

NavMesh Car Controller AI - Documentation

Vivid Blue Studios

v1.0



Table of contents:

- Description
- Scripts
- Setup Guide
 - Vehicle setup
 - NavMesh setup
- Gizmos
- Notes
- Additional Resources

Description:

The **NavMesh Car Controller AI** asset provides a solution for using Unity physics based car controllers with the Unity navigation **NavMesh** system. Unity's built-in **NavMeshAgents** are not well-suited for vehicle navigation, particularly for cars that can only steer and move in their facing direction.

The **NavMesh Car Controller AI** solution is to use a Unity **NavMeshAgent** as a steering head for your vehicle controllers, calculating suitable paths for your vehicle to follow on a **NavMesh** in real-time.

Scripts:

- **WheeledVehicleController**

This is a simple Unity based car-controller script, using Unity's default 3d rigidbody and wheel collider components to move the vehicle on 3d surfaces, controlled by a steering and acceleration input.

Public Properties:

float Speed

Current speed in velocity units.

float NormalizedSpeed

Current speed relative to **maxSpeed**. (normalized, 0–1 range)

Rigidbody vehicleRigidbody

Reference to the vehicle's Rigidbody component.

Settings:

float motorTorque

Maximum torque applied to motorized wheels for acceleration.

float brakeTorque

Maximum torque applied to wheels when braking.

float maxSpeed

Maximum speed the vehicle can reach in velocity units.

float steeringRange

Maximum steering angle in degrees at low speeds.

float steeringRangeAtMaxSpeed

Maximum steering angle at maximum speed (reduces for stability at high speeds).

float steeringInputMaxDelta

Controls how quickly steering responds to input changes (smoothing).

float centreOfGravityOffset

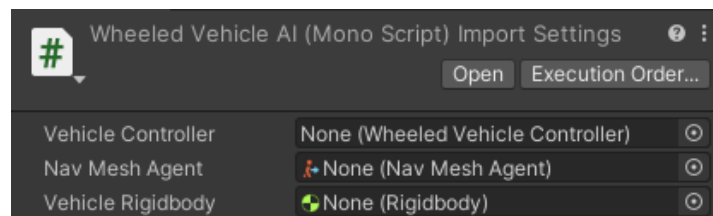
Vertical offset for the center of gravity. Negative lowers center of mass.

List<Wheel> wheels

Collection of wheel configurations for this vehicle.

• **WheeledVehicleAI**

This script handles navigating the vehicle around 3d surfaces and obstacles using a **NavMeshAgent** to calculate paths and guide steering. It can be controlled externally with Stop and Move commands, providing a target destination, it also tries to correct and reverse the vehicle if it leaves the valid **NavMesh**.



References:

WheeledVehicleController vehicleController

Reference to the vehicle controller that handles the actual vehicle physics and movement.

NavMeshAgent navMeshAgent

NavMeshAgent used for pathfinding and navigation.

Rigidbody vehicleRigidbody

Reference to the vehicle's Rigidbody for physics calculations.

Public Methods:

void Move(Vector3 targetLocation)

- Commands the vehicle to move to a specified target location.

void Stop()

- Commands the vehicle to stop moving.

Settings:

Vector2 vehicleFront

Defines the front point of the vehicle used for **NavMesh** position checking.
X is forward distance, Y is height from vehicle center.

float vehicleFrontRadius (default 1f)

Radius around the front point to check for valid NavMesh positions.

float navMeshAgentForwardOffset (range 0–10)

Forward offset for the NavMeshAgent relative to the vehicle.

float slowdownNearCornersDistance

Distance at which the vehicle starts slowing down when approaching corners.

float slowdownNearCornersRatio (range 0–1)

Ratio of normal speed to maintain when approaching corners.

float stoppingDistance

Distance at which the vehicle starts stopping when approaching the final destination.

• ClickMoveStopCommands

This script is used to send commands for the vehicle to move to a location, raycast from the cursor on the screen, in the demo scenes.

Left mouse button to move, right mouse button to stop.

References:

WheelVehicleAI vehicleAI

Reference to the AI controller for managing the vehicle's movement.

Controls:

- **Left Mouse Button:** Tell the vehicle to move to the mouse position.
- **Right Mouse Button:** Tell the vehicle to stop.

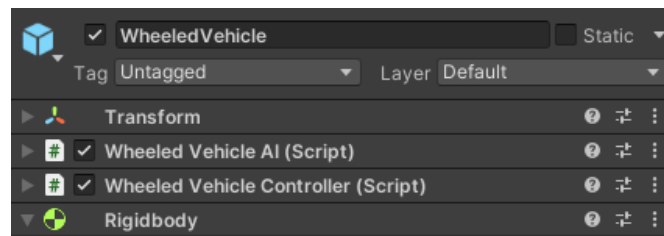
Setup

Vehicle Setup:

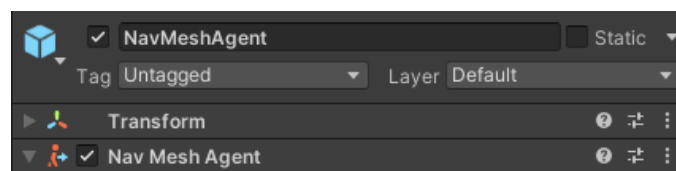
- Create two gameobjects under a parent object (*for grouping*).
- One is for the **WheeledVehicleAI** script, this controls the vehicles steering and navigation. The vehicle model and wheels should be children of this object.
- Add a **NavMeshAgent** to the other, this is used for generating paths for the vehicle to follow, to reach it's target destination.



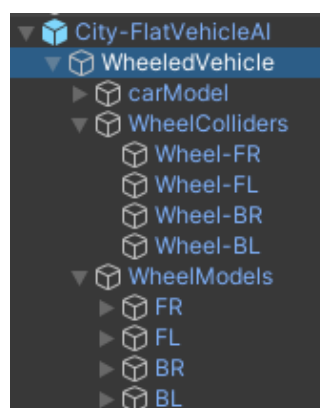
- Each **WheeledVehicleAI** script should have an attached **WheeledVehicleController** script, for controlling the vehicles' wheels, and a **Rigidbody** for physics calculation.

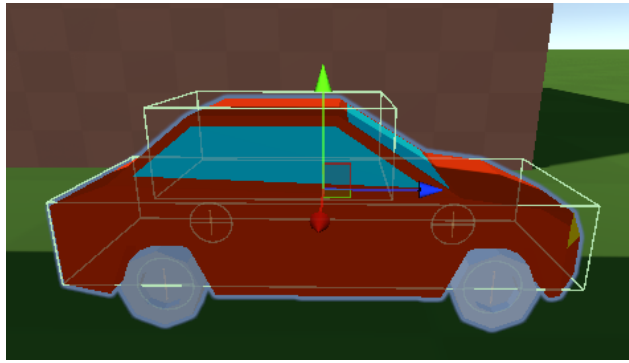


- It must also have an attached **NavMeshAgent** on a separate gameobject, for calculating paths on a **NavMesh** surface and guiding the vehicles' steering.



- The **WheeledVehicleController** script has to have wheel colliders and wheel models separated into different hierarchy's under the root. With it's own colliders preferably placed on the same root gameobject.



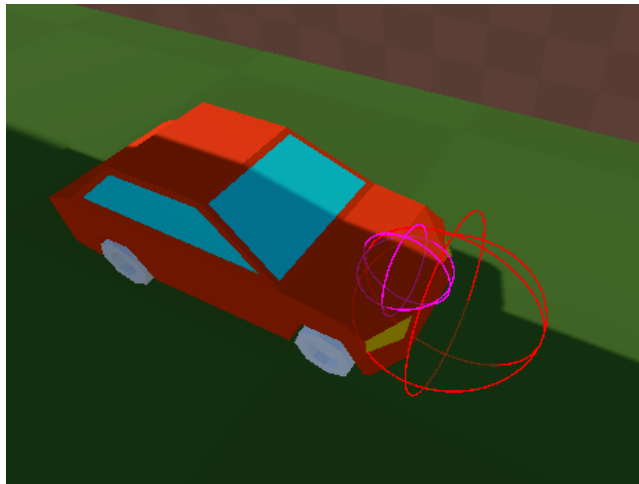


Using basic box colliders for the vehicle collision shape.

Gizmos:

The **Magenta** wire sphere gizmo shows the follow offset of the **NavMeshAgent**, this is the distance from the front of the car that the **NavMeshAgent** will try and maintain.

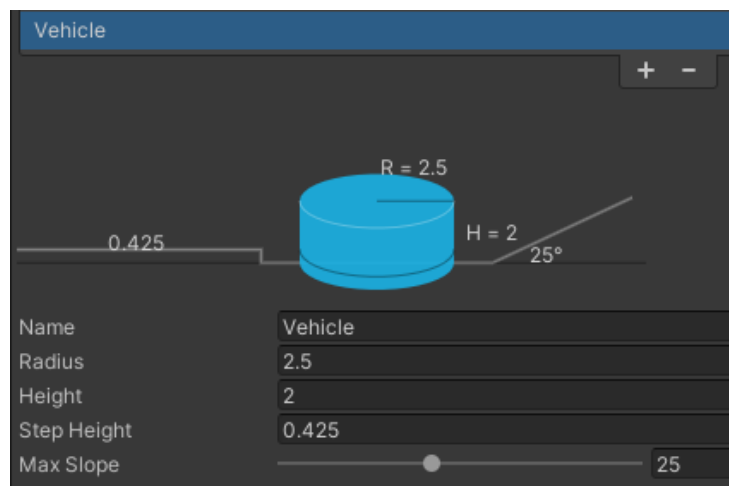
The **Red** wire sphere gizmo shows the radius of the positions sampled by the **NavMeshAgent** to determine if the vehicle is on a valid **NavMesh**.



NavMesh Setup:

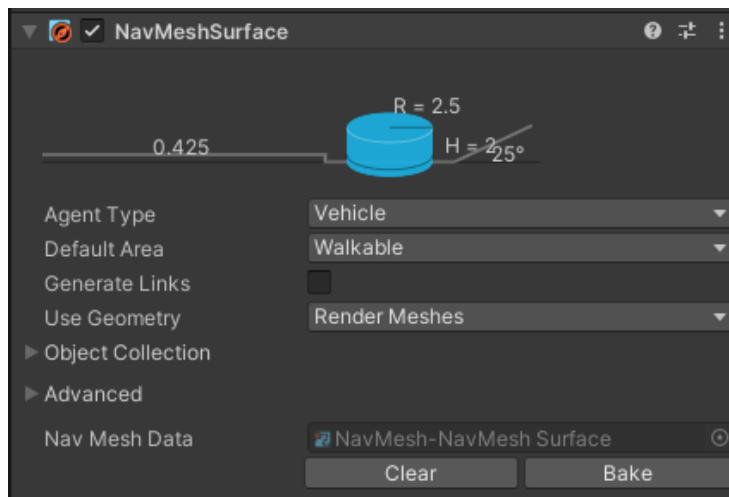
To use the **VehicleControllerAI** most efficiently, it is recommended that you create a new NavMesh agent type for each class of vehicle sizes you wish to use in your game.

In the agent type settings try to set the radius to at least the width of your vehicle.

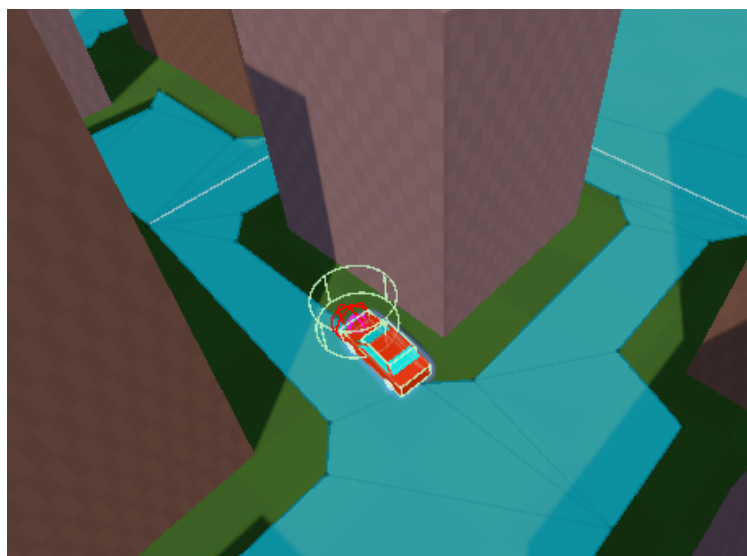


It is recommended to use the new Unity Navigation system (AI Navigation package).

In your scene, create a gameobject and add the **NavMeshSurface** component to it, and click the bake button.



Adjust the settings, particularly the agent radius, until your vehicle will not collide with walls or surrounding objects if it is right on the edge of the baked NavMesh.



Notes:

The **NavMesh** layer used by the **WheeledVehicleAI** is determined by the agent type in the attached **NavMeshAgent** settings.

Keeping the **NavMeshAgent** closer to the front of the car results in sharper turning, making it easier for the vehicle to turn into tight gaps. This can be changed with the **navMeshAgentForwardOffset** variable on the **WheeledVehicleAI** script.

If you want to use a different car controller, but still would like to use the NavMeshAgent for pathfinding and navigation, try these steps:

- Copy the **WheeledVehicleAI** script to a new script.
- Replace the reference to **WheeledVehicleController** with your own vehicle controller script.
- Create a new function that takes in two float values, one for acceleration input (0f to 1f), one for steering input (-1f to 1f), and a Boolean input for if the brakes are being applied. The function should call the appropriate methods in your replacement vehicle controller script to apply these input controls to the vehicle.
- Finally, replace all instances in the script for **vehicleController.UpdateControl()** with your new function, keeping the same passed in parameters.
-


A pro version with more, improved features is planned on being released. Please feel free to submit any feedback or suggestions!

Additional Resources:

How to setup the AI Navigation package:

- <https://docs.unity3d.com/Packages/com.unity.ai.navigation@2.0/manual/CreateNavMesh.html>

How to setup Unity's built-in wheel colliders:

-  Unity - Manual: Create a car with Wheel colliders