AWS DeepRacer Student

User Guide

AWS DeepRacer Student: User Guide

Table of Contents

What is AWS DeepRacer Student?

AWS DeepRacer Student is a place for high school and college-enrolled learners around the globe to develop machine learning (ML) skills. It provides access to educational material, the optional AWS AI & ML Scholarship program, and the opportunity to train and test reinforcement learning (RL) models for the AWS DeepRacer Student League. To get started, see the topics in the ??? (p. 1) section.

AWS DeepRacer Student features

- **Home** – Find details about upcoming events, practice training RL models, access ML educational content, and track your model training hours. You can also manage your AWS DeepRacer Student profile and account info from the left navigation pane.
- **Learn** – Access ML content, including videos, developed by AWS experts. Students with no prior experience can learn ML fundamentals from easy-to-understand, self-paced material.
- **Practice** – Choose a track, algorithm, and reward function to create a RL model. Optionally, take a guided walkthrough of the reward function Python code and choose to customize it. Train your model in a simulated 3D racing environment using the AWS DeepRacer service. Clone your top performing models and iterate on their reward functions to climb the AWS DeepRacer Student League leaderboard.
- **Compete** – Enter your models into monthly competitions for the chance to win prizes, including devices and experiences. All students have 10 hours that can be used to train RL models for the AWS DeepRacer Student League. Only models trained using AWS DeepRacer Student can be used in the AWS DeepRacer Student League.

Topics

Are you interested in the AWS AI & ML Scholarship program?

If you self-identify as underserved or underrepresented in technology, opt into the AWS AI & ML Scholarship program. To learn more about who qualifies, how to apply, and what you win, see What is the AWS AI & ML Scholarship program? (p. 2)

Are you a first-time AWS DeepRacer Student user?

If you are a first-time user of AWS DeepRacer Student, we recommend that you begin by reading the following sections:

- Getting started with AWS DeepRacer Student (p. 7)
- Join an AWS DeepRacer Student race (p. 15)
- Training a reinforcement learning model in AWS DeepRacer Student (p. 11)

What is the AWS AI & ML Scholarship program?

Launching as part of AWS DeepRacer Student, the AWS AI & ML Scholarship program is designed to bring diversity to the field of artificial intelligence (AI) and machine learning (ML) by offering successful applicants the opportunity to earn up to two Udacity Nanodegrees. The Udacity Nanodegree is massive open online courses (MOOCs) designed to bridge the gap between learning and career goals. For more details, see What is a Nanodegree Program? in the *Udacity support documentation*. Successful applicants will also have access to exclusive events and mentoring to help them further their careers. For more details see, What you win in the AWS AI & ML Scholarship (p. 2).

This scholarship is focused on people who are underserved and underrepresented in tech. Applicants must be at least 16 years old, and currently enrolled in high school, university, or community colleges. For more details about how you prequalify, see How do I prequalify for the AWS AI & ML Scholarship program? (p. 3)

Participation in the AWS AI & ML Scholarship program is free.

The AWS AI & ML Scholarship program officially launched on April 11, 2022.

The AWS AI & ML Scholarship program works on a cohort-based approach. Each year, two cohorts of 1,000 students are selected (2,000 total). The applications for the first cohort are due to Udacity on May 31, 2023, and the applications for the second cohort are due on September 30, 2023.

What you win in the AWS AI & ML Scholarship

For the AWS AI & ML Scholarship program, AWS is collaborating with Udacity and Intel.

Each year, the AWS AI & ML Scholarship program will globally offer 2,500 Udacity Nanodegree scholarships spread across two different cohorts. All students receive the following:

- Free admission to the *AI Programming with Python* Nanodegree. This course teaches the foundational skills necessary to start using AI techniques and develop your skills in programming, linear algebra, and neural networks.
- Access to office hours with Udacity instructors during the week to help answer questions about class content. Also, students can participate in weekly case study exercises lead by Udacity instructors.
- Exclusive access to AMA (Ask Me Anything) events, fireside chats with professionals in the AI/ML industry, and additional office hours with AI/ML professionals from Amazon and presenting sponsors. Finally, the program will also include exclusive career fairs complete with workshops such as resume editing and mock interviews.

Also, the top 500 students from the AI Programming with Python class (based on performance in course assessments) are given access to a more advanced Nanodegree that covers the fundamentals of deep learning, and provides you with skills required to be a machine learning engineer.

The same top 500 students will receive 12 months of mentorship with AI/ML professionals from AWS or Intel to help them prepare for a career in tech.

How do I prequalify for the AWS AI & ML Scholarship program?

The AWS AI & ML Scholarship program is intended for underserved and underrepresented students who are 16 years or older. Underrepresented and underserved students include (but are not limited to) women, people with disabilities, people of color (Black, Latinx, and Indigenous), and members of the LGBTQ+ community.

Students who prequalify for the AWS AI & ML Scholarship receive a unique access code they can redeem on the Udacity site to access the Udacity Nanodegree scholarship application form. Students who prequalify are not guaranteed Udacity Nanodegree scholarships. Udacity determines which prequalified students are awarded Udacity Nanodegree scholarships.

Prequalifying for the scholarship is based on two criteria:

Review coursework and pass assessments

To prequalify, you must score 80% or higher on all required assessments. Each assessment is based on different chapters you can find in the **Learn** section in the AWS DeepRacer Student navigation pane. The scholarship application opens on February 1, 2023.

Achieve a minimum lap time

Each month from February to September, a new leaderboard is launched in the AWS DeepRacer League, featuring a new track. To prequalify for the scholarship, you will need to complete a *time trial* in less than 3 minutes on any single leaderboard.

Frequently asked questions (FAQs) about the AWS AI & ML Scholarship program and the Udacity platform

What are the official AWS AI & ML Scholarship terms and conditions?

To view the official terms and conditions, see Official AWS AI & ML Scholarship program Terms and Conditions.

Who is this scholarship for?

This AWS AI & ML Scholarship program is intended for underserved and underrepresented students around the world who are 16 years or older. Underrepresented and underserved students include (but are not limited to) women, people with disabilities, people of color (Black, Latinx, and Indigenous), and members of the LGBTQ+ community.

What is a nanodegree?

A nanodegree is an online skills-based educational program that helps to bridge the gap between learning and career skills.

If I prequalify for the AWS AI & ML Scholarship, do I automatically receive a Udacity Nanodegree scholarship?

No. Students who prequalify for the AWS AI & ML Scholarship are given access to the application for the Udacity Nanodegree scholarship. Udacity determines which prequalified students are awarded Udacity Nanodegree scholarships.

Am I required to provide proof of enrollment before being selected for the AWS AI & ML Scholarship?

You might be asked to provide proof of enrollment (such as a college transcript) to receive the AWS AI & ML Scholarship.

How do I indicate that I am currently enrolled in high school when applying for the AWS AI & ML Scholarship program?

To indicate that you are enrolled in high school, see Step 4: Update your profile (p. 8) in the *AWS DeepRacer Student User Guide*.

How is my personally identifiable information (PII) data protected?

Your data is secured in the AWS cloud. For more information, see the AWS privacy notice.

Is there an age requirement in order to apply for the AWS AI & ML Scholarship?

Yes. To apply for the AWS AI & ML Scholarship program, you must be at least 16 years old when you sign up.

How will I be notified if I receive the AWS AI & ML Scholarship?

You will receive an email from Udacity regarding the status of your nanodegree scholarship application after the deadline for applications has passed.

When will I be notified if I receive the AWS AI & ML Scholarship?

Udacity will contact applicants about their scholarship status. Students who are selected for a scholarship will be provided with information about important dates.

What happens if I am accepted into the AWS AI & ML Scholarship program?

After being notified that you have received the AWS AI & ML Scholarship, a representative from the Udacity onboarding team will contact you using the email that you provided to Udacity when you completed your application.

If I receive a Udacity Nanodegree scholarship, how much time should I expect to spend on my nanodegree studies?

Students should expect to spend approximately 10 hours per week on their Udacity nanodegree studies.

Is English language proficiency required?

Although English proficiency is not required, learning materials are in English. To be successful, students should have good reading and writing skills in English.

What if I don't find the options I identify with in the race/gender list?

For both, you can use the option **I prefer to self identify (Select to type)**. For more details, see Step 3: (Optional) Read about and opt in to AWS AI & ML Scholarship consideration (p. 8) in the *AWS DeepRacer Student User Guide*.

What are AWS Player accounts?

AWS Player accounts are a managed identity solution for AWS Bugbust, AWS DeepRacer multi-user, and AWS DeepRacer Student created by AWS. Your AWS Player account holds *all* of the resources created in each of these AWS services.

Creating an AWS Player account for supported services

When you create an account for either AWS BugBust, AWS DeepRacer multi-user, or AWS DeepRacer Student you automatically create an AWS Player account. When you use different features in these services, new resources are added automatically into your AWS Player account. To get started with AWS Bugbust, AWS DeepRacer multi-user, and AWS DeepRacer Student use the following links.

Creating an AWS DeepRacer Student account

To use AWS DeepRacer Student, get started by creating an account. To learn how to create an account see, Step 1: Sign up for AWS DeepRacer Student (p. 7) in the *AWS DeepRacer Student User Guide*.

Creating an AWS BugBust account (admin and player)

To get started with AWS BugBust you need to have *both* an AWS Player account and access to the AWS Management Console. To learn how to get started see Creating the AWS BugBust accounts required to use AWS BugBust in the *AWS BugBust User Guide*

Authorized email addresses are used in AWS BugBust events to score points. Points are scored based on pull requests created on GitHub. After creating your AWS BugBust account you might need to add additional **Authorized emails**. To learn more, see Adding authorized email addresses in the AWS BugBust player portal in the *AWS BugBust User Guide*.

Use AWS DeepRacer multi-user to sponsor multiple participants under one account.

AWS DeepRacer multi-user mode support two different user profiles, admin and participant. Both have different setup requirements. To get started, see Multi-user Mode in the *AWS DeepRacer Developer Guide*.

Deleting an AWS Player account

If you delete an AWS Player account, you immediately lose access to all supported services. This includes any achievements (badges, points, avatars, etc) that you earned.

Deleting your AWS Player account account does not delete your AWS account. If you would also like to delete your AWS account, use the steps outlined in Closing your AWS account.

If you have used your AWS Player account account to create an event in either AWS BugBust or AWS DeepRacer multi-user you cannot delete your AWS Player account account. This is to ensure that participants in events you have created are not left with a broken experience. To learn more about how an admin creates events in AWS BugBust or AWS DeepRacer multi-user mode use these topics.

Setting up events using AWS DeepRacer multi-user mode (admin)

To learn how to create events using multi-user mode, see Set up multi-user mode (admin) in the *AWS DeepRacer Developer Guide*.

Creating an AWS BugBust event (admin)

To get started creating a bug bash event, see Create an AWS BugBust event (admin) in the *AWS BugBust User Guide*.

AWS Player accounts do not have access to any AWS resources other than those created in the service's account. Any AWS Identity and Access Management policies and associated resources in the service account are limited to only the required resources.

Getting started with AWS DeepRacer Student

Use this tutorial to get started with AWS DeepRacer Student. The tutorial explains how to log in to AWS DeepRacer Student, update your profile, opt in to consideration for the AWS AI & ML Scholarship, start taking free courses in machine learning (ML) and reinforcement learning (RL), and create AWS DeepRacer models. If you don't opt in to the scholarship during account creation, you can opt in at a later time when you update your profile.

Topics

Prerequisites

To access AWS DeepRacer Student and participate, you need:

- To be a student who is at least 16 years old and currently enrolled in a high school, community college, or college.
- Or be an educator or event organizer for students in high school, university, or community college.
- A valid email address.

Step 1: Sign up for AWS DeepRacer Student

You can sign up for AWS DeepRacer Student by using the URL provided in this procedure. When you sign up, you create an AWS Player account. This account can be used with certain other AWS services. If you already have an AWS Player account, you can use that account with AWS DeepRacer Student.

1. Open the http://deepracerstudent.com/ landing page.
2. Choose **Get started**.
3. On the **Sign in** page, if you don't already have an AWS Player account, choose **Sign up**.

   Note
   If you already have an AWS Player account, enter your information here. For more information about the AWS Player account, see What are AWS Player accounts? (p. 5)
4. On the **Sign up** page, enter the following information:

   - **Email address**
   - **Password**
5. Choose **Sign up**. An email with a confirmation code is sent to the email address you specified.

6. In the pop-up that appears, enter your verification number and choose **Verify**.
7. On the AWS DeepRacer Student **Sign in** page, enter your **Email address** and **Password** and choose **Sign in**.
8. On the **Welcome to the AWS DeepRacer Student** pop-up, choose **Complete sign-up**.

   Note
   You can choose **I will do this later. Sign out for now** if you want to sign up at a later time.

Step 2: Complete sign-up for AWS DeepRacer Student

Complete the section to create your AWS DeepRacer Student account. All fields are required unless otherwise stated.

1. Fill in the fields in the **Add your personal information to create your AWS DeepRacer Student account** section to create your account.
2. Select the checkbox to certify that you are a student enrolled in either high school or a university or community college.

   Note
   If you are a high school student, do the following:

   - For **School**, choose **Other**. Then, add the name of your high school to the field **Enter the name of your school**.
   - For **Current or prospective major,** choose **Undecided** or choose a possible prospective major from the list.

Step 3: (Optional) Read about and opt in to AWS AI & ML Scholarship consideration

Read the information about the AWS AI & ML Scholarship and for whom it is intended in the **Do you want to be considered for the AWS AI & ML Scholarship program?** section.

1. If you meet the criteria for the AWS AI & ML Scholarship, you can opt in to be considered. Choose the checkbox to confirm that you want to be considered for a scholarship.

   Note
   If you don't want to be considered for a scholarship or are undecided, leave the checkbox unselected and choose either **I will do this later. Sign out for now.** to sign out for now, or **Submit** to continue without opting in. You can also opt in to consideration for a scholarship when you update your profile.
2. (Optional) Use the dropdown lists to enter your information in the **Choose gender** and **Choose race (US participants only)** fields.
3. Choose **Submit**.

Step 4: Update your profile

To update your profile, use the **Your profile** page. You can also choose to opt in to the AWS AI & ML Scholarship program.

To update your profile

1. In AWS DeepRacer Student, in the left navigation pane, choose **Your profile**.
2. On the **Your profile** page, in **Racer name**, choose **Change your racer name**.

   Note
   Your racer name can have between 2 and 24 characters. Letters, numbers, and hyphen (-) are allowed.
3. In the **Racer name** pop-up, enter your racer name and choose **Save**. If you decide that you don't want to change your racer name, choose **Cancel**.
4. In the **Your profile information** section, you can make changes to the following fields:

   - Name
   - Racer name
   - School name
   - Name of major
   - Year of graduation
   - Country of residency
5. (Optional) In the **Do you want to be considered for the AWS AI & ML Scholarship program?** section, you can view information about the scholarship and how to apply.

   a. Select the checkbox to opt in for consideration in the scholarship program.
   b. (Optional) Enter your information in the **Choose gender** and **Choose race** fields.
   c. Choose **Submit**.

Step 5: Explore AWS DeepRacer Student from the **Home** page

The AWS DeepRacer Student **Home** page is the perfect place to start your exploration of all that AWS DeepRacer Student has to offer. From the **Home** page, you can do the following:

Get started learning the fundamentals of machine learning (ML)

You can use the free courses available in the **Learn** section of AWS DeepRacer Student. This robust offering helps you build a foundation for your machine learning journey with AWS DeepRacer Student.

Practice using your machine learning knowledge

After spending some time using the **Learn** courses, you're ready to create and train an AWS DeepRacer model. For more information, see Training a reinforcement learning model in AWS DeepRacer Student (p. 11).

Compete in AWS DeepRacer Student races

When you've trained your first AWS DeepRacer model, you're ready to join a race. Choose a track and pick a season in which to race. When you've completed racing, you see your model on the leaderboard along with the information and data you need to make changes and improve your model. For more information, see Join an AWS DeepRacer Student race (p. 15).

Check model training hours

When you train and clone models, you use a portion of your free model training time. You can check on your model training hours remaining on the home page.

Check the AWS DeepRacer Student racing calendar

View the racing calendar and start planning for your race day.

Learn about the AWS AI & ML Scholarship

You can find out about the AWS AI & ML Scholarship and what you can do to prepare. For more information, see What is the AWS AI & ML Scholarship program? (p. 2)

View other resources

You can discover other resources that can help you on your exploration of AWS DeepRacer Student such as the Discord channel and AWS DeepRacer website. These resources help connect you to a community of racers and fans sharing tips and insights.

Training a reinforcement learning model in AWS DeepRacer Student

This walkthrough demonstrates how to train your first model in AWS DeepRacer Student. It also provides you with some useful tips to help you make the most of your experience and fast-track your learning.

Step 1: Train a reinforcement learning model using AWS DeepRacer Student

Begin your journey in AWS DeepRacer Student by learning where to find the **Create model** button and start training your first model. Keep in mind that creating and training a model is an iterative process. Experiment with different algorithms and reward functions to achieve your best results.

To train a reinforcement learning model

1. In the AWS DeepRacer Student **Home** page, choose **Create a model**. Alternatively, navigate to **Your Models** in the left navigation pane. In the **Models** page, in **Your models**, choose **Create model**.
2. In the **Overview** page, read about how to train a reinforcement model. Each step in the process is explained on this page. When you've finished reading, choose **Next**.

Step 2: Name your model

Name your model. It's good practice to give your models unique names to quickly locate individual models when you want to improve and clone them. For example, you may want to name your models using a naming convention such as: *`yourinitials-date-version`*.

To name your model

1. On the **Name your model** page, enter a name in the **Model name** field.

   Note
   When you begin training a model, the model's name becomes fixed and is no longer changeable.
2. Choose **Next**.

Step 3: Choose your track

Choose your simulation track. The track serves as the environment and provides data to your car. If you choose a very complex track, your car requires a longer total training time and the reward function you use is more complex.

To choose your track (environment)

1. On the **Choose track** page, choose a track to serve as a training environment for your car.
2. Choose **Next**.

Step 4: Choose an algorithm

The AWS DeepRacer Student has two training algorithms from which to choose. Different algorithms maximize rewards in different ways. To make the most of your AWS DeepRacer Student experience, experiment with both algorithms. For more information about algorithms, see AWS DeepRacer Training Algorithms.

To choose a training algorithm

1. On the **Choose algorithm type** page, select an algorithm type. Two algorithm types are available:
   - **Proximal Policy Optimization (PPO)**. This stable but data hungry algorithm performs consistently between training iterations.
   - **Soft Actor Critic (SAC)**. This unstable but data efficient algorithm can perform inconsistently between training iterations.
2. Choose **Next**.

Step 5: Customize your reward function

The reward function is at the core of reinforcement learning. Use it to incentivize your car (agent) to take specific actions as it explores the track (environment). Just as you would encourage and discourage certain behaviors in a pet, you can use this tool to encourage your car to finish a lap as fast as possible and discourage it from driving off of the track and zig-zagging.

When training your first model, you may want to use a default sample reward function. When you're ready to experiment and optimize your model, you can customize the reward function by editing the code in the code editor. For more information about customizing the reward function, see Customizing a reward function (p. 17).

To customize your reward function

1. On the **Customize reward function** page, choose a sample reward function. There are 3 sample reward functions available that you can customize:
   - **Follow the centerline**. Rewards your car when it autonomously drives as close as it can to the centerline of the track.
   - **Stay within borders**. Rewards your car when it autonomously drives with all four wheels staying within the track borders.
   - **Prevent zig-zag**. Rewards your car for staying near the centerline. Penalizes your car if it uses high steering angles or goes off track.

   Note
   If you don't want to customize the reward function, choose **Next**.
2. (Optional) Modify the reward function code.
   - Select a sample reward function and choose **Walk me through this code**.
   - For each section of the code, you can view more information by selecting the **+** to reveal a pop-up textbox with explantory text. Progress through the code walkthrough by choosing **Next** in each pop-up. To exit out of a pop-up textbox, choose the **X** in the corner. To exit the walkthrough, choose **Finish**.

     Note
     You can choose not to edit the sample reward function code by selecting **Go with default code**.

- Optionally, edit the sample reward function code by selecting a sample reward function and choosing **Edit sample code**. Edit the code and select **Validate** to check your code. If your code can't be validated or you would like to reset the code to its original state, choose **Reset**.

3. Choose **Next**.

Step 6: Choose duration and submit your model to the leaderboard

The duration of your model's training impacts its performance. When experimenting in the early phase of training, you should start with a small value for this parameter and then progressively train for longer periods of time.

In this step of training your model, your trained model is submitted to a leaderboard. You can opt out by deselecting the checkbox.

To choose duration and submit a model to the leaderboard

1. On the **Choose duration** page, select a time in **Choose duration of model training**.
2. In the **Model description** field, enter a useful description for your model that will help you to remember the selections you made.

   Tip
   It's good practice to add information about your model such as current selections and modifications for the reward function and algorithm as well as your hypothesis about how the model will perform.
3. Select the checkbox to automatically have your model submitted to the AWS DeepRacer Student leaderboard after training is complete. Optionally, you may opt out of entering your model by deselecting the checkbox.

   Tip
   We recommend that you submit your model to the leaderboard. Submitting your model helps you to see how your model compares to others and provides you with feedback so you can improve your model.
4. Choose **Train your model**.
5. In the **Initializing model training** pop-up, choose **Okay**.
6. On the **Training configuration** page, you can review your model's training status and configuration. You can also view a video of your model training on the selected track when the training **Status** is **In progress**. Watching the video can help you develop valuable insights that you can use to improve your model.

Step 7: View your model's performance on the leaderboard

After you have trained your model and submitted it to a leaderboard, you can view its performance.

To view your model's performance

1. In the left navigation pane, navigate to and expand **Compete**. Choose a season. On the **Leaderboard** page, your model and your rank appear in a section. The page also includes a **Leaderboard** section with a list of the submitted models, race details, and a **Race details** section.

2. In the page that displays the leaderboard, in the section with your profile, select **Watch Video** to view a video of your model's performance.

Step 8: Use **Clone** to improve your model

After you have trained and optionally submitted your model to a leaderboard, you can clone it to improve it. Cloning your model saves you steps and makes training more efficient by using a previously trained model as the starting point for a new model.

To clone and impove a model

1. In AWS DeepRacer Student, in the left navigation pane, navigate to **Your models**.
2. On the **Your models** page, select a model and choose **Clone**.
3. In the **Name your model** field, provide a new name for your cloned model and choose **Next**.
4. On the **Customize a reward function** page, customize the reward function and choose **Next**. For more information about customizing the reward function, see Step 5: Customize your reward function (p. 12).
5. In the **Choose duration** page, enter a time in the **Choose duration of model training** field, enter a description in the **Model description** field, and select the checkbox to submit the cloned model to the leaderboard.
6. Choose **Train your model**. Your training is initialized. The **Training configuration** page appears with information about your cloned model. You can also view a video of your model training on the selected track when the training **Status** is **In progress**.
7. Continue cloning and modifying your pre-trained models to achieve your best performance on the leaderboard.

Step 9: (Optional) Download a model

After training a model and optionally submitting it to the leaderboard, you may want to download it for future use on a AWS DeepRacer physical device. Your model is saved as a `.tar.gz` file.

To download a model

1. In AWS DeepRacer Student, in the left navigation pane, navigate to **Your models**.
2. On the **Your models** page, select a model and choose **Download**.
3. Track the progress of the model download in your browser. When your model is downloaded, you can save it to your local hard drive or other preferred storage device.

   To learn more about working with AWS DeepRacer devices, see Operate Your AWS DeepRacer Vehicle in the *AWS DeepRacer guide*.

Join an AWS DeepRacer Student race

After successfully training and evaluating your model in simulation, compare your model's performance to other racers' models by participating in a race. Racing is a great way to get feedback about your model, win awards and prizes, meet other AWS DeepRacer Student community members virtually, hear about opportunities to learn and improve your skills, and have fun. There are two types of student races: student league and community.

A *student league race* is a monthly virtual competition that all students can join. A *student community race* is a private race created by an educator or event organizer in the AWS console that students can join by invitation only.

This section discusses how to participate in an AWS DeepRacer Student student league race and student community race.

Join a student league race

In this section, learn how to submit your model to an AWS DeepRacer Student student league race. You can join a race by submitting a trained model directly to a student leaderboard. This procedure is described in the following section. For more information about training models, see Training a reinforcement learning model in AWS DeepRacer Student (p. 11).

To join a student league race

1. In the left navigation pane in AWS DeepRacer Student, navigate to and expand **Compete**. Choose the current season, such as **Student league pre-season**.
2. The page that appears displays your profile information, the race details, and leaderboard. Choose **Race again**. If you haven't already created a model, choose **Create model** to start the process. For more information, see Training a reinforcement learning model in AWS DeepRacer Student (p. 11).
3. In the **Choose a model to race** section, use the dropdown list to choose a model in the **Choose a model** field.
4. Choose **Enter race** to submit your model.
5. If your model is evaluated sucessfully against the racing criteria, check the leaderboard to see how your model ranks against other participants.
6. Optionally, choose **Watch** to view a video of your car's performance.
7. Choose **Race again** to enter another model.

Join a student community race

In this section, learn how to submit your model to an AWS DeepRacer Student student community race. You can join the student community race by receiving an invitation link from your educator or event organizer through email.

To join a student community race

1. Go to the invitation link and log in to your AWS DeepRacer student account.
2. Once you are signed in, choose the **Enter race** button.
3. In the **Choose a model to race** dropdown list, select your model to use in the community race.

4. Choose **Enter race** to submit your model.
5. If your model is evaluated sucessfully against the racing criteria, check the leaderboard to see how your model ranks against the models of other participants.
6. Optionally, choose **Watch** to view a video of your car's performance.
7. Choose **Race again** to enter another model.

Customizing a reward function

Creating a reward function is like designing an incentive plan. Parameters are values that can be used to develop your incentive plan.

Different incentive strategies result in different vehicle behaviors. To encourage the vehicle to drive faster, try awarding negative values when the car takes too long to finish a lap or goes off the track. To avoid zig-zag driving patterns, try defining a steering angle range limit and rewarding the car for steering less aggressively on straight sections of the track.

You can use waypoints, which are numbered markers placed along the centerline and outer and inner edges of the track, to help you associate certain driving behaviors with specific features of a track, like straightaways and curves.

Crafting an effective reward function is a creative and iterative process. Try different strategies, mix and match parameters, and most importantly, have fun!

Topics

Editing Python code to customize your reward function

In AWS DeepRacer Student, you can edit sample reward functions to craft a custom racing strategy for your model.

To customize your reward function

1. On the **Step 5: Customize reward function** page of the AWS DeepRacer Student **Create model** experience, select a sample reward function.
2. Use the code editor below the sample reward function picker to customize the reward function's input parameters using Python code.
3. Select **Validate** to check whether or not your code will work. Alternatively, choose **Reset** to start over.
4. Once you're done making changes, select **Next**.

Use Input parameters of the AWS DeepRacer reward function (p. 17) to learn about each parameter. See how different parameters are used in reward function examples.

Input parameters of the AWS DeepRacer reward function

The AWS DeepRacer reward function takes a dictionary object passed as the variable, `params`, as the input.

```
def reward_function(params) :

    reward = ...

    return float(reward)
```

The `params` dictionary object contains the following key-value pairs:

```
{
    "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

}
```

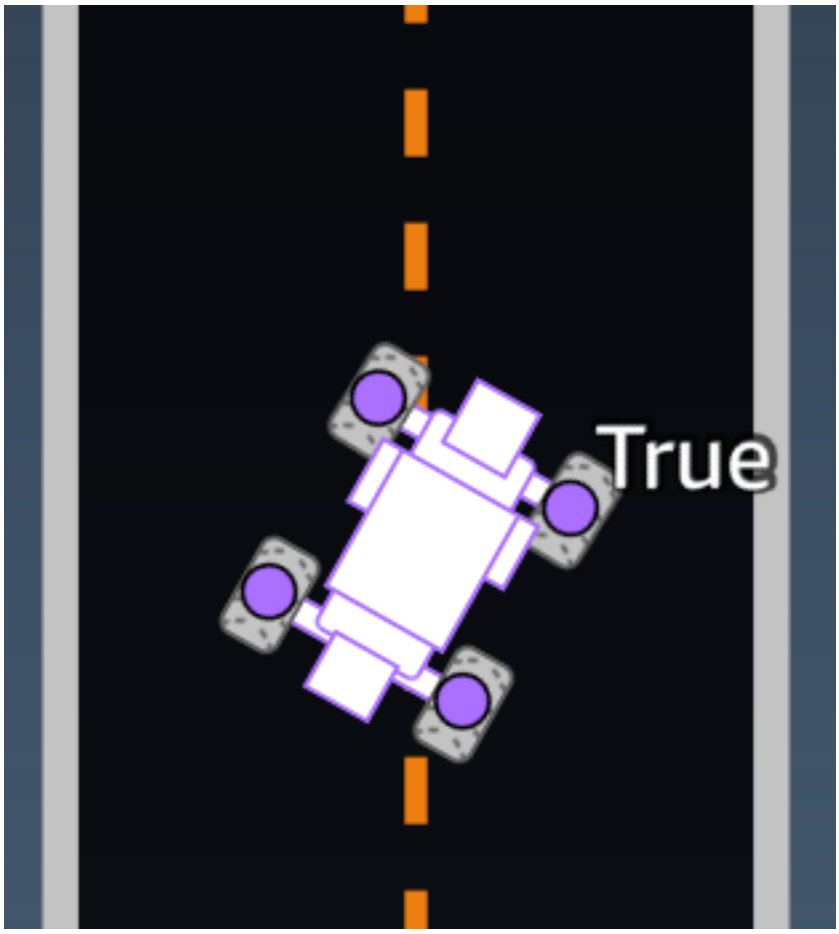
Use the following reference to get a better understanding of the AWS DeepRacer input parameters.

all_wheels_on_track

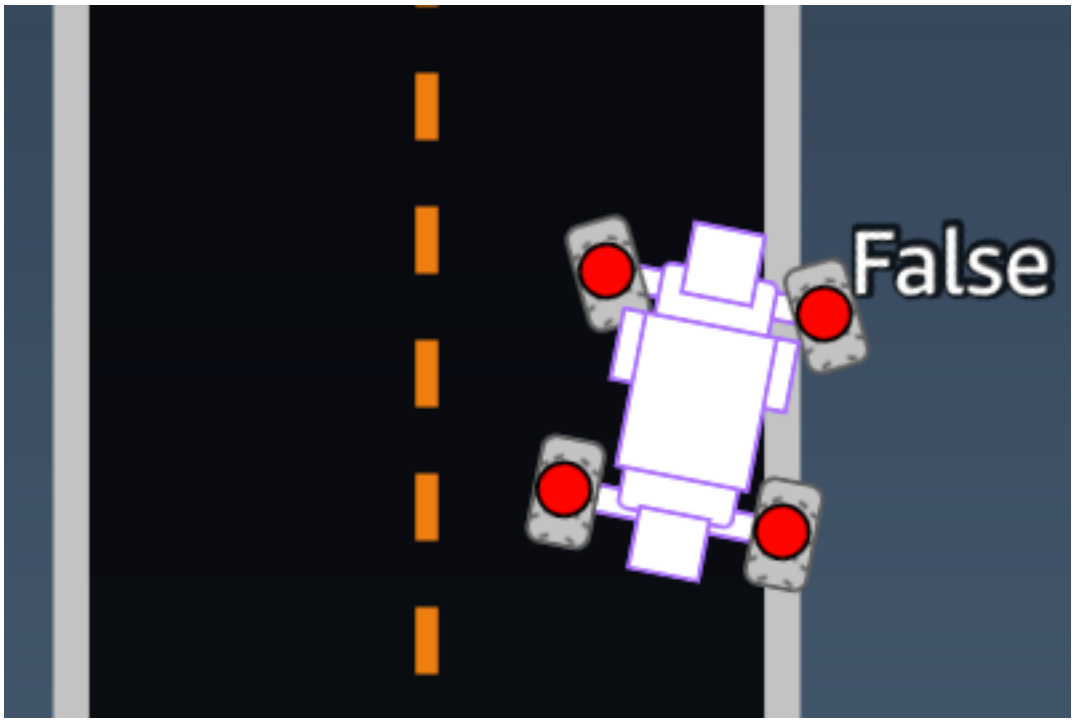
Type: `Boolean`

Range: `(True:False)`

A `Boolean` flag to indicate whether the agent is on track or not on track. The agent is not on track (`False`) if any of its wheels are outside of the track borders. It's on track (`True`) if all four wheels are inside the inner and outer track borders. The following illustration shows an agent that is on track.



The following illustration shows an agent that is not on track because two wheels are outside of track borders.



Example: *A reward function using the `all_wheels_on_track` parameter*

```
def reward_function(params):
    ################################################################################
    '''
    Example of using all_wheels_on_track and speed
    '''

    # Read input variables
    all_wheels_on_track = params['all_wheels_on_track']
    speed = params['speed']

    # Set the speed threshold based your action space
    SPEED_THRESHOLD = 1.0

    if not all_wheels_on_track:
        # Penalize if the car goes off track
        reward = 1e-3
    elif speed < SPEED_THRESHOLD:
        # Penalize if the car goes too slow
        reward = 0.5
    else:
        # High reward if the car stays on track and goes fast
        reward = 1.0

    return float(reward)
```

closest_waypoints

Type: `[int, int]`

Range: `[(0:Max-1),(1:Max-1)]`

The zero-based indices of the two neighboring `waypoints` closest to the agent's current position of `(x, y)`. The distance is measured by the Euclidean distance from the center of the agent. The first element refers to the closest waypoint behind the agent and the second element refers the closest waypoint in front of the agent. `Max` is the length of the waypoints list. In the illustration shown in waypoints (p. 31), the `closest_waypoints` are `[16, 17]`.

The following example reward function demonstrates how to use `waypoints` and `closest_waypoints` as well as `heading` to calculate immediate rewards.

AWS DeepRacer supports the following Python libraries: `math`, `random`, `numpy`, `scipy`, and `shapely`. To use one, add an import statement, `import` *`supported library`*, preceding your function definition, `def reward_function(params)`.

Example: *A reward function using the `closest_waypoints` parameter.*

```
# Place import statement outside of function (supported libraries: math, random, numpy,
 scipy, and shapely)
# Example imports of available libraries
#
# import math
# import random
# import numpy
# import scipy
# import shapely

import math

def reward_function(params):
    ################################################################################
    '''
    Example of using waypoints and heading to make the car point in the right direction
    '''

    # Read input variables
    waypoints = params['waypoints']
    closest_waypoints = params['closest_waypoints']
    heading = params['heading']

    # Initialize the reward with typical value
    reward = 1.0

    # Calculate the direction of the centerline based on the closest waypoints
    next_point = waypoints[closest_waypoints[1]]
    prev_point = waypoints[closest_waypoints[0]]

    # Calculate the direction in radius, arctan2(dy, dx), the result is (-pi, pi) in
 radians
    track_direction = math.atan2(next_point[1] - prev_point[1], next_point[0] -
 prev_point[0])
    # Convert to degree
    track_direction = math.degrees(track_direction)

    # Calculate the difference between the track direction and the heading direction of the
 car
    direction_diff = abs(track_direction - heading)
    if direction_diff > 180:
        direction_diff = 360 - direction_diff

    # Penalize the reward if the difference is too large
    DIRECTION_THRESHOLD = 10.0
    if direction_diff > DIRECTION_THRESHOLD:
        reward *= 0.5

    return float(reward)
```

closest_objects

Type: `[int, int]`

Range: `[(0:len(object_locations)-1), (0:len(object_locations)-1]`

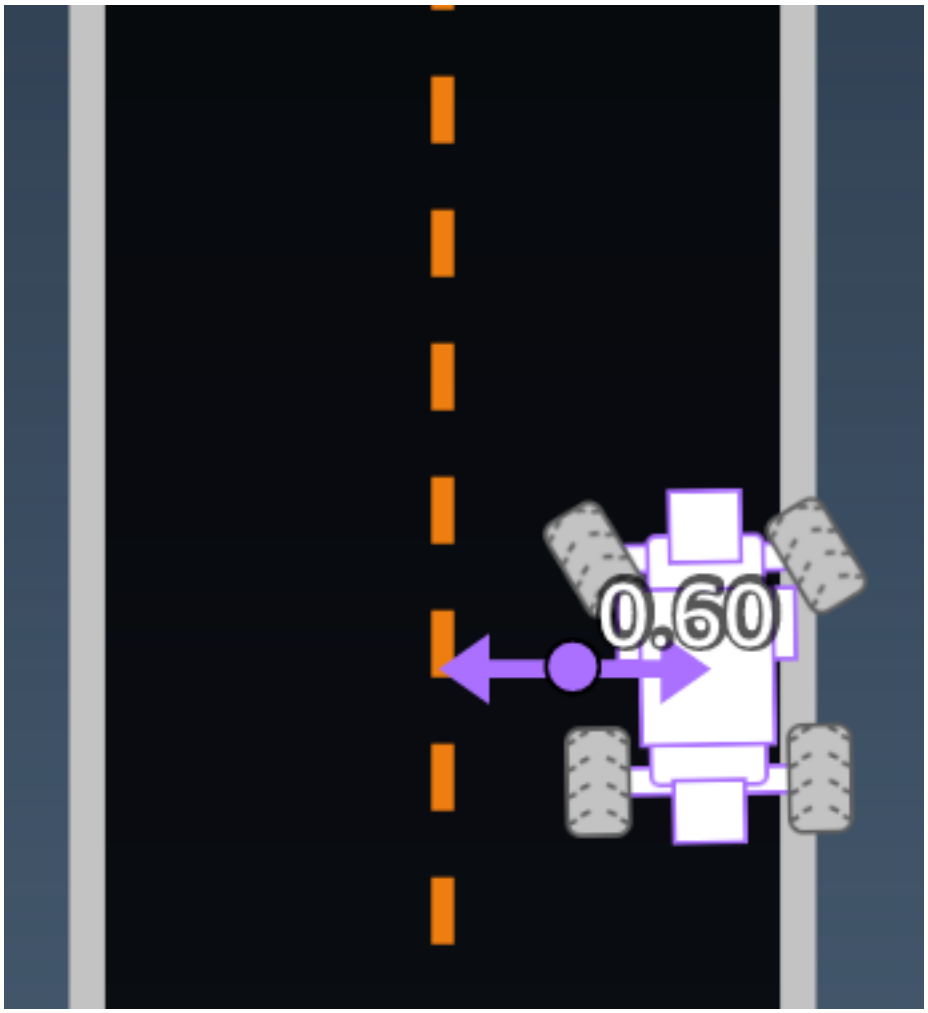
The zero-based indices of the two closest objects to the agent's current position of (x, y). The first index refers to the closest object behind the agent, and the second index refers to the closest object in front of the agent. If there is only one object, both indices are 0.

distance_from_center

Type: `float`

Range: `0:~track_width/2`

Displacement, in meters, between the agent's center and the track's center. The observable maximum displacement occurs when any of the agent's wheels are outside a track border and, depending on the width of the track border, can be slightly smaller or larger than half the `track_width`.



Example: *A reward function using the `distance_from_center` parameter*

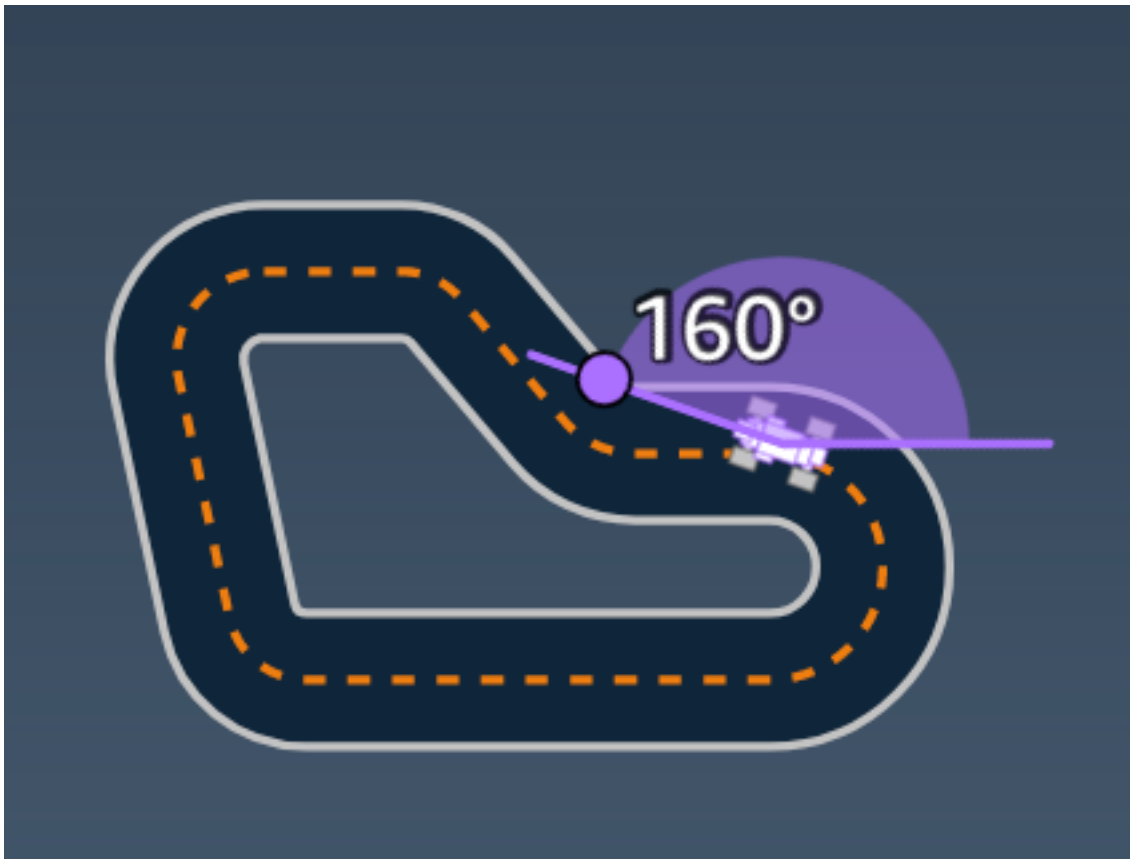
```
def reward_function(params):
    ################################################################################
    '''
    Example of using distance from the center
    '''

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

    # Penalize if the car is too far away from the center
    marker_1 = 0.1 * track_width
    marker_2 = 0.5 * track_width

    if distance_from_center <= marker_1:
        reward = 1.0
    elif distance_from_center <= marker_2:
        reward = 0.5
    else:
        reward = 1e-3  # likely crashed/ close to off track

    return float(reward)
```

heading

Type: `float`

Range: `-180:+180`

The heading direction, in degrees, of the agent with respect to the x-axis of the coordinate system.



Example: *A reward function using the* `heading` *parameter*

For more information, see `closest_waypoints` (p. 20).

is_crashed

Type: `Boolean`

Range: `(True:False)`

A `Boolean` flag that indicates whether the agent has crashed into another object (`True`) or not (`False`) as a termination status.

is_left_of_center

Type: `Boolean`

Range: `[True : False]`

A `Boolean` flag that indicates if the agent is left of the track center (`True`) or not left of the track center (`False`).

is_offtrack

Type: `Boolean`

Range: `(True:False)`

A `Boolean` flag that indicate if all four of the agent's wheels have driven outside of the track's inner or outer boarders (`True`) or not (`False`).

is_reversed

Type: `Boolean`

Range: `[True:False]`

A `Boolean` flag that indicates if the agent is driving clockwise (`True`) or counterclockwise (`False`).

It's used when you enable direction change for each episode.

objects_distance

Type: `[float, … ]`

Range: `[(0:track_length), … ]`

A list of distances between objects in the environment in relation to the starting line. The i^{th} element measures the distance in meters between the i^{th} object and the starting line along the track center line.

> **Note**
> abs | (var1) - (var2)| = how close the car is to an object, WHEN var1 = ["objects_distance"][index] and var2 = params["progress"]*params["track_length"]
> To get an index of the closest object in front of the vehicle and the closest object behind the vehicle, use the `closest_objects` parameter.

objects_heading

Type: `[float, … ]`

Range: `[(-180:180), … ]`

List of the headings of objects in degrees. The i^{th} element measures the heading of the i^{th} object. Stationary objects' headings are 0. For bot cars, the corresponding element's value is the bot car's heading angle.

objects_left_of_center

Type: `[Boolean, … ]`

Range: `[True|False, … ]`

List of `Boolean` flags. The i^{th} element value indicates whether the i^{th} object is to the left (`True`) or right (`False`) of the track center.

objects_location

Type: `[(x,y), ...]`

Range: `[(0:N,0:N), ...]`

This parameter stores all object locations. Each location is a tuple of (x, y (p. 31)).

The size of the list equals the number of objects on the track. The objects listed include both stationary obstacles and moving bot cars.

objects_speed

Type: `[float, … ]`

Range: `[(0:12.0), … ]`

List of speeds (meters per second) for the objects on the track. For stationary objects, their speeds are 0. For a bot vehicle, the value is the speed you set in training.

progress

Type: `float`

Range: `0:100`

Percentage of track completed.

Example: *A reward function using the `progress` parameter*

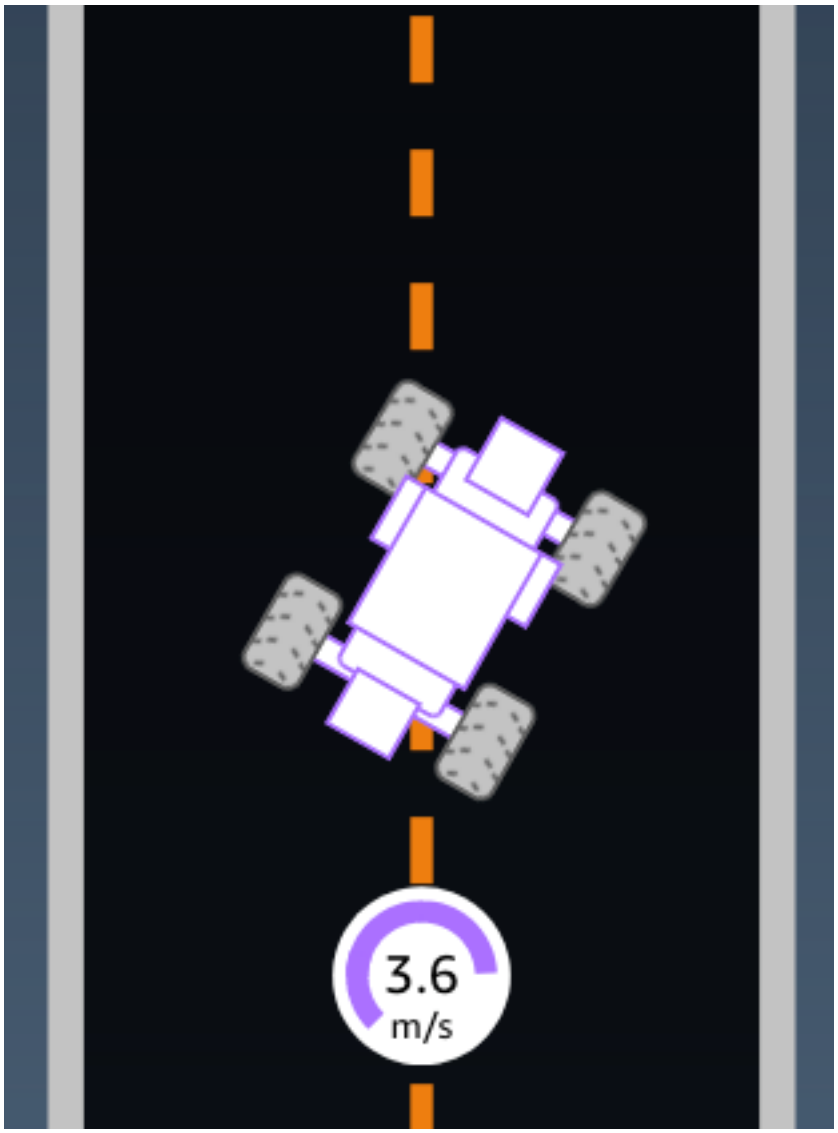
For more information, see steps (p. 29).

speed

Type: `float`

Range: `0.0:5.0`

The observed speed of the agent, in meters per second (m/s).



Example: *A reward function using the speed parameter*

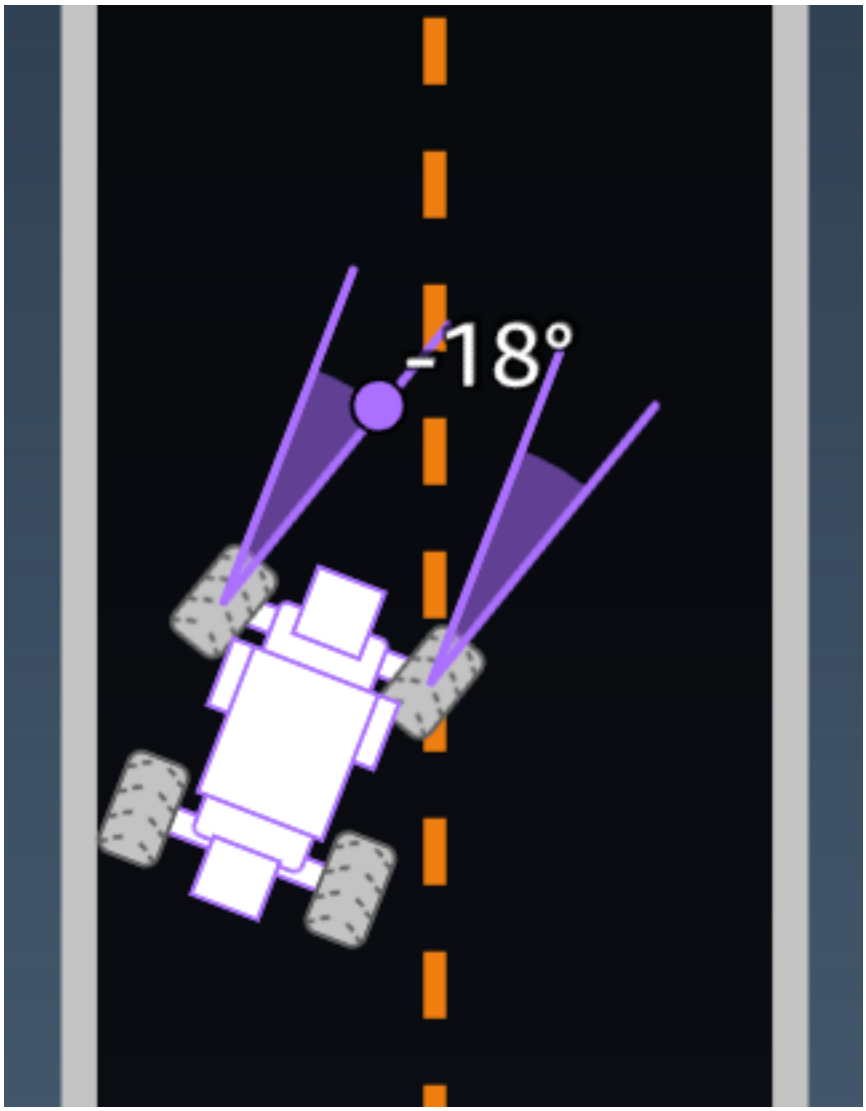
For more information, see all_wheels_on_track (p. 18).

steering_angle

Type: `float`

Range: `-30:30`

Steering angle, in degrees, of the front wheels from the center line of the agent. The negative sign (-) means steering to the right and the positive (+) sign means steering to the left. The agent's centerline is not necessarily parallel with the track center line as is shown in the following illustration.



Example: *A reward function using the `steering_angle` parameter*

```
def reward_function(params):
    '''
    Example of using steering angle
    '''

    # Read input variable
    abs_steering = abs(params['steering_angle']) # We don't care whether it is left or
 right steering

    # Initialize the reward with typical value
    reward = 1.0

    # Penalize if car steer too much to prevent zigzag
    ABS_STEERING_THRESHOLD = 20.0
    if abs_steering > ABS_STEERING_THRESHOLD:
        reward *= 0.8

    return float(reward)
```

steps

Type: `int`

Range: `0:`N_{step}

The number of steps completed. A step corresponds to one observation-action sequence completed by the agent using the current policy.

Example: *A reward function using the `steps` parameter*

```
def reward_function(params):
    ################################################################################
    '''
    Example of using steps and progress
    '''

    # Read input variable
    steps = params['steps']
    progress = params['progress']

    # Total num of steps we want the car to finish the lap, it will vary depends on the
 track length
    TOTAL_NUM_STEPS = 300

    # Initialize the reward with typical value
    reward = 1.0

    # Give additional reward if the car pass every 100 steps faster than expected
    if (steps % 100) == 0 and progress > (steps / TOTAL_NUM_STEPS) * 100 :
        reward += 10.0

    return float(reward)
```

track_length

Type: `float`

Range: `[0:`L_{max}`]`

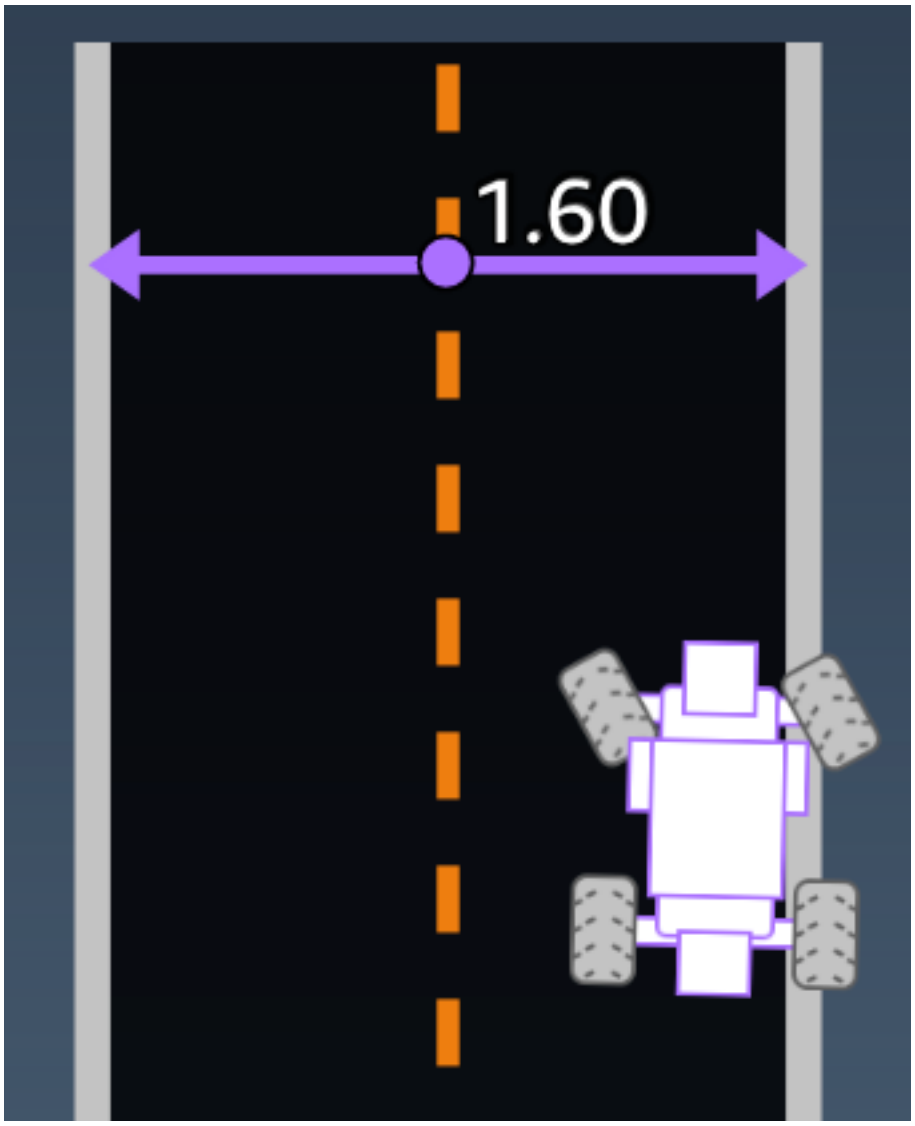
The track length in meters. L_{max} `is track-dependent.`

track_width

Type: `float`

Range: `0:`D_{track}

Track width in meters.



Example: *A reward function using the `track_width` parameter*

```
def reward_function(params):
    ################################################################################
    '''
    Example of using track width
    '''

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

    # Calculate the distance from each border
    distance_from_border = 0.5 * track_width - distance_from_center

    # Reward higher if the car stays inside the track borders
    if distance_from_border >= 0.05:
        reward = 1.0
    else:
        reward = 1e-3 # Low reward if too close to the border or goes off the track
```

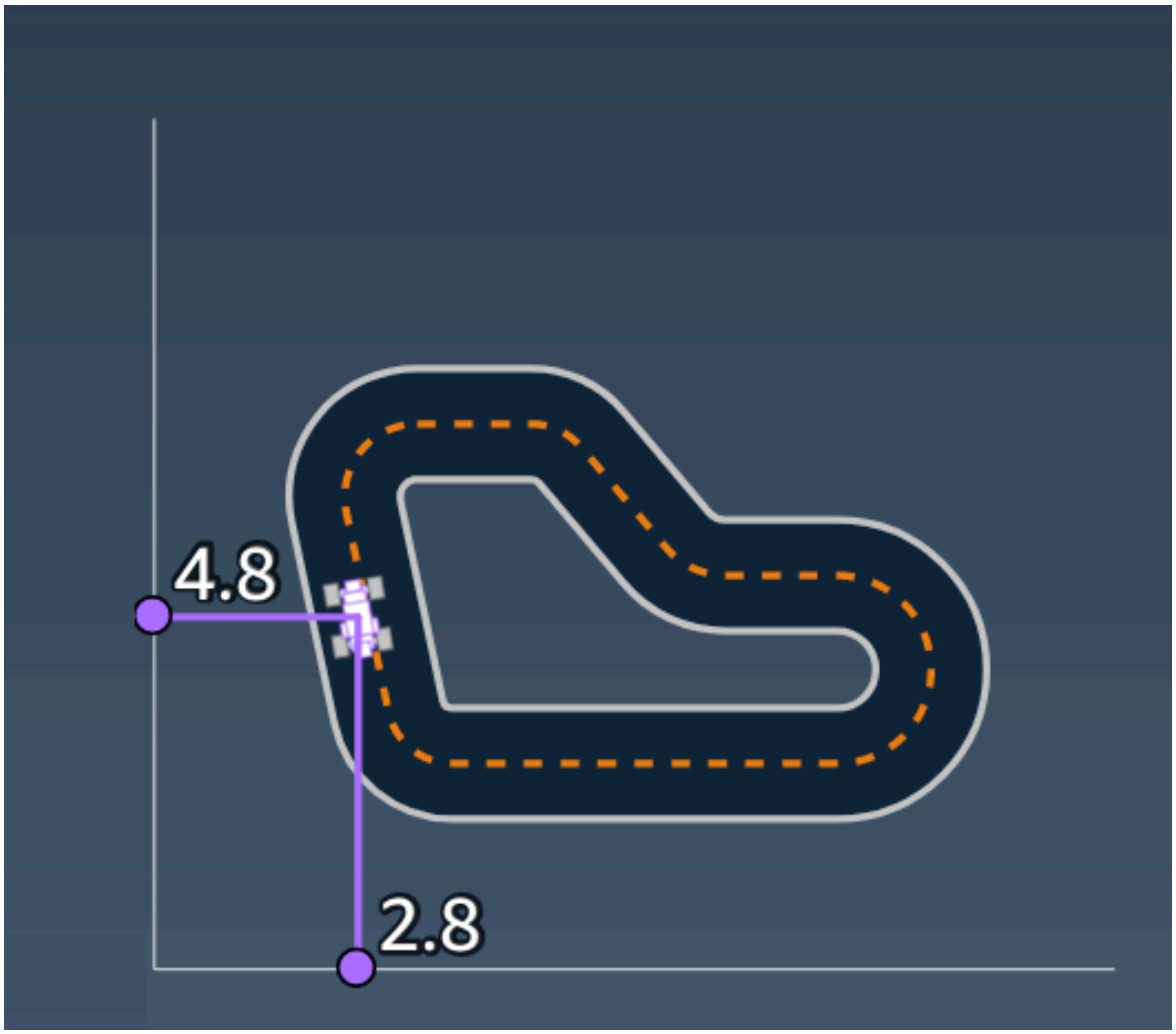
```
    return float(reward)
```

x, y

Type: `float`

Range: `0:N`

Location, in meters, of the agent's center along the x and y axes of the simulated environment containing the track. The origin is at the lower-left corner of the simulated environment.

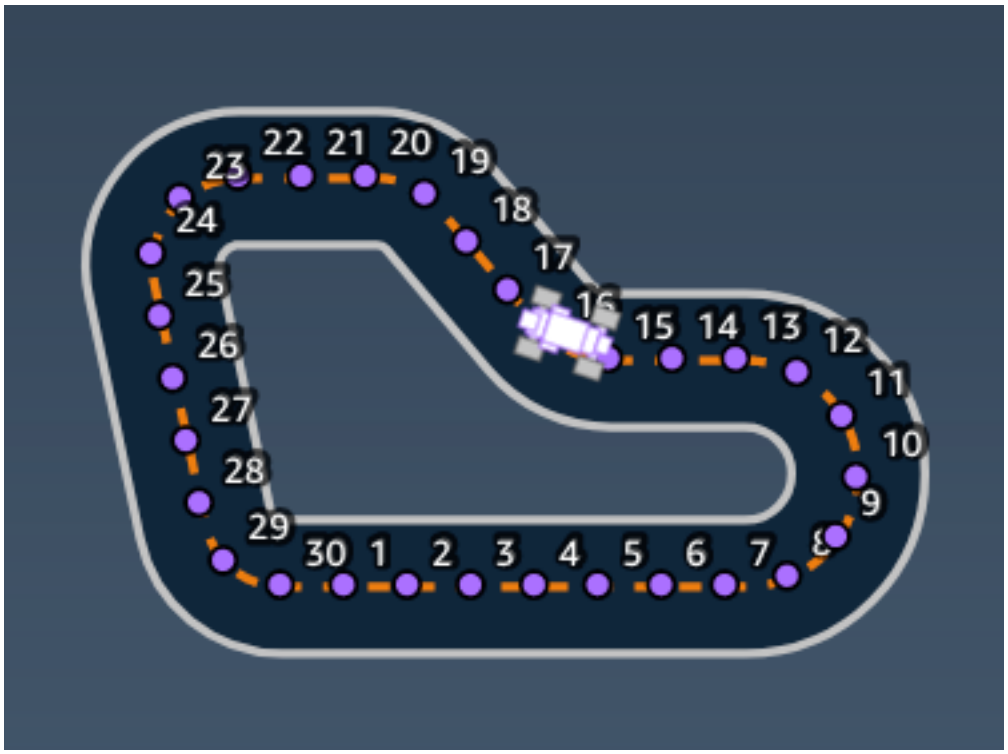


waypoints

Type: `list` of `[float, float]`

Range: $[[x_{w,0}, y_{w,0}]$ … $[x_{w,Max-1}, y_{w,Max-1}]]$

An ordered list of track-dependent `Max` milestones along the track center. Each milestone is described by a coordinate of $(x_{w,i}, y_{w,i})$. For a looped track, the first and last waypoints are the same. For a straight or other non-looped track, the first and last waypoints are different.



Example *A reward function using the* `waypoints` *parameter*

For more information, see `closest_waypoints` (p. 20).

Security in AWS DeepRacer Student

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from a data center and network architecture that is built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The shared responsibility model describes this as security of the cloud and security in the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the AWS Compliance Programs. To learn about the compliance programs that apply to AWS DeepRacer Student, see AWS Services in Scope by Compliance Program.
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your company's requirements, and applicable laws and regulations

This documentation helps you understand how to apply the shared responsibility model when using AWS DeepRacer Student. It shows you how to configure AWS DeepRacer Student to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your AWS DeepRacer Student resources.

Contents

Data protection in AWS DeepRacer Student

The following sections explain what data is captured by AWS DeepRacer Student, and where AWS DeepRacer Student uses data encryption to protect your data.

When you create a AWS DeepRacer Student account you also create an AWS Player account. Resources created in your AWS DeepRacer Student account are stored in your AWS Player account. For more details about AWS Player accounts, see What are AWS Player accounts? (p. 5) in the *AWS DeepRacer Student User Guide*.

Topics

Captured data in the AWS DeepRacer Student portal

To use the AWS DeepRacer Student portal, the required data is stored in your AWS Player account. The data captured in the AWS DeepRacer Student portal is not used to help improve the service.

Captured data in AWS DeepRacer Student.

The following is a summary of data created in AWS DeepRacer Student and stored in your AWS Player account.

- Your email address and password used to register your account.
- Your racer name
- Your standing on the Student League leaderboard
- Your trained models
- Reward function code

Encryption at rest in AWS DeepRacer Student portal

Data captured by AWS DeepRacer Student portal is encrypted by default.

AWS Player accounts use Amazon Cognito to encrypt and store the email and password used to login to AWS DeepRacer Student. For more information, see Data Protection in Amazon Cognito.

All other data captured in AWS DeepRacer Student is encrypted at rest in the cloud using AWS owned keys through AWS Key Management Service with AES-GCM and using keys of size 256-bits. This data is stored and encrypted in Amazon Simple Storage Service (S3) and Amazon DynamoDB.

Encryption in transit in AWS DeepRacer Student portal

Your registered and authorized email addresses are encrypted with client-side encryption. All other data captured in AWS DeepRacer Student (p. 33) is copied out of your account and processed in an internal AWS system. By default, AWS DeepRacer Student uses secure connections over HTTPS to encrypt data in transit.

Identity and access management for AWS DeepRacer Student

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be authenticated (signed in) and authorized (have permissions) to use AWS resources. AWS DeepRacer Student does not directly integrate with IAM to control user access to AWS resources. Instead, AWS DeepRacer Student uses an authenticated proxy API managed by AWS DeepRacer to secure user resources.

Compliance validation for AWS DeepRacer Student

Third-party auditors assess the security and compliance of AWS DeepRacer Student as part of multiple AWS compliance programs.

For a list of AWS services in scope of specific compliance programs, see AWS Services in Scope by Compliance Program. For general information, see AWS Compliance Programs.

You can download third-party audit reports using AWS Artifact. For more information, see Downloading Reports in AWS Artifact.

AWS provides the following resources to help with compliance:

- Security and Compliance Quick Start Guides – These deployment guides discuss architectural considerations and provide steps for deploying security- and compliance-focused baseline environments on AWS.
- AWS Compliance Resources – This collection of workbooks and guides might apply to your industry and location.
- Evaluating Resources with Rules in the *AWS Config Developer Guide* – AWS Config; assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- AWS Security Hub – This AWS service provides a comprehensive view of your security state within AWS that helps you check your compliance with security industry standards and best practices.

Resilience in AWS DeepRacer Student

The AWS global infrastructure is built around AWS Regions and Availability Zones. Regions provide multiple physically separated and isolated Availability Zones, which are connected through low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see AWS Global Infrastructure.

Infrastructure security in AWS DeepRacer Student

As a managed service, AWS DeepRacer Student is protected by the AWS global network security procedures that are described in the Amazon Web Services: Overview of Security Processes whitepaper.

Troubleshooting common AWS DeepRacer Student issues

Topics

Why was I automatically signed out of my AWS DeepRacer Student account?

In compliance with AWS security policy, you are automatically signed out of your AWS DeepRacer Student account after 30 days.

- To continue using the service, navigate to the AWS DeepRacer Student sign-in page and use your credentials to sign back in.

How do I opt out of the AWS AI & ML Scholarship program?

The AWS AI & ML Scholarship program is optional and intended for underserved and underrepresented students who are 16 years or older. When you sign up for AWS DeepRacer Student, by default you are not enrolled in the AWS AI & ML Scholarship program.

To participate, you must first opt in by checking the box in the **Do you want to be considered for the AWS AI & ML Scholarship program?** section as you sign up for AWS DeepRacer Student or later from the **Your profile** page, which is accessible from the site's left navigation pane.

- Opting in to the program only gives you access to the application process. You can still choose not to apply.

I can't delete my AWS DeepRacer Student account

If you can't delete your AWS DeepRacer Student account, check to see if you've created an AWS BugBust or AWS DeepRacer multi-user event. AWS Player accounts are a managed identity solution created by AWS for AWS BugBust, AWS DeepRacer multi-user, and AWS DeepRacer Student. Your AWS Player account holds all of the resources created in each of these AWS services.

- To ensure that participants of events you create are not left with a broken experience, you can't delete your AWS DeepRacer Student account if it contains resources for an AWS BugBust or AWS DeepRacer multi-user event.

I can't find my school name on the dropdown list

You may not find all schools on the dropdown list, particularly high schools.

- If your school isn't on the dropdown list, choose **Other** and enter your school name.

I can't continue training my model

You may have exceeded the monthly model training-hour limit.

- Go to the **Home page** to check your **Used training hours** in the **Model training hours remaining** section. If you have exceeded your model training hours, wait until your hours reset to begin training again.

I get an "An account is registered with this email" error message

You receive this error message when you enter a confirmation code into the AWS Player account **Sign up** page and you've already used the same email address to sign up for an AWS Player account through AWS BugBust or AWS DeepRacer multi-user. You also receive this error when you've previously used the same email address to sign up for an AWS DeepRacer Student account.

- Sign in to the AWS DeepRacer Student sign-in page using the credentials you've previously created or request a password reset by selecting **Forgot password?** under the **Password** field.

I signed up with a Gmail account and can't find my verification code

If you've signed up for an AWS Player account using a Gmail account and can't find your verification code message, it may have been delivered to the wrong folder.

- Sign in to your Gmail account and check your **Promotions** folder for a message titled, "Your AWS Player profile verification code".

Quotas for AWS DeepRacer Student

Every student participating in AWS DeepRacer Student receives 10 free hours of monthly model training compute resources and 5GB of storage.

Deleting your AWS DeepRacer Student account

The AWS DeepRacer Student portal stores the following information in your AWS Player account:

- Email address
- Your password
- Your racer name
- Your Student League leaderboard rank

To learn more about the data collected, see Data protection in AWS DeepRacer Student (p. 33) in the *AWS DeepRacer Student User Guide*.

If you want to remove this information from AWS's servers, use the following procedure to delete your AWS DeepRacer Student portal account. Deleting your AWS DeepRacer Student account also deletes your AWS Player account and all associated resources.

To learn more about AWS Player accounts, see What are AWS Player accounts? (p. 5)

> **Note**
> If you are an AWS BugBust event administrator or have created AWS DeepRacer multi-user event, you cannot delete your AWS Player account. For more details, see I can't delete my AWS DeepRacer Student account (p. 36) in the *AWS DeepRacer Student User Guide*.

To delete your AWS DeepRacer Student player account

> **Important**
> Deleting your AWS DeepRacer Student account is an action that cannot be undone. When you delete your AWS DeepRacer Student you are also deleting your AWS Player account and all associated resources.
> When you delete your AWS DeepRacer Student account, the resources in your AWS Player account are removed from our servers within one year.

1. Open the AWS DeepRacer Student landing page: https://student.deepracer.com/signIn.
2. If prompted, log in to your AWS DeepRacer Student account.
3. Choose **Your account**.
4. On the **Your account** page, choose **Delete your account**.
5. Under **To confirm deletion, type Delete in the field,** type **`Delete`**.
6. Choose **Delete**.

When your account is deleted successfully, the message **Account deleted successfully** appears and you are returned to the AWS DeepRacer Student login page.

If you would also like to delete your AWS account, use the steps outlined in Closing your AWS account.

We know customers care deeply about privacy and data security and we implement responsible and sophisticated technical and physical controls designed to prevent unauthorized access to or disclosure of customer content. Maintaining customer trust is an ongoing commitment. You can learn more about AWS data privacy commitments on our Privacy Notice page.

Document history for the AWS DeepRacer Student User Guide

The following table describes the documentation releases for AWS DeepRacer Student.

Change	Description	Date
Initial release (p. 40)	Initial release of the AWS DeepRacer Student User Guide including support for the AWS AI & ML Scholarship program and AWS DeepRacer Student League.	December 1, 2021