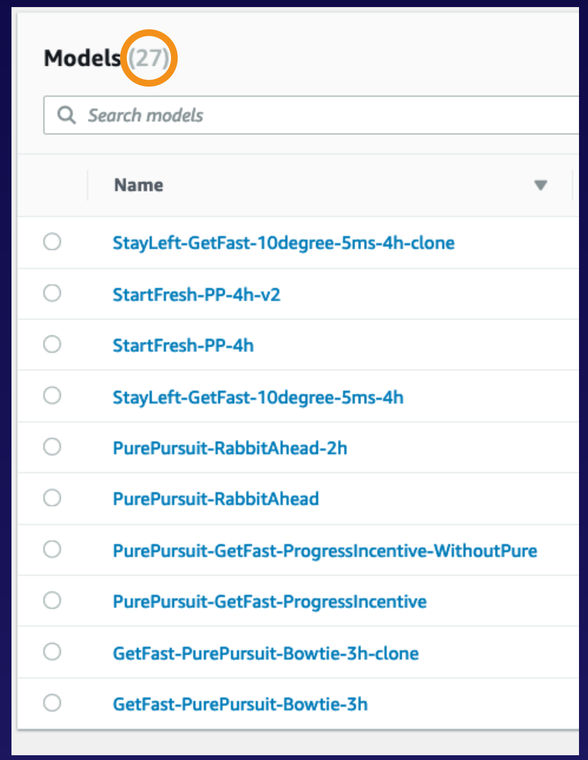
Round	Learning Rate	Coverged At
1	0.0010	50 points
2	0.0005	75 points
3	0.0003	100 points
4	0.0001	104 points



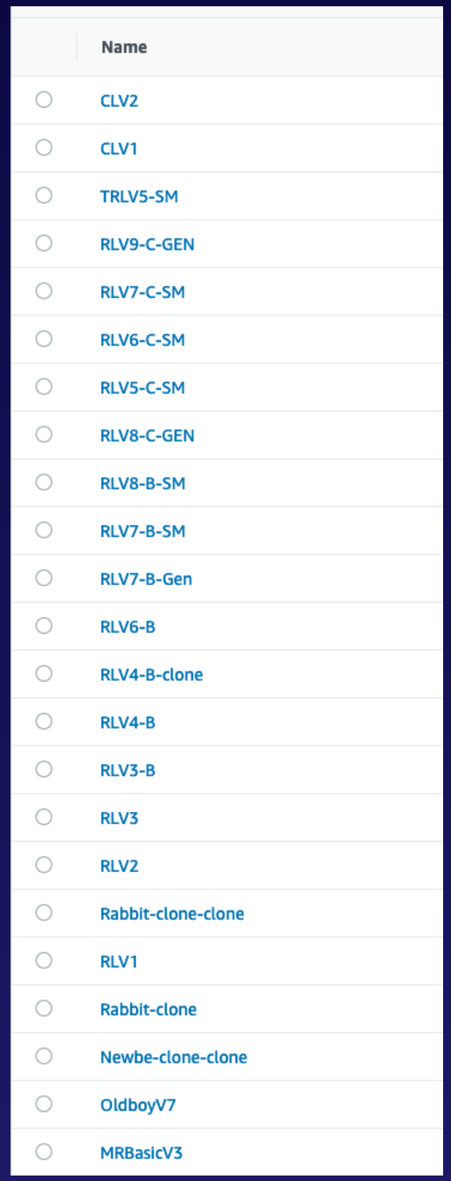


A CLOUD GURU

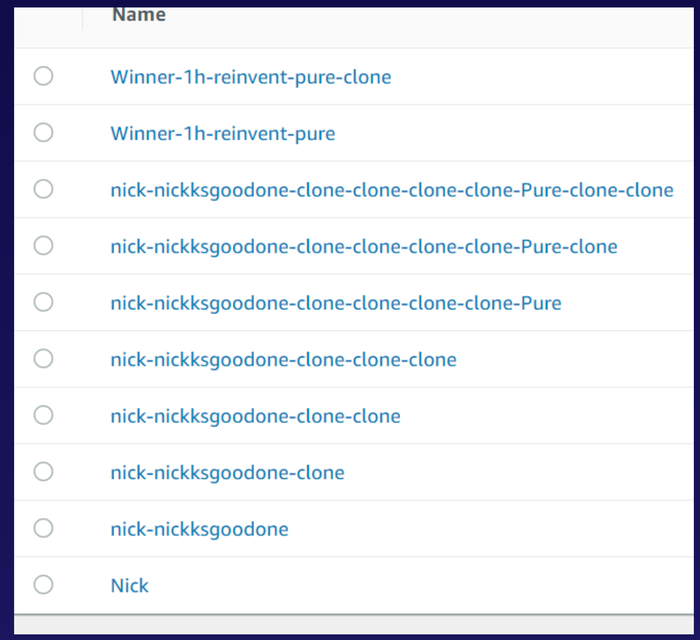
SCOTT



NICK K.



NICK T.



A CLOUD GURU

Scott's Slightly Credible

Tips For DeepRacer Glory™

Dare to be Different

Turn the Knobs

Don't Fear the Greek

Iterate. Iterate. Iterate.

A CLOUD GURU

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.

A CLOUD GURU

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.

A CLOUD GURU

Use the Reinforce!

A CLOUD GURU

Thank You & Keep Being Awesome!

@scottpletcher

linkedin.com/in/scottpletcher