

Robotic simulation

12/19/2025

100 Points Possible

Attempt 1



In Progress

NEXT UP: Submit assignment

Add comment

Unlimited Attempts Allowed

Details

Additional information:

- Please familiarise yourself with [regulations governing assessment offences, including Plagiarism and Collusion](https://www.herts.ac.uk/_data/assets/pdf_file/0007/237625/Structure-and-Assessment-Regulations-Undergraduate-and-Taught-Postgraduate-Programmes-AS14-Apx-3-Academic.pdf) (https://www.herts.ac.uk/_data/assets/pdf_file/0007/237625/Structure-and-Assessment-Regulations-Undergraduate-and-Taught-Postgraduate-Programmes-AS14-Apx-3-Academic.pdf) (UPR AS14).
- Engage with guidance for avoiding plagiarism via the [University SkillUp module](https://ask.herts.ac.uk/library-skillup) (<https://ask.herts.ac.uk/library-skillup>) and [1-2-1 sessions with our specialist SASH team](https://herts.instructure.com/courses/100562/pages/dot-dot-dot-a-one-to-one-session-to-talk-about-my-studies-slash-assignments?module_item_id=2430174) (https://herts.instructure.com/courses/100562/pages/dot-dot-dot-a-one-to-one-session-to-talk-about-my-studies-slash-assignments?module_item_id=2430174).
- For **postgraduate modules**:
 - A score of 50% or above represents a pass mark.
 - late submission of any item of coursework for each day or part thereof (or for hard copy submission only, working day or part thereof) for up to five days after the published deadline, coursework relating to modules at Level 7 submitted late (including deferred coursework, but with the exception of referred coursework), will have the numeric grade reduced by 10 grade points until or unless the numeric grade reaches or is 50. Where the numeric grade awarded for the assessment is less than 50, no lateness penalty will be applied.
- In this assessment, **the student is expressly prohibited** from (i) using GenAI tools for the creation of content and (ii) using GenAI tools or a proofreader or a proofreading service for proofreading.

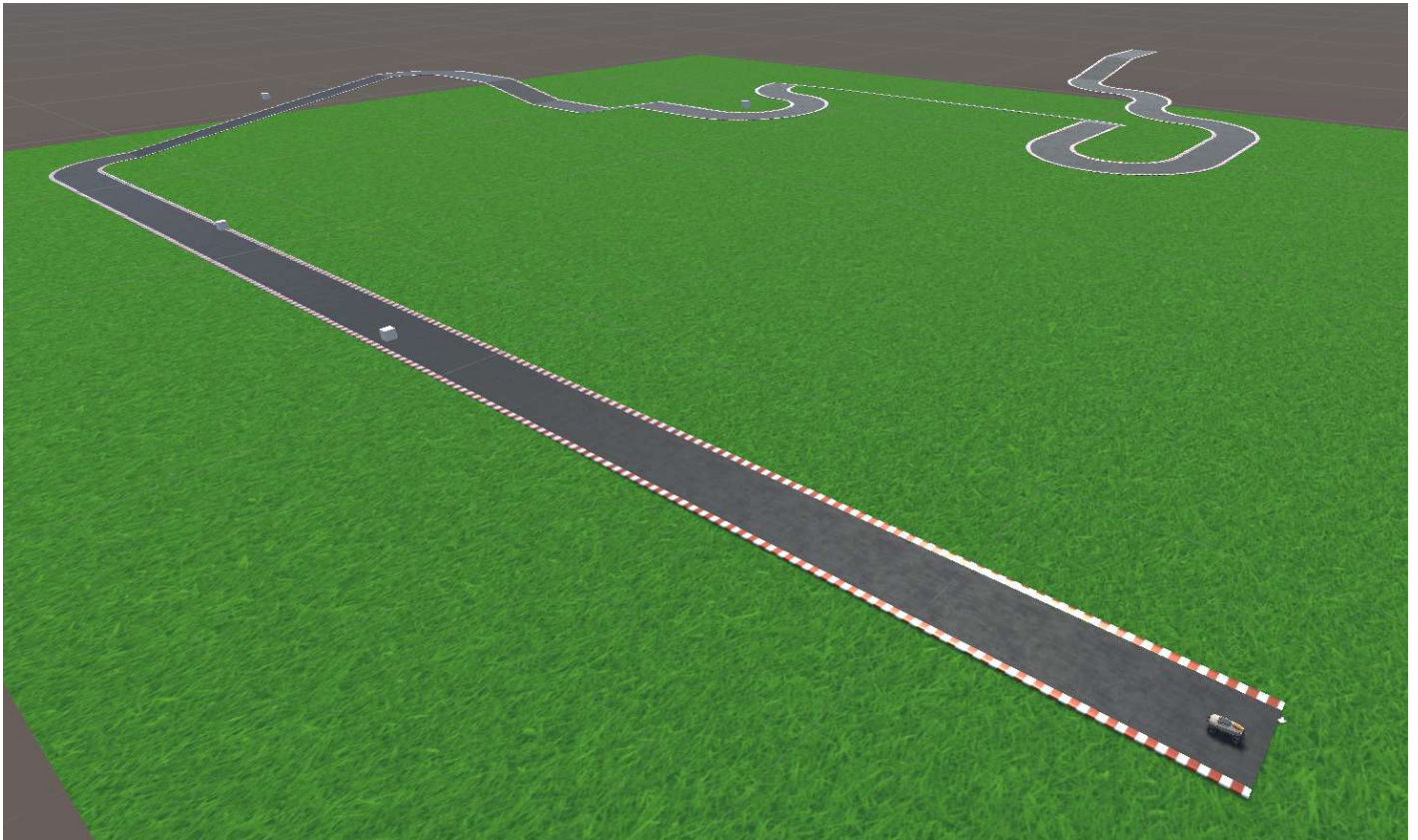
This Assignment assesses the following module Learning Outcomes (from Definitive Module Document):

- LO1 - Demonstrate knowledge and understanding of a variety of Artificial Life techniques and methods applicable across domains, including computer science and robotics.
- LO3 - Demonstrate the ability to implement an Artificial Life system according to a defined specification.

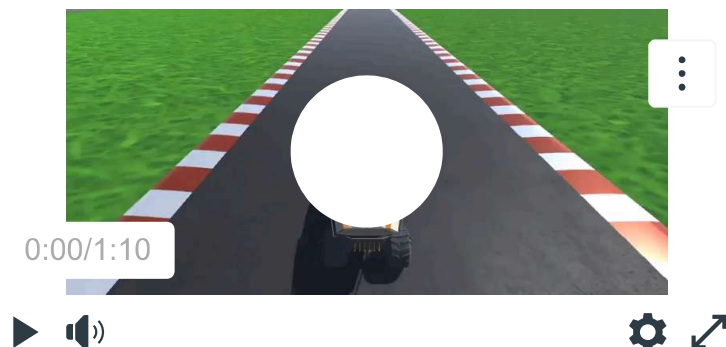
Robotic navigation

In this project, you must program a simulated robot in Unity to autonomously navigate a track with a range of obstacles. You will be provided with a complete Unity project for this assignment, and must not change any parameters of this project; the only aspect of this project you should edit is the

RobotController.cs script. You are free to develop the controller in any way you see fit. The measure of success on this project will be how far the robot makes it around a course, the time it takes, and the smoothness of the trajectory taken. This is an entirely results-based assessment.



Example of basic expected behaviour



Click here to download the starting project file: **CW2.zip**

(<https://herts.instructure.com/courses/124806/files/12439213?wrap=1>)



(https://herts.instructure.com/courses/124806/files/12439213/download?download_frd=1)

You must write a piece of code to control the robot to enable it to complete a track successfully. NOTE: the robot must not be hard-coded to the track as the track will change when it is marked, but the track will be representative of the marking track. Your code must adhere to the following requirements:

- All code must be in the **cs** file
- Do not change the name of any pre-existing variables in the code.
- Do not modify the model in Unity; all students are being assessed against the same model, meaning only your **cs** file will be used for testing.
- You are only allowed to use the Sensors provided in the model; these are named:
 - RaycastSensorFront – Must be a **Raycast** sensor (for proximity)
 - RaycastSensorLeft – Must be a **Raycast** sensor (for proximity)
 - RaycastSensorRight – Must be a **Raycast** sensor (for proximity)
 - EulerAnglesSensor – Must be an **eulerAngles** sensor (for robot orientation)
- Whilst you must use these sensors as defined above, you can set these up however you see fit.
- You can rotate the sensors placed on the robot in your code, but their physical position coordinates and scale must not change, and any changes to the rotation of the sensor must be applied via the code.
- You must use the **WheelCollider** to facilitate the robot's movements; these are named:
 - WheelColliderFrontLeft - for the front left wheel collider
 - WheelColliderFrontRight - for the front right wheel collider
 - WheelColliderBackLeft - for the back left wheel collider
 - WheelColliderBackRight - for the back right wheel collider
- You must use **Transform** to visualise the movement of the wheels. These are named:
 - TransformWheelFrontLeft - for the front left wheel
 - TransformWheelFrontRight - for the front right wheel
 - TransformWheelBackLeft - for the back left wheel
 - TransformWheelBackRight - for the back right wheel

It is advised that you use at least the following two functions in your code:

- private void Start() – This will be called before the first frame is rendered, so it is useful for setting up parameters.
- private void FixedUpdate() – This is called around 50 times per second, so ideal for physics elements.

Marks will be awarded for

Basic Model Requirements (20 marks)

- Wheels are being driven appropriately using **motorTorque**
- Visual representation of wheels appropriately applied (steering and rotational movement visible)
- Sensors are reasonably oriented to facilitate robots' perception of the environment
- The robot is physically capable of moving forward
- The robot is physically capable of steering to turn

Basic self-driving capabilities (70 marks)

- Successfully follows a track to the end without coming off
- Maintains a smooth, optimal path in completing a track
- Avoids obstacles without collision
- Adjusts speed and power to the wheels appropriately whilst travelling around the track
- Compensates for angled road surfaces
- Successfully completes a track in a good time (not crawling round slowly), the maximum time is 15 minutes
- Successfully navigates tight spaces with multiple obstacles

A screen recording of your robot running (10 marks)

- Regardless of the performance, you must provide a screen recording of your robot running from start to finish using the controller you developed to illustrate the performance of the robot.

Marks will be deducted for:

- You will lose 2 marks each time you hit the side of the track.
- You will lose 5 marks each time you hit an obstacle.

VERY IMPORTANT NOTE:

You MUST submit a screen recording, and it MUST be from the robot running the code you developed. Since you are being provided with the full project and must not change anything apart from the RobotController.cs code, it will run identically on another machine. If there is an obvious difference in the performance of the file you submit against the recording that you submit, this will be considered an attempted false representation and will subsequently receive a grade of 0.

What to Submit

This assignment requires the submission of:

- Upload one .txt file named <student ID>.txt (i.e. 1234567.txt) to Canvas containing all of the code from the C# file named RobotController.cs in your project – this is what your project will be marked on.
- Upload one video screen recording to Canvas of your robot running from start to finish on the track using the controller you developed.

Choose a submission type