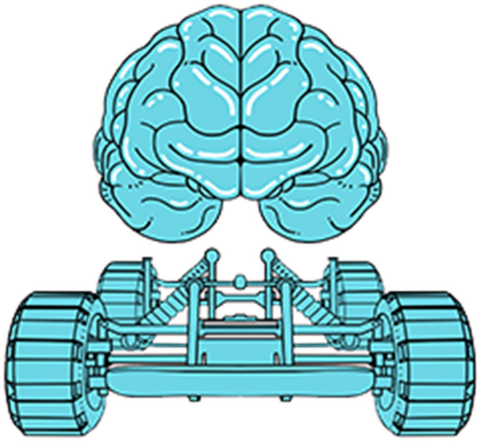
Vehicle AI Plugin



Documentation last updated: *15th April 2021*
Plugin last updated: *15th April 2021*

Documentation: 2.2
Plugin Version: 1.4

Contents

Introduction

Thanks for purchasing the Vehicle AI Plugin! To see how the plugin can be used in different use cases, please download the associated example project which is on the marketplace store page. If you do find this plugin and this documentation useful, then leaving a review is very much appreciated. If you have suggestions or requests for the plugin, feel free to send me an email to the address at the bottom of this page.

This documentation will show you the different features available with the plugin and how to use them. If you just want to get started and work it out as you go along, start with the Quickstart Guide. The system uses the default AI system, so after doing the Quickstart guide, just make your AI like you would with a character, and then configure this system to make the vehicle behave how you would want them to.

Plugin Support

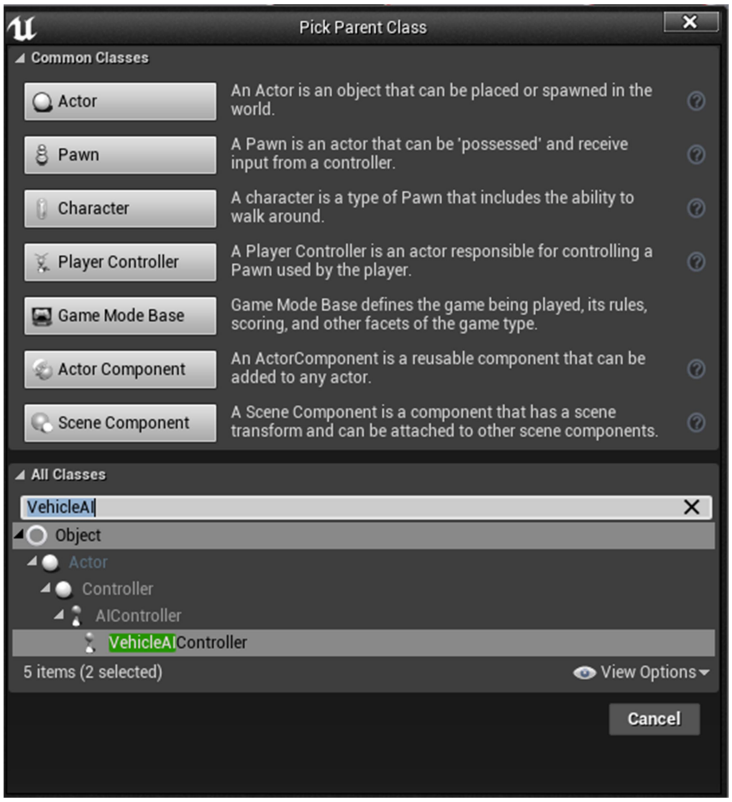
If you encounter issues using the plugin or want to understand a particular feature better, you should first read this documentation, especially the Frequently Asked Questions. If you want to know how to configure a setting, hover over the setting and it should give you a tooltip on what it does. Then consider if it's a problem with the plugin, and not your setup as I can only provide assistance for issues relating to the plugin. To test if that's the case, you could quickly create a normal character AI which if that also doesn't work, then it is probably an issue with your setup. I also can't help you set up specific behaviours as that is out of the scope of the plugin.

If you do need support, I try to respond as fast as I'm able to. I can provide better support if you tell me in detail what your problem is, and preferably have things such as screenshots and/or videos showing the issue you're having, otherwise it might be hard to debug. If you need support just contact me via the email below.

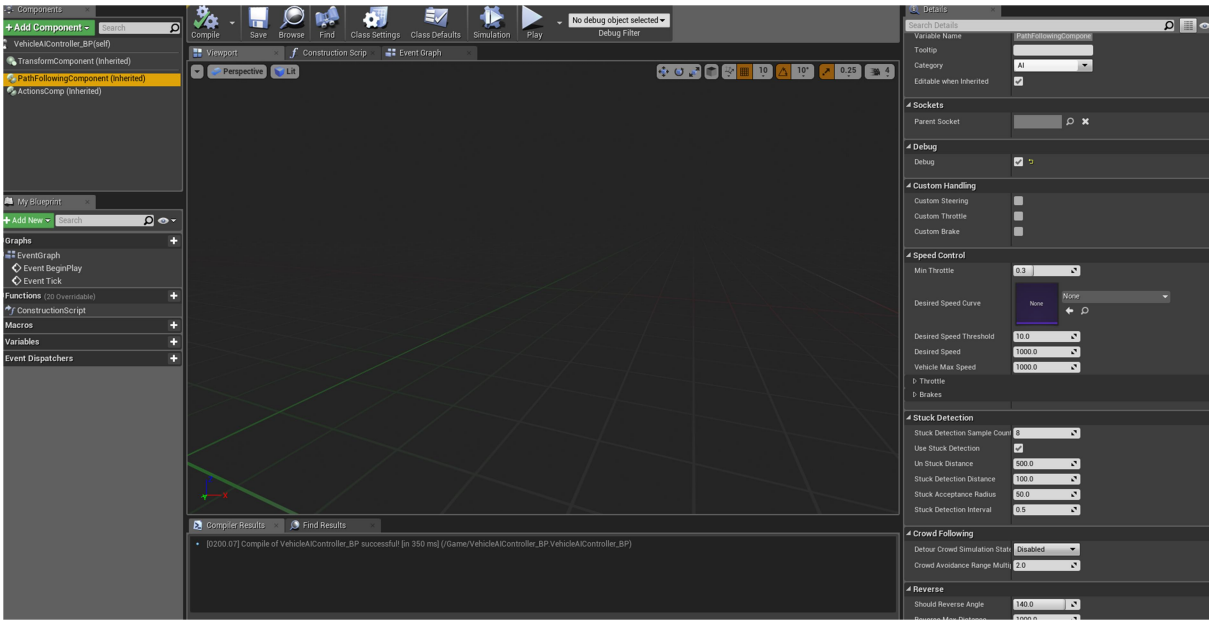
support@313-studios.com

Quickstart Guide

1. To use vehicle AI, you must first create an AI Controller that derives from the VehicleAIController.



2. Open the new custom controller and click on the PathFollowingComponent to see all the vehicle related settings.



If you are using standard PhysX Vehicles (like in the example project) that is all you need to do. You can do any normal AI activity like run a behavior tree on pawns that utilise that controller and it will work just like a character.

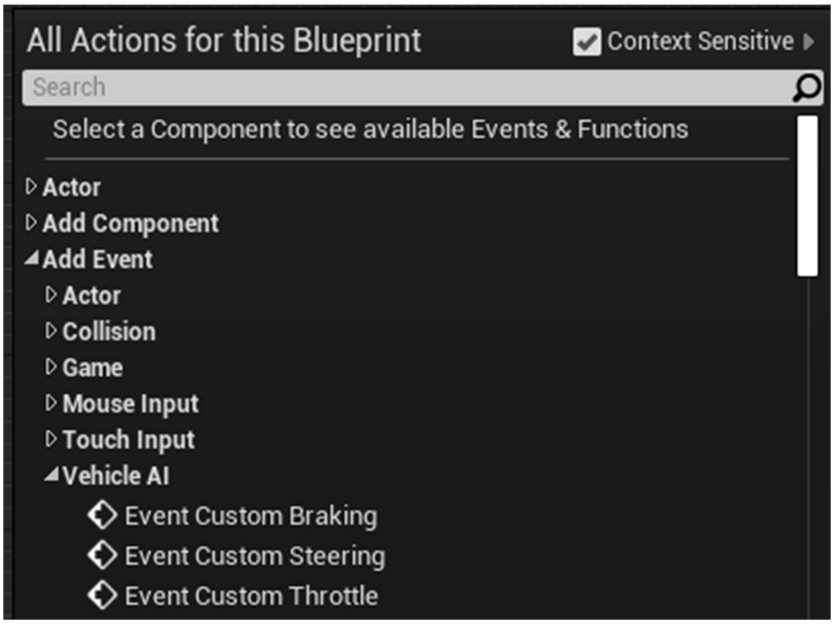
However, if you're using any other type of vehicle, you will need to use the interface and custom controls.

Custom Vehicle Setup

Note: I'm using another Marketplace pack, the Driveable Vehicle Pack to demonstrate how you could get this plugin to work with this, this isn't included within the plugin and you must purchase it separately if you want to use the assets within it.

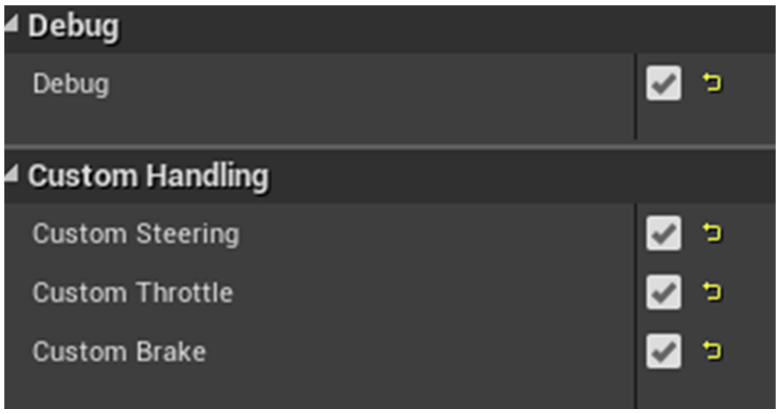
1. Implement the Interface within your pawn; this is done in the 'Class Settings' of your pawn.

2. Then add the events of Steering, Braking and Throttle to the blueprint through the context menu.



3. Then connect the 3 events up to the functions that execute that functionality. Below you can see how I've set up the steering. In this case I've added another event so both players and AI can control this vehicle. Then you should do this for the throttle and braking too. Each interface event has the calculated values that are generated by the plugin; it also provides other numbers which might be useful for doing your own custom behaviour.

4. Then in the Vehicle AI Controller tick all the custom settings, this will make the AI use the interface events.

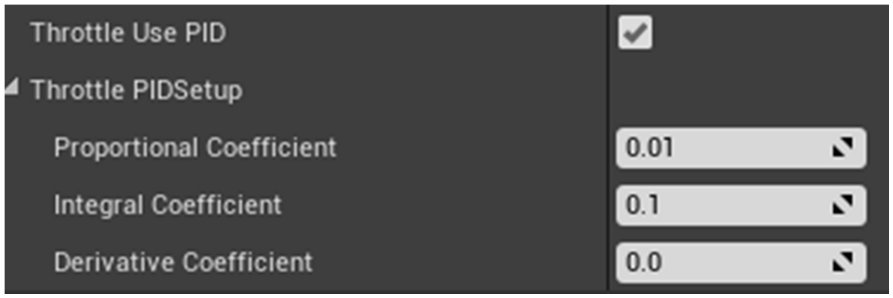


Configuring Vehicle AI Behaviour

With most of the plugin settings simply hovering over a setting is enough to explain its function.

PID Settings

For both the steering and the throttle the plugin uses a PID (Proportional, Integral, and Derivative) controller as default. This is a control system that tries to eliminate a difference between values. For example, the steering PID controller takes in the calculated steering value, and tries to reduce that to 0 by adjusting the vehicles steering. The speed at which it does this and the amount of steering it applies is determined by three values. The PID takes a sample every frame, and then these samples are used to calculate the output.



There are many online guides explaining how PID controllers work and how to configure them. The default settings for the plugin are usually decent enough for good results, and you shouldn't need to change them. Enable debug mode to see what value of steering and throttle is applied as a result of the PID controller, and then adjust the values accordingly to get the best results.

Avoidance

Implemented into the plugin is support for the two types of avoidance in Unreal Engine 4. These are used to avoid other agents, but both have their advantages and disadvantages. If you are considering to use avoidance within your project, you should read this section to find out which one suits your needs better, and then experiment for the settings to try and make it work better.

RVO (Reciprocal Velocity Obstacles) Avoidance

A common type of avoidance implemented into games, it's a simple solution that performs well and adds a relatively good level of avoidance. This is already implemented for vehicles in Unreal Engine, it works by stepping up or down the throttle and steering in order to try and avoid other agents. The main problem with RVO is that it does not respect the navigation mesh. This means that when it tries to avoid other agents, it might end up leaving the navmesh, which in some cases means colliding with other objects, or just leaving the navmesh and not being able to path back onto it.

Because of its good performance and the problem of going off the navmesh isn't as pronounced, its enabled for the racing example, however the steering rate had to be reduced from its default to prevent swerving and this also keeps it more along its original path. However in some situations it does collide with other agents still.

Detour Crowd Avoidance

Detour Crowd is a navmesh based avoidance algorithm. Unlike RVO, it respects the navmesh, and only will move agents to places that exist on the navmesh; otherwise it will just stop them until the agent it's avoiding stops getting in the way. This avoidance algorithm also supports obstacle avoidance based on the navmesh, which means it will try and avoid things in its way, such as the barriers on the racetrack example. This provides a much better avoidance solution and will generally look more realistic, but it comes with some problems, depending on your project setup.

- Costs more than RVO in performance (use CPU profiler)
- Crowd simulation only occurs on the main navmesh (generally not a problem as in crowd simulation agents offset their paths by their size)
- Because it's a more complicated simulation than RVO, you will need to tweak the associated values to get it working as intended

This is worth keeping in mind for your own project, as you will have to use the avoidance which you think will work with the limitations of the two systems. Although you can do, I wouldn't recommend using both systems at the same time.

Crowd Avoidance State
There are 3 states for the crowd following component.

- Enabled – Which avoids both agents and obstacles
- Obstacle Only – Only avoids obstacles, not agents
- Disabled – Reverts to normal path following behaviour

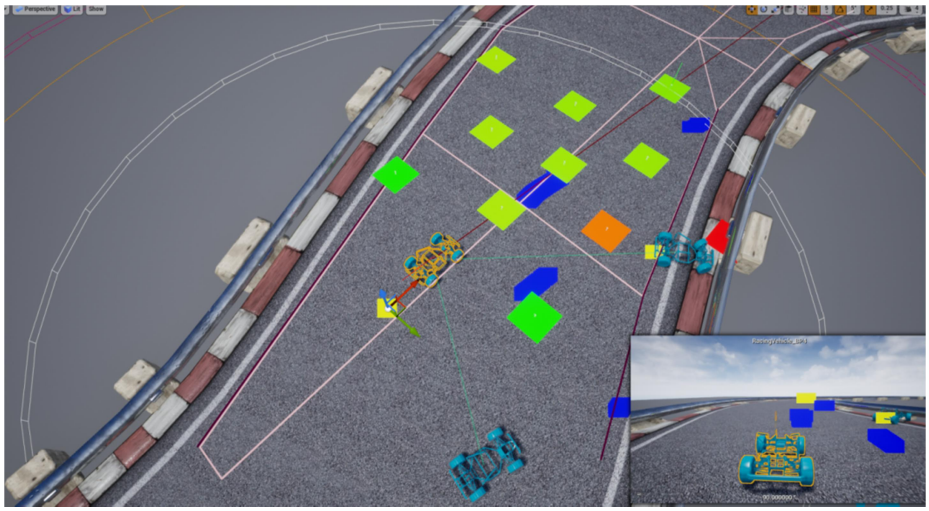
Avoidance Groups, Groups to Ignore, Groups to Avoid
These features are exactly the same as they are for RVO avoidance, and are not present on the AI controller. The crowd system will use the values you put into the RVO Avoidance section on the vehicle movement component, even if RVO isn't enabled.

Vehicle Max Speed
Max speed is used to adjust where the obstacle samples should test. This is usually the property that will need adjusting if you're not getting very good results with the crowd following. To test this you should enable crowd debugging, see the section below for more information on how to do this.

Debugging Crowds
To get optimal results using the crowd avoidance, you will most likely need to modify some of the settings.

Firstly type in the console command: ai.crowd.DebugSelectedActors 1

Then click on your vehicle actor. If the vehicle is moving you will also see the coloured squares that you see below. These are the avoidance samples; the closer to red a square is the more the algorithm will try and avoid it. It also draws a line to each other agent which is registered with the manager, this is the light blue line which is drawn from the selected vehicle to the other vehicles.



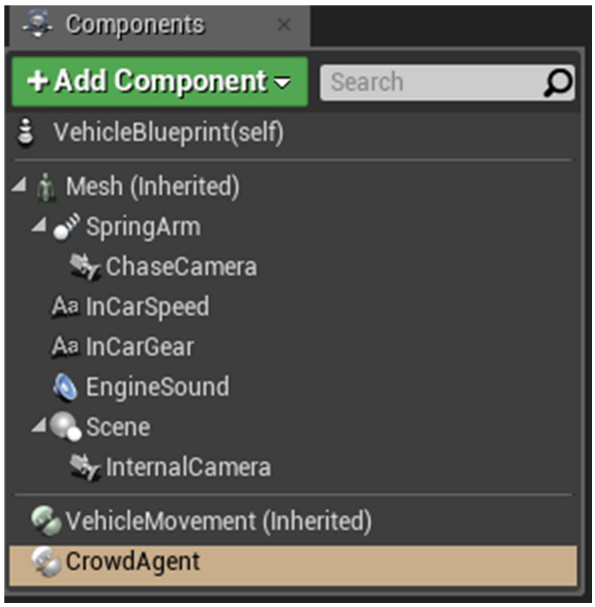
Crowd Manager Settings

In the project settings there is a section for Crowd specific changes. The main thing to change here is the Max Agent Radius, because we're using vehicles these are much greater radius than the normal character radius which is default value here. Setting this to 200 in the example project allows it to use the default vehicle. The other settings are self-explanatory, and can be adjusted depending on your project requirements.

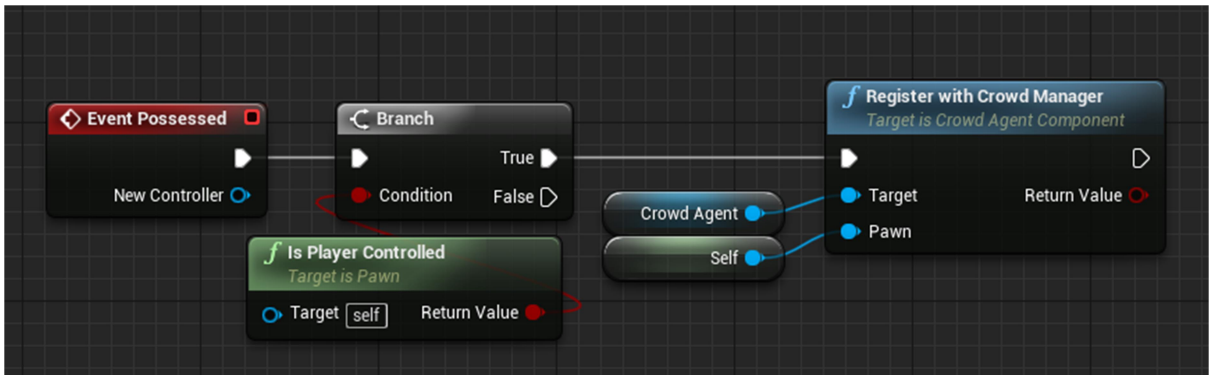
Crowd Agent Component

You usually have to have an AI controller possessing a vehicle for it to be registered automatically with the crowd system. This means that player vehicles don't usually get avoided, as they aren't integrated into the crowd. This component allows you to add player vehicles into the crowd using a registration function.

Firstly, add the component to your actor.



Next, add an Possessed event into the graph and register the crowd manager as done below.



Make sure you check if its player controlled before registering it with the crowd, as registering it twice when its already possessed by an AI controller might have unintended consequences. I also am using Event Possessed rather than BeginPlay; this ensures that there is a valid controller.

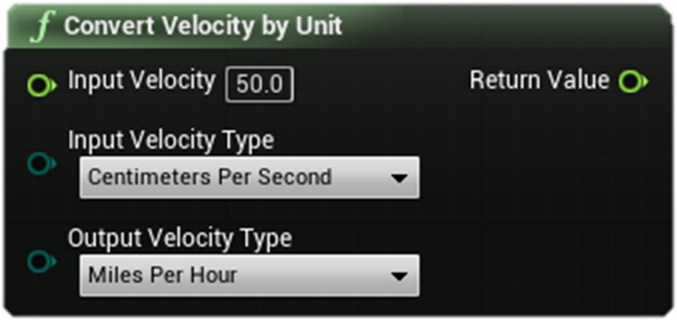
The function returns true if it was registered successfully.

Changing Runtime Values & Helper Functions

If you want to change the properties of the AI at runtime, the plugin includes some functions where you can do this. They must be called from the Vehicle Path Following Component.

Helper Functions

To access the Vehicle Path Following Component, you can use this node with a Pawn input, and it will find the component. Just keep in mind that AI controllers will only ever exist on the server, so if making a multiplayer game, this will always is invalid on clients! Use this to call functions which can change runtime values like speed, or force the vehicle to reverse.



This node converts the input velocity of a certain type, to a different output type, and then returns it. This is useful if you want to work with more understandable units than the default 'Unreal Units' (centimetres), which should be much easier to quantify. This is especially useful for things like the desired speed setting. Bear in mind all the functions that the plugin uses are in Unreal Units for simplicity, so make sure that you are outputting to the correct value!

The Tank Steering node returns two values to make creating tracked vehicles easier with the plugin, when also using the interface. It a good starting point for creating your own tank AI, and should work with the default way that tanks are driven by PhysX. The Turn in Place threshold is the threshold between 0 and 1 that the tank will turn on the spot. I'd recommend a value of about 0.5. The normal turn multipliers affects how quickly the tank turns when not turning in place, a value of 3 is a good start, but then check how your tank behaves.

Frequently Asked Questions

General

Vehicle Not Moving

- Ensure that your vehicle is possessed by a Vehicle AI controller.
- Ensure you have a Nav Mesh in the level (Press P to visualise it).
- Your vehicle should either derive from a Wheeled Vehicle Movement Component, or you should be implementing the interface.
- You have given it a path point on the navmesh to move to (Enable Debug in the Vehicle Path Following Component to see the path).

Avoidance causes vehicles to crash into walls

- If using RVO, this is one of the problems with it. Reduce the steering step value to prevent the vehicle from turning too quickly
- Refer to the section in this documentation to learn more about how to configure crowd avoidance

How do I change values at runtime?

- Most of the key values can be changed at runtime by calling the relevant functions. See the section earlier in the documentation to see if it's possible
- If you want to change values not changeable by the function, consider using the interface and implementing custom controls

My vehicle keeps stopping and starting at low speeds, but not at high speeds

- For this you need to tune the

Use Cases

How do I get my vehicle to follow a road?

- You want your vehicle to probably follow a spline. In the Version 1.3 (4.25+ only) example project, there are some examples on how you can do this. The example project demonstrates how to set this up in a way that gives you quite good performance.
- There are a few ways within that example on how you can achieve that. Which solution you use depends on your projects requirements
- The example is NOT comprehensive; it is designed as a starting point and demonstration. It will not support every possible use case. Some cases such as one way roads will probably not work as expected, as this would require modifying the pathfinder. Support isn't available for helping set up specific road use cases as this is out of the scope of the plugin.

How can I use the plugin to make racing AI?

- The example project shows how you can set up a race track and have a vehicle move around on it
- Depending on the type of racing game you want (Simulation, Arcade etc); you might need to use different aspects of the plugin. For example a simulation game you might want to use more precise path control as demonstrated in the road following example. In an arcade game, you might not need that level of control.
- You could have the vehicle follow a spline as in the road following example. The spline should be set up as a 'racing line' where it's the perfect route for the race car to take, and then depending on the ability of the car, you could decide how closely it follows that line.
- You should definitely use NavAreas as in the racing example to prevent the AI from cutting corners, but this would also mean if it goes off the track it can still path back onto it
- You might want to use the interface to really fine tune the steering and throttle systems, especially if doing a more simulation based game

Change log

Version 1.0
Initial Version

Version 1.1
- Added support for other platforms, and other engine versions
- Small usability fixes

Version 1.2
- Added Async Pathfinding function
- Added path point acceptance radius to controller so it can be customised per agent
- Added PID controller for better steering, throttle and braking control
- Added additional slowdown at destination logic
- Added 'Is Reversing' pure function, to check if the vehicle is currently reversing
- Fixed an issue which caused undesired unstuck behaviour
- OnVehicleStuck should be set to return true if wanting to implement custom stuck behaviour now
- Added Tank Steering node that can control tracks based on interface received data, a good starting point for basic tank AI
- Fixed a bug where toggling debug at runtime wouldn't work and a similar one involving changing stuck detection at runtime

Version 1.2.1
- Tweaked some default values to give a better starting point for adjustment
- Updated some tooltips to explain things better
- Fixed a bug affecting reverse speed being too slow
- Added a slowdown system, enabled by default to assist with cornering and also to prevent overshooting of a destination. Disable if you have configured your vehicles already and don't want to tweak this.
- Improved throttle adjustment so vehicles can achieve the desired speed and remain at it without it oscillating, this is very useful at lower speeds, and is a big improvement over the old system.

Version 1.3 (4.25+)

- Added functions
  - OnNewPathSegment – Lets the user change where a generated path point is placed (Executes only when a new path point is selected)
  - GetCustomDestination – Lets the user customise the current target destination the vehicle is heading towards (Executes every frame)
- Added SplineNavLinkComponent
  - Let's a spline act as a NavLink at both ends
  - Useful for handling complex areas for vehicles to navigate through, like a junction
  - If AI doesn't use the link, make sure you rebuild navigation (this appears to be rare and not game breaking. Will attempt to fix in the next version)
- Updated Example Project
  - Added a detailed and thoroughly commented road following example, with examples of expected use cases to demonstrate how the plugin can be used to enable these systems

Version 1.4 (4.25+)

Fixed

- Steering multiplier still affected the steering PID controller, it now is completely separate
- Slowdown was stopping at individual path points, when it should be slowing down only at the path end

Improved

- Throttle PID controller greatly improved and should handle speed control better at all speeds, previously only handled certain lower speeds well. The PID coefficients have been updated to reflect this
- Sharp corners are now handled better – Normalised turn threshold can set when to cut the throttle to the minimum throttle, and this can be much quicker to react than the PID controller
- Starting desired speed now can be set with units other than centimetres a second, the default unit is still cm/s to maintain backwards compatibility
- Tidied up some of the code and added some more descriptive comments which are also viewable as tooltips over the variables, this will still need improving in the future however

Added

- Splines following function – Ideally used for things like racing games where the NavMesh isn't that useful. This isn't fully working as expected yet but will be in a future release, and the example project will be updated to include an example of how to use this
- Physics based functions that can calculate the stopping distance and the maximum speed to go around a corner based on real physical properties such as the friction coefficient and mass.
- Emergency braking – If the vehicle cannot stop within the stopping distance using the brakes, it will apply emergency braking; which as default is 100% braking
- Slowdown at NavLink – The vehicle can additionally slow down at NavLink; this is enabled by default as it's assumed the vehicle might need to stop at these points. This has the accidental benefit of helping vehicle slowdown before junctions or difficult to traverse areas

Deprecated

Desired Speed Curve

This adjusts the speed based on the normalised steering angle, it's a decent solution. You can implement the exact same thing using the custom interface however, and because I don't believe this is very much used due to it being a bit confusing, it will be removed in a future release. If you do use this and want to continue using it, please do let me via the support email as if it's used enough I will keep it in.

Steering Multiplier

This will just be replaced with the Steering PID as it offers the same functionality anyway (on the proportional setting)

1.4 Update Notes

This is a relatively big update with lots of changes, it has been tested more than a normal release with the example project to maintain backwards compatibility however some bugs may exist still, please let me know via the support email if you encounter any issues.

PhysX & Chaos

With PhysX being deprecated in the next release, the plugin will move to using the interface only, this will remove dependencies for PhysX and Chaos, and therefore not require either plugin to be enabled. This will also benefit

people using neither vehicle system as it removes on plugin dependency. The setup of the interface is very simple anyway, so not much will change.

Advanced Speed Control

This type of physics based speed control is ideally what the plugin will move to over time, it's currently disabled as default to maintain backwards compatibility, but please do try it out (after setting up the mass for the vehicle in the settings) as it might give you much better results.

As default it only runs at 50% of the maximum corner speed that it calculates, this is so the vehicle will stay on the path as best as possible in most scenarios, but if you want the vehicle to go faster around corners and don't mind about staying on the path, you can increase this. The default value should be fine for most normal vehicles, but for racing vehicles this isn't ideal just yet. This is because radius of the corner is approximated and so won't give completely accurate results, and also vehicles handling characteristics will vary. This will be refined in future updates to give a more realistic behaviour, but in general use cases it should be much better.

Auto-Calculate Stopping Distance

The slowdown/stopping distance is automatically calculated now based on the physical properties provided as well as the default braking amount. This seems to give a larger space than necessary to decelerate and therefore is pretty smooth, similar to how you'd expect vehicles to operate in most situations, but this is because the default braking is 0.5. Increase this to 1 to get a harsher braking that is closer to what players do and would be more appropriate for aggressively driven vehicles.

General Notes & Known Issues

Async Pathfinding is not completely stable or tested extensively, it can still crash and might not be supported on all platforms, as it just utilises Engine functions, I cannot make necessary crash fixes without engine changes so its worth testing on your project and seeing if it works, if so it's a free performance boost!