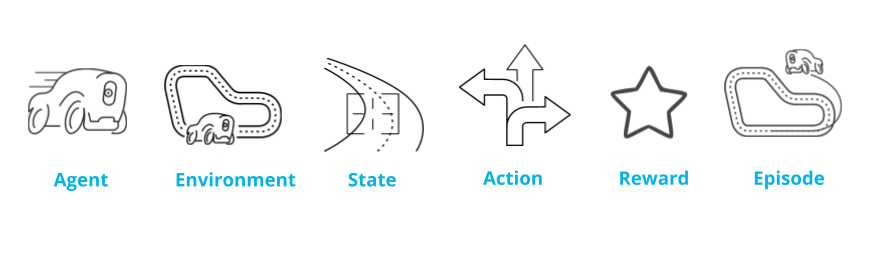
**Reinforcement Learning with AWS DeepRacer**

**Reinforcement Learning Concepts**

In this section, we’ll learn some basic reinforcement learning terms and concepts using AWS DeepRacer as an example.

**Summary**

This section introduces six basic reinforcement learning terms and provides an example for each in the context of AWS DeepRacer.



Basic RL terms: Agent, environment, state, action, reward, and episode

**Agent**

* The piece of software you are training is called an agent.
* It makes decisions in an environment to reach a goal.
* In AWS DeepRacer, the agent is the AWS DeepRacer car and its goal is to finish \* laps around the track as fast as it can while, in some cases, avoiding obstacles.

**Environment**

* The environment is the surrounding area within which our agent interacts.
* For AWS DeepRacer, this is a track in our simulator or in real life.

**State**

* The state is defined by the current position within the environment that is visible, or known, to an agent.
* In AWS DeepRacer’s case, each state is an image captured by its camera.
* The car’s initial state is the starting line of the track and its terminal state is when the car finishes a lap, bumps into an obstacle, or drives off the track.

**Action**

* For every state, an agent needs to take an action toward achieving its goal.
* An AWS DeepRacer car approaching a turn can choose to accelerate or brake and turn left, right, or go straight.

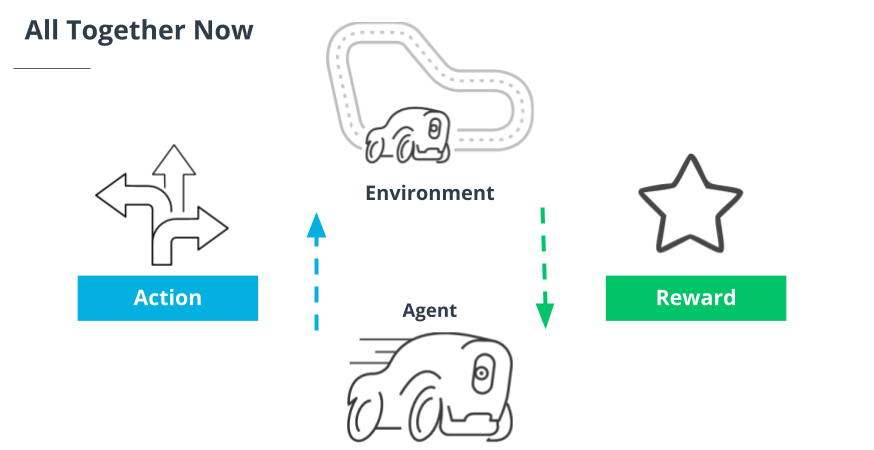
**Reward**

* Feedback is given to an agent for each action it takes in a given state.
* This feedback is a numerical reward.
* A reward function is an incentive plan that assigns scores as rewards to different zones on the track.

**Episode**

* An episode represents a period of trial and error when an agent makes decisions and gets feedback from its environment.
* For AWS DeepRacer, an episode begins at the initial state, when the car leaves the starting position, and ends at the terminal state, when it finishes a lap, bumps into an obstacle, or drives off the track.

In a reinforcement learning model, an **agent** learns in an interactive real-time **environment** by trial and error using feedback from its own **actions**. Feedback is given in the form of **rewards**.



In a reinforcement learning model, an agent learns in an interactive real-time environment by trial and error using feedback from its own actions. Feedback is given in the form of rewards.

**Putting Your Spin on AWS DeepRacer: The Practitioner's Role in RL**

**Summary**

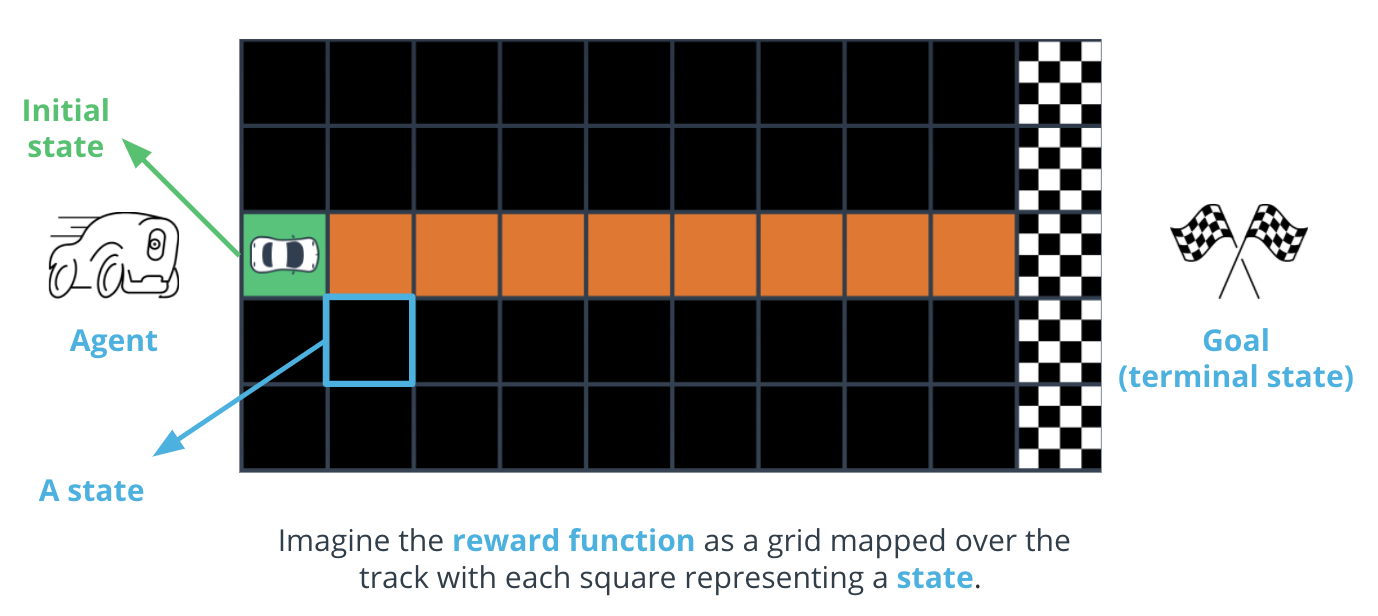
AWS DeepRacer may be autonomous, but you still have an important role to play in the success of your model. In this section, we introduce the **training algorithm, action space, hyperparameters,** and **reward function** and discuss how your ideas make a difference.

* An *algorithm* is a set of instructions that tells a computer what to do. ML is special because it enables computers to learn without being explicitly programmed to do so.
* The *training algorithm* defines your model’s learning objective, which is to maximize total cumulative reward. Different algorithms have different strategies for going about this.
  + A *soft actor critic (SAC)* embraces exploration and is data-efficient, but can lack stability.
  + A *proximal policy optimization* (PPO) is stable but data-hungry.
* An *action space* is the set of all valid actions, or choices, available to an agent as it interacts with an environment.
  + *Discrete action space* represents all of an agent's possible actions for each state in a finite set of steering angle and throttle value combinations.
  + *Continuous action space* allows the agent to select an action from a range of values that you define for each sta te.
* *Hyperparameters* are variables that control the performance of your agent during training. There is a variety of different categories with which to experiment. Change the values to increase or decrease the influence of different parts of your model.
  + For example, the *learning rate* is a hyperparameter that controls how many new experiences are counted in learning at each step. A higher learning rate results in faster training but may reduce the model’s quality.
* The *reward function*'s purpose is to encourage the agent to reach its goal. Figuring out how to reward which actions is one of your most important jobs.

**Putting Reinforcement Learning into Action with AWS DeepRacer**

**Summary**

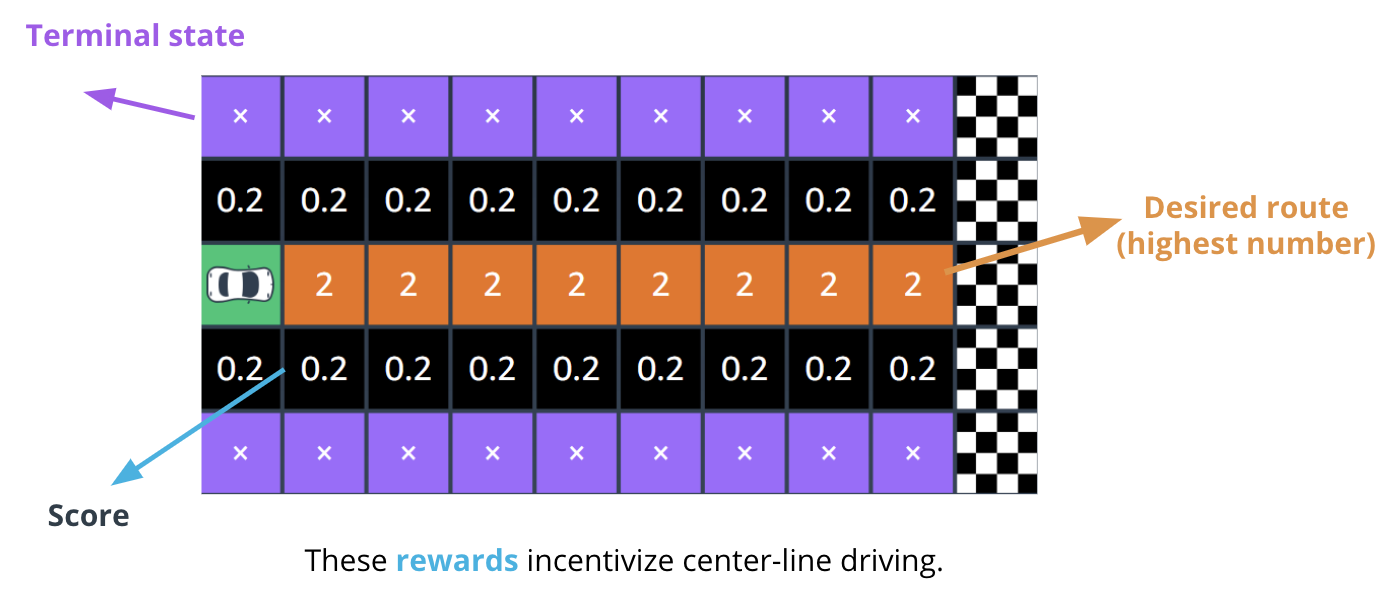
This video put the concepts we've learned into action by imagining the reward function as a grid mapped over the race track in AWS DeepRacer’s training environment, and visualizing it as metrics plotted on a graph. It also introduced the trade-off between exploration and exploitation, an important challenge unique to this type of machine learning.



Each square is a state. The green square is the starting position, or initial state, and the finish line is the goal, or terminal state.

Key points to remember about **reward functions:**

* Each state on the grid is assigned a score by your reward function. You incentivize behavior that supports your car’s goal of completing fast laps by giving the highest numbers to the parts of the track on which you want it to drive.
* The [reward function](https://docs.aws.amazon.com/deepracer/latest/developerguide/deepracer-reward-function-input.html?utm_source=Udacity&utm_medium=Webpage&utm_campaign=Udacity%20AWS%20ML%20Foundations%20Course) is the [actual code](https://docs.aws.amazon.com/deepracer/latest/developerguide/deepracer-reward-function-examples.html?utm_source=Udacity&utm_medium=Webpage&utm_campaign=Udacity%20AWS%20ML%20Foundations%20Course) you'll write to help your agent determine if the action it just took was good or bad, and how good or bad it was.



The squares containing exes are the track edges and defined as terminal states, which tell your car it has gone off track.

Key points to remember about **exploration versus exploitation:**

* When a car first starts out, it *explores* by wandering in random directions. However, the more training an agent gets, the more it learns about an environment. This experience helps it become more confident about the actions it chooses.
* *Exploitation* means the car begins to exploit or use information from previous experiences to help it reach its goal. Different training algorithms utilize exploration and exploitation differently.

Key points to remember about the **reward graph:**

* While training your car in the AWS DeepRacer console, your training metrics are displayed on a *reward graph*.
* Plotting the total reward from each episode allows you to see how the model performs over time. The more reward your car gets, the better your model performs.

Key points to remember about **AWS DeepRacer:**

* [AWS DeepRacer](https://aws.amazon.com/deepracer/?utm_source=Udacity&utm_medium=Webpage&utm_campaign=Udacity%20AWS%20ML%20Foundations%20Course) is a combination of a [physical car](https://aws.amazon.com/deepracer/robotics-projects/?utm_source=Udacity&utm_medium=Webpage&utm_campaign=Udacity%20AWS%20ML%20Foundations%20Course) and a [virtual simulator](https://console.aws.amazon.com/deepracer/home?region=us-east-1&utm_source=Udacity&utm_medium=Webpage&utm_campaign=Udacity%20AWS%20ML%20Foundations%20Course#createModel) in the [AWS Console](https://aws.amazon.com/console/?utm_source=Udacity&utm_medium=Webpage&utm_campaign=Udacity%20AWS%20ML%20Foundations%20Course), the [AWS DeepRacer League](https://aws.amazon.com/deepracer/league/?utm_source=Udacity&utm_medium=Webpage&utm_campaign=Udacity%20AWS%20ML%20Foundations%20Course), and [community races](https://console.aws.amazon.com/deepracer/home?region=us-east-1&utm_source=Udacity&utm_medium=Webpage&utm_campaign=Udacity%20AWS%20ML%20Foundations%20Course#communityRaces).
* An AWS DeepRacer device is not required to start learning: you can start now in the AWS console. The 3D simulator in the AWS console is where training and evaluation take place.

**New Terms**

* **Exploration versus exploitation:** An agent should exploit known information from previous experiences to achieve higher cumulative rewards, but it also needs to explore to gain new experiences that can be used in choosing the best actions in the future.

**Additional Reading**

* If you are interested in more tips, workshops, classes, and other resources for improving your model, you'll find a wealth of resources on the [AWS DeepRacer Pit Stop](https://aws.amazon.com/deepracer/racing-tips/?utm_source=Udacity&utm_medium=Webpage&utm_campaign=Udacity%20AWS%20ML%20Foundations%20Course) page.
* For detailed step-by-step instructions and troubleshooting support, see the [AWS DeepRacer Developer Documentation](https://docs.aws.amazon.com/deepracer/index.html?utm_source=Udacity&utm_medium=Webpage&utm_campaign=Udacity%20AWS%20ML%20Foundations%20Course).
* If you're interested in reading more posts on a range of DeepRacer topics as well as staying up to date on the newest releases, check out the [AWS Discussion Forums](https://forums.aws.amazon.com/forum.jspa?forumID=318&utm_source=Udacity&utm_medium=Webpage&utm_campaign=Udacity%20AWS%20ML%20Foundations%20Course).
* If you're interested in connecting with a thriving global community of reinforcement learning racing enthusiasts, join the [AWS DeepRacer Slack community](http://join.deepracing.io/?utm_source=Udacity&utm_medium=Webpage&utm_campaign=Udacity%20AWS%20ML%20Foundations%20Course).
* If you're interested in tinkering with [DeepRacer's open-source device software](https://aws.amazon.com/deepracer/robotics-projects/?utm_source=Udacity&utm_medium=Webpage&utm_campaign=Udacity%20AWS%20ML%20Foundations%20Course) and collaborating with robotics innovators, check out our [AWS DeepRacer GitHub Organization](https://github.com/aws-deepracer?utm_source=Udacity&utm_medium=Webpage&utm_campaign=Udacity%20AWS%20ML%20Foundations%20Course).

;