DAGA STUDIOS

iRDS - Intelligent Race Driver System

Developer Manual – for Online API docs go to [http:\\irdsapi.dagagames.com](http:\\irdsapi.dagagames.com)

By Rhodnius
Created: 12/17/2012
Last Updated: 9/12/2016

Contents

Introduction

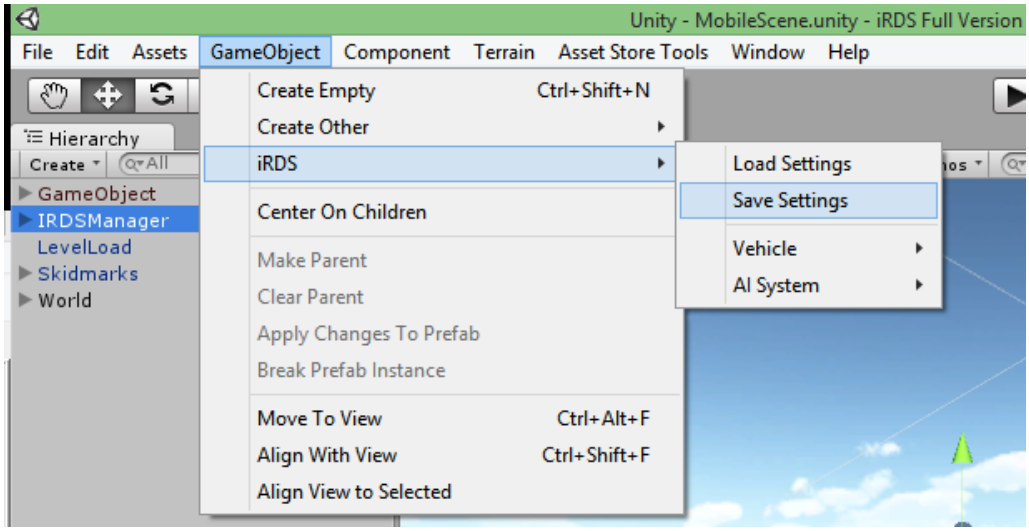
Intelligent Race Driver System is an application developed to control the cars on a race track and compete with a human on a race. The AI System has its own car Physics, which enables some features like max speed on corners based on tire wear.

WARNING: IF YOU ARE ON VERSION 2.1.0.5 YOU MUST SAVE YOUR CAR SETTINGS AND IRDSMANAGERS SETTINGS BEFORE UPGRADING TO ANY NEXT VERSION OF IRDS OR YOU WOULD LOSE YOUR SETTINGS!

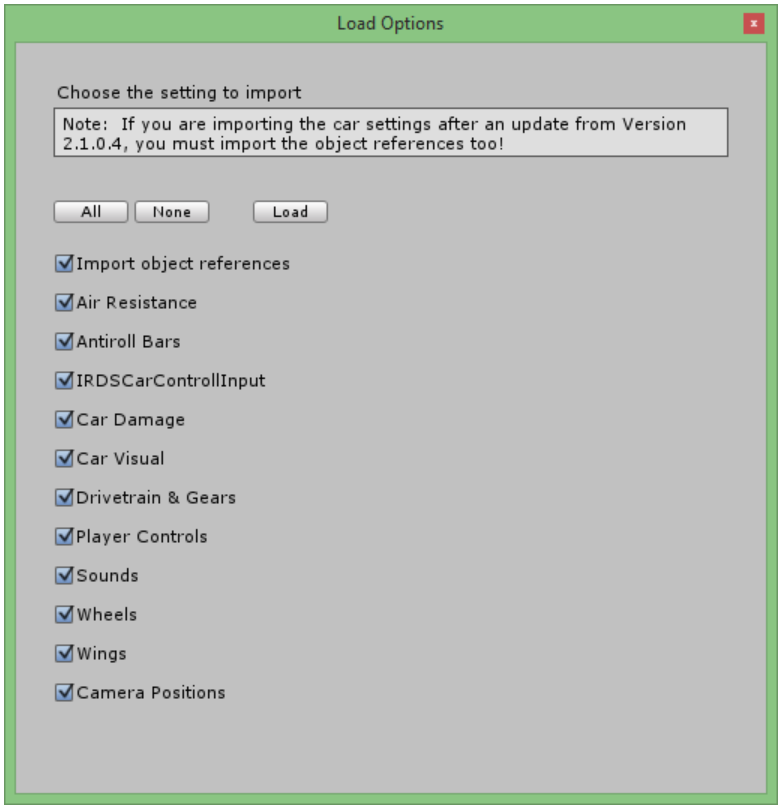
Upgrading note: Please Click once on each scene that have the IRDSManager on it, the "Process Track Data" button, this way the track would be ready to be played on the new version (this is only needed when upgrading from versions older than V2.x).

Save car and iRDSManager settings

For saving your car settings (which is required for upgrading from version 2.1.0.5 to any next version or you lose your settings) you just need to select the car prefab or the IRDSManager on the scene view and go to the GameObject->iRDS->Save Settings and it would prompt you to select the file name to save the settings to. See the following screenshot for an example:



For loading the settings you need to do exactly the same, and select the option “Load Settings”, if the object selected is a car prefab or a car on the scene it would show you the following window with some options:



In this window you can select which parts of the cars you want to load the settings and if you want to also import the objects references. If you just upgraded from version 2.1.0.5 to any next version, you must import the object references.

When you click the “load” button you would be prompted to select the file with the settings you want to load.

Features included on this version

Version Log

V2.1

Added rolling starts as an option.

Fixed an issue when using combined the always respawn and last man standing mode, that was causing that the DNF cars were always getting respawned after they got DNF

Relocated the respawn settings of the levelload object into the respawn / instantiate tab option.

Included some modifications to the championship system

Fixed an exception reported hen using Unity5 on the UpdateMaxSlips method of the IRDSWheel class.

Added the methods: GetCurrentTotalRaceTime and GetCurrentTotalRaceTimeString, which returns the total elapsed race time

V 2.0.2.2f

Added 2 new variables for the skid sound, skidEffectsThreshold and skidEffectsSensitivity.

Deleted unnecessary skid sound checks on IRDSCarControllInput class.

Corrected the AI behavior on the pits for 3rd party physics.

Added a new option for the Analogue inputs, Combined Axis Pedals, this enables to get the inputs from the axis from their raw values, so you could use the same

Axis for getting the throttle and the brake, useful for game pads with analogue sticks, like the Shield and PS Vita

Changed the way that the IRDSManager hides and show the markers, and improved the performance of the editor mode a little.

V 2.0.2.2.d

Added a new bool variable named runCameraInFixedUpdate to make the late update loop be run on the fixed update

this is useful to avoid some jittering when in networking mode.

V 2.0.2.2c

Fixed a bug that was only shown if the scene view was not focused.

V 2.0.2.2b

Fixed a bug about showing the wrong way label and signal

Fixed the points shown on Championship system (included on the asset store only)

V 2.0.2.2a

Add button to enable disable the racing line, maybe on the levelload object as an option

Add some handles to the spline to treat it as Bezier curves

If you clear all track data the waypoint numbering does not reset and continues using the last higher waypoint from previous data.

iRDS V2.0.2.2

Added new callbacks to the IRDSCarControllerAI class (onUnstuck, onRespawn, onRaceEnd, onRaceStart and onLapCompleted) you need to register your methods

on the start or awake methods on your custom script.

Fixed an issue with Edys and UnityCar or any third party physics that avoided the AI to do overtaking and avoid side collisions correctly.

Improved the code for the car engine on the IRDSDrivetrain class

Added new championship system.

Fixed when cars get obstructed they pull back in a very crazy manner.

Added buttons support for steering, throttle and brake for the analog inputs.

Added a new method for starting the race manually on the IRDSStatistics class and the option on the levelload to enable it (startRaceManually)

Added visible tire lateral and longitudinal curves for better tweaking the curves (using animation curves for showing them only, not for editing).

Fixed an issue when inserting new markers for the racing line and the track limits

Added a delete marker option on IRDSManager to improve the workflow, you can now delete the markers without having to select them.

Fixed the clear track data, now it resets the markers numeration.

iRDS V2.0.2

Fixed issues that didn’t allowed to compile on windows phone 8

Some car physics improvements

iRDS V2.0.1.9

Fixed a bug for the RemoveCar method on the IRDSPlaceCar class that was not removing the car from the IRDSManager drivers list

iRDS V2.0.1.8

Added default audio source properties to the extra sounds, so you can set them up

Added Doppler level settings to gear, backfire, blow off and tire bump effects

Fixed the suspension compression when going at higher speeds

Fixed an issue when selecting a car prefab on Mac that throw an error

More stable suspension system now, you may have to increase the spring rate force, since now the rate of the spring

is calculated correctly

Made some improvements on the calculations of the slipratio and slipangle, now they are more stable too

iRDS V2.0.1.7

Added new variables to be able to better setup the automatic shifting.

Added handbrakeTime and handbrakeReleaseTime to the playercontrols class to fine tune how fast the hand brake is applied

Improved the revlimiter of the engine, now it limits the engine rpm if you downshift too early

iRDS V2.0.1.6

Added a better way to load all cars when you select a car prefab for using for the copy functions. (Avoids reading everything from the resources folders at once)

Fixed performance issues with the sound controller class

Added the ability to put the markers upsidedown, so now you can make iRDS work with twists and loops, as the ones that can be made with TrackBuildR, we have tested it using TrackBuildR

Added new variables to have more control over the heel_toe settings and the auto throttle for shifting features.

Added shift up and down events on the sound controller class

iRDS V2.0.1.5

Added RPM Limiter to the engine class, now you can define where the engine cuts-off

Improved the engine code, now it feels more real.

Added Auto-resize option for the mobile controls layouts.

Added the option to make a forced Throttle lift up when shifting.

Added Heel-Toe downshift auto technique.

Added a change layout method for the mobile controls to work at runtime

Fixed a bug on the manually instantiated cars that are not getting passed the settings of overrideNitroInput

Added the option of Unique Opponent Car on the race

Now the minimap is totally deactivated and shot off when SetMiniMapsActive(false) is used.

Added Copy properties functionality for tires settings

Added Copy properties functionality for wing settings

Added a validation to Verify if there is another car on the respawn point, to avoid cars collision and explosion respawns.

Fixed bug about when changing the track center offset not updating the track automatically and not enabling the process all track data button.

Added wheelie multiplier to make the cars do wheelies easier and for bikes too

iRDS V2.0.1.4

Fixed an issue that if the checkpoints are deleted manually the IRDSManager custom inspector throws an missing reference exception.

Fixed minor bugs on capture the flag race mode.

Fixed some Capture the flag bugs and improved the AI behavior on this race mode.

iRDS V2.0.1.3

Added the option to choose between KPH and MPH for the digital odometer

Added the ability to see the power and torque curves of the cars on the engine settings (shown using animation curves, these are currently not editable)

Moved some initializations from the Start method to the awake method on the IRDSDrivetrain class to avoid overriding values that are set by the SetMaxTorque and SetMaxPower methods in runtime.

iRDS V2.0.1.2

Fixed bug when using last man standing race type in lap knockout mode not making the cars to get disqualified correctly.

Added the feature to "pre-load" the cars on the resources folders into a string array to avoid loading them all into memory at the beginning of the race, this would help a lot on mobile device with lower memory capacity.

iRDS V2.0.1.1

Fixed bug when importing the markers from easyroads that causes a missing component error.

Version 2.0.1

- Fixed minor jittering bug on the camera when using the min and max Filed of view with a change multiplier higher than 1

- Fixed a bug that makes the cars on the pits to get re-spawned on the main track.
- Fixed a bug that makes some cars not to pit on their pit point, and instead they pit on the pit lane
- Add an option on levelload to not instantiate cars, to be able to make single and multiplayer games easily
- Fixed a bug that makes the front tires not been detected after a new car is rigged.
- Added new sounds options for engine dependent sounds to be able to use animation curves to blend the pitch and volume, this is very useful to make more realistic engine sounds.
- Some performance improvements on the overall of the car physics engine.
- Added more options to the camera, to be able to add some external effects to the camera.

Version 2.0

- Added speed limit to be changeable, by adjusting the hill radius on the Race Line waypoints, now you can put on any of those waypoints the maximum speed for the AI's and override the speed calculated by default by the system, this is useful for adjusting and tuning the max speed on corners that are more complex, like chicanes, or sharp turns.
- Also now the racing line in edit mode shows in color red the areas of slower speed and in green the areas of higher speeds.
- Added a visual gizmo to the selected racing line waypoints at the moment of editing the speed limits
- Fixed touch steering sector size to be restricted to max value of 1
- Fixed minor bugs on IRDSManager process track data and on IRDSMarker process track data
- Added option to use the accelerometer settings for the touch steering
- Fixed auto shifting not changing gears up on mobile control
- Fixed some errors thrown on newer version of Unity and in windows phone 8
- Fixed touch steering not working properly when using other buttons at same time.
- Fixed shifting for mobile going to fast if buttons keep pressed
- Minor performance improvement on AI
- Fixed Last man standing races not working if the track was loaded from another scene
- Add new suspension system that can be used instead of the old one (is an option, you can choose to use the new or keep the old) This feature is experimental, and is intended to get some performance gain.
- Fixed the IRDSStatistics class not updating the value correctly of the variable racePosition on the class IRDSCarControllerAI if the standings are disabled, causing also the last man standing race mode to not work properly.
- Improved engine shake on the cars
- Added the ability to enable or disable independently the On race standings and the end race standings.
- Fixed the sandgrass setup on the skidmark object not been applied correctly.

- extra sounds keep playing if engine is off, added a bool to check engine on and off
- auto gear - downshift too late, fixed timing
- sound - there's no "play One shot" based on "GetButtonUp" - fixed, added playoneshot option
- Fixed initialcounter object getting disabled when deactivating it and clicking on the button hide Gui
- Added new shifter gear buttons setups for the cars.
- Auto throttle now is independent of auto Shifting, if you want to have auto shifting when auto throttle is enabled, you need to enable the Automatic Gear shifting feature.
- Fixed a null reference exception if IRDSManager was selected on play mode if the initial counter was disabled
- Fixed disable and enable not been effective on play mode for the standings and the minimap.
- Fixed LabelOBject position
- Fixed Enabling minimap or standing on runtime (when it was disabled originally on IRDSManager) for causing some null reference errors
- Added masterVolume variable to the class IRDSSoundController for easy mute all sounds.
- Fixed some bugs on gear change for the gear shifter and on the clutch
- Added a new method on the class IRDSCarControllerAI for getting if the Car AI or with AutoThrottle on is stuck
- Added support for 2 wheeled and 3 wheeled vehicles
- Fixed Waypoints advance bug. The waypoints when the car was going back were resetting to 0.
- Fixed issue when placing the markers on the IRDSManager object
- Added property name of the shader used for the brake lights, so this way you can use your own brake light shader for the tail lights.
- Added a bool variable so you can choose to use the input controls from the car or levelload object, it is on Level Tab on levelload object and is named “Use Car Inputs?”
- Add a flag to make the player car unique per race
- Changed the new mobile controls to use GuiTextures instead of OnGUI with exactly the same functionality it had, so no difference would be noticed.
- Fixed on the car physics the brakes, now the car does brake, this could affect how the AI would take turns now, and they should be able to brake now on shorter distances.
- Added 3 new variables to levelload object, 2 bools to activate the always re-spawn option, so the cars if get stuck would be re-spawned instead of unstuck, and a float to set the speed at which they get spawned.
- Added additional custom buttons for mobile, you can set on them a target game object were a message would be sent, and the function it would send the message to, also a parameter (float, int, string, gameobject, vector2 3 and 4).
- Added copy function to suspensions settings

- Added new variable to the engine, so you can control the engine torque applied when cranking and revving up with a new multiplier. This is very handy when using cars that are very small like RC cars.
- Fixed the new GUITextures position changed if the level is restarted and only on levelload object is used on the entire project.
- Fixed camera preview mode not working properly
- Introduced 4 new variables to the tire physics models, now you can set the slip ratio and slip angle max clamp value, and also set the longitudinal and lateral forces multipliers, this are quite handy for arcade games, and also for fine tunning the behavior of the tires and the car.
- Also we exposed on the tire setup the variable wear, which is used to enable or disable if the tire wears with burns, this is in case you are using the car physics alone.
- Fixed for the player to force brake at the beginning of the race before the race starts, to avoid the car for moving forward or backward on non-flat surfaces.
- Better obstacle avoidance and new obstacle Sensor, which is a box collider, which is added to the main car object automatically if there is any obstacle on the track (obstacles needs to have the IRDSCarControllerAI script on them and be on a different layer), for better getting it working, you would need to put only the main car object in a different layer for all cars and set this layer in Physics Manager to only recogninze as collision the layer asigned to the obstacles.
- Added a new variable for the look ahead distance, this is called LookAheadConst and it is on the driver settings, default value is 10, as it have been always, use smaller values for smaller vehicles, like RC cars.
- Added new variables for controlling the throttle when out of the tack (offTrackThrottleMultiplier) and when the AI have performed a jump (jumpThrottleMultiplier) and the time it would be applying the Jump Multiplier (jumpThrottleTime) all of them are in the AI Driver settings
- Added clear track data button on IRDSManager to easy clean all the IRDSManager data (does not have undo)
- Added 5 variables to the IRDSCarCamera class, Override Position, Override Rotation, Override LookAt, Override FieldOfView Override Camera Collision, this allow an easy way to manipulate the camera, and still having the benefits of using the IRDS HUD or find other way to customize the way the camera follows the player.
- Added some null checks on the IRDSCarDamage script, to avoid null referenfe exceptions to be thrown
- Added 1 variable to override the nitro input on IRDSPlayerControls Class, this is usefull when yo want to control the activation and deactivation of the nitro, to make games like Mario Kart.
- Added 1 variable to override the Nitro Input for the AI's on the Class IRDSCarControllerAI
- Added 2 variables on Levelload object to override the nitro input for AI and player independently

- Fixed the bumps factor not affecting on the new suspension system
- Added a new default null physics material settings on the skidmarks object, this one is especially implemented for the new version of unity 4.3, since the terrain collider no longer contains the physics material on it, this way is possible to have the particle effects on the terrain for unity version 4.3x
- Added 2 new variables in the IRDSCarCamera class to allow more camera positions costumizations it is called distanceSides and sidesDamping, the first one is a vector 3 that sets the camera position with respect to the target and the second is the damping factor.
- Turn off engine from script, we made a method for killing the engine on the IRDSDrivetrain class, is called TurnOff()
- Made the player car to be still on the race starts
- Fixed some issue on the pits stop method, now the cars make a better pit stop
- Fixed the auto start engine when using the car physics alone without the AI system, now if you reload the scene the car would start again.
- Added a new variable to the CarDamage Class, which holds the total sqr Magnitude force from collision and gets reset if the car is repaired.
- Fixed the start engine on start for standalone car physics.
- Added some checks on the car rigger wizard to avoid the selection of the car parts outside from the car on the scene view.
- Simplified slip angle formula.
- Fixed the rig car wizard that was not saving the prefab when rigging bikes.
- Fixed differential lock not affecting the tires torque, now it is affecting it correctly, so if differential lock is at 100% all tires that are powered spins at same angular velocity.
- Fixed the mass fraction auto calculation for vehicles with less and more than 4 wheels.
- Added new features for the motorcycles type of vehicles, now the fron wheel rotates with the steering handle, so it looks nicer and feels better too.
- Fixed some bugs on the pits stop, now the AI cars would stop smoother at their pit stop point.
- Added new get and set methods to the IRDSCarCamera class, for the Height, Distance, HeightDamping
- NEW** Added new suspension system simulation, now you can choose between 4 types, old, Basic, MacPherson and double wishbone. For this to work you need to setup the hub, hinge, top strut, etc positions to emulate the chosen suspension, all position are relative to the car, also we added a visual lines while you play in editor for make it easy to see your changes live.
- Added new methods to IRDSPlaceCars class so now you can instantiate the cars and then add them to iRDS by using AddNewPlayerCar() Method, instead of using the internal instantiate of iRDS, this could be useful for controlling were you want the car to be instantiated.
- Fixed some minor issues on the pitting methods that make the cars not pitting as they should.

- Fixed some bugs on the new suspension types, and made them more performance friendly by making some improvements.
- Added a new option for the last man standing race type, now you can setup the dnf cars to be set in another layer, and you can manually setup that layer to not collide witht he layer of the cars that are still racing, this would make the dnf cars to be invisible for the cars that remains racing, and they can pass through them.
- Fixed bugs on the pits stop of the AI, now the standings are correct while on the pits, and the AI behavior is also corrected while in the pits
- Added a new method on the IRDSStatistics class (GetAllDriversList) for getting all drivers on the race, this one is much faster than previous GetAllDrivers since the original variable is a list and not an array, and no conversion is needed. Also you can now grab the reference only once on the Start method and it would get updated automatically since it is a reference.
- Changed the way the mobile controls settings are shown
- Fixed minor bugs in the engine clutch and shifting
- Fixed minor bugs in the engine clutch and shifting that were still having some issue
- Changed the size calculation of the cars from getting them from the renderers, to the colliders, since this are going to be the real boundaries of the cars, this would also avoid the cars to get stuck to each other, when the renerer bounds are bigger than the colliders bounds.
- Fixed minor bugs related to the enable and disable car damage for the other players (second and third and so on)
- Added a Get Values button for the new suspension to get default values to start tweaking from for the MacPherson, wishbone and basic suspensions, so it could be easier to get it setup
- Fixed some bugs when adding player and ai cars at runtime that made the Obstacle trigger senor to be added without any obstacle on the track
- Fixed minor bugs that duplicated the items on the Cars list for the Camera targets and the IRDSStatistics.
- Now the methods AddNewPlayerCar and AddNewAICar from IRDSPlaceCars class can be used on the awake method.
- Added 1 new method on the IRDSPlaceCars Class AddOpponent(IRDSCarControllerAI newOpp) this new method is usefull for registering new opponents when making a network game, for the clients to register the other cars that are added by the master host.
- Added integration of other car physics, now you can use Edy's and UnityCar car physics engines directly with this Full iRDS version, you would be able to even use all 3 car physics at the same time if you want to, we are including the necesary scripts to make the interface from Edys to iRDS and from UnityCar to iRDS, they are on the compressed file named "ExternalPhysicsScripts.rar"
- Fixed a bug on the Tire code, that made the car to not move forward if the steer was all to a side and trying to go forward from a full stop (car physics side)

- Made some minor changes to the core code to make it compatible with networking (Tested with photon).

Version 1.09

- Auto steering
- Auto throttle
- Auto brake
- New custom inspector for the IRDSLeveLoadVariables class (make it more user friendly)
- Add Nitro to the cars (the player and AI can use nitro if you want on the race)
- Adding players dynamically, add cars to the first available Grid position
- Fixed duplicates markers when placing them on editor mode
- Removing players dynamically on the race (useful for making last man standing races)
- Ability to add more than one player (the system by now doesn't handle split screen and multi audio for multiplayer)
- Multi editing the width for multiple markers.

Version 1.08.3

- Disable Camera Follow During start of the race Delay Time
- Delay Time Between Scene Load and Game Start
- Change Left and Right Track Limits and Widths independently
- Functions to Get Player Audio Sources
- Customizable GUI through Array (Add new labels and Textures)
- Change the countdown number
- Zooming values min and max for camera road camera mode
- Better loading performance of the levels, the heavy calculations are made now on edit mode.
- Three Wide Racing Style
- Add Methods to let know if the AI is overtaking or avoiding others
- Improve Overtake and obstacle avoidance
- Improve Ramming issue on collisinbrakeprevention

- Camera views, stay on same point when changing vehicle
- Make methods for laptime that returns the string
- Add static friction to the car physics
- Add clutch feature to the car engines
- Add start engine option, automatic or manual
- Add sound effect for starting the engine
- Add volume multipliers to all the sound effects of the car
- Add tire bump sound effects
- Make the skidmark and skid smoke to be tweakable their thresholds
- Add method to change camera target by instance ID of IRDSCarControllerInput
- Change camera target by position order
- Add variable "wrong Way" to know if the player cars is going backwards
- Add a multiplier to the fuelconsuption
- Add wind sound effect
- Add break squeak sound effect
- Add Scratch sound effect
- Add grass sound effect
- Add pitch min and max variables available for setting up for all sounds effects that required pitch
- Add scratch visual effect (sparks)

Version 1.07

AI System

- Compatible with Edy's Car Physics
- Compatible with UnityCar Pro 2.2
- AI steering
- Breaking
- Overtake
- Braking if opponent in front
- Collision prevention - if opponents on sides
- Track limits (left and right limits of the track)
- Different AI behavior
- Pitting
- Counter steering if the car drifts
- Compatible with Easyroads, the system creates the track limits from the EasyRoads marker and road width information
- Racing line
- Grid positions
- pit line
- Road cameras position (for using on road camera view mode)
- The AI cars are configurable about the following:
    1. You can set how fast they would take curves by changing a single value
    2. You can set when the AI should enter pits by changing 2 values, min fuel to go to pits (value from 0 to 1), min tire wear before enter pits (value from 0 to 1)
    3. You would be able to save AI drivers, so they could be randomly chosen for race

4. You could set how aggressive the AI would be braking on curves, by changing a single value (this would make the AI brake earlier or later)
5. You could set the overtaking capabilities of the AI, like how much faster than the opponent it should go to overtake it.
6. The AI are capable of avoiding side collisions with other cars
7. Configurable sensitivity of the AI for making maneuvers (You could set amount of overtaking maneuver, side collision avoiding and getting out of track avoidance)
8. Set the Fuel tank capacity
9. Set the distance for the AI to look forward on another cars to make avoiding maneuvers
10. Set the min sides distance to make avoiding maneuvers from other cars
11. Set the min Height distance to lookup on other cars to avoid them
12. For each waypoint, you could be able to set an speed override for the AI to Slow down on some part of the tracks
13. Counter steering if the car drifts, set this value too.

- The max speed of corners is automatically calculated from the race line
- The pits includes two sections where you could set different max pits speeds, for example, the entrance could have a max speed of 80Kph and when the car enters the pits itself, the cars (including the player's cars) could only drive at a maximum speed of 50 kph(you can set the desired speed limit also).
- The system sets the positions on the race grid randomly
- The system assigns each car a pit stop available.
- The system includes its own physics (to support the tire wearing and the automatic calculation of maximum speed of corners).
- When the race is finished, the AI system takes control of the player car
- Choose if the race would use pits or not.
- Also works on sprint race Tracks (single laps too)

Car Physics

- The Tire wears during race, by making the car loose grip and heats (for maximum grip)
- The fuel is consumed during race
- All the regular Car Physics features (Springs, Dampers, Tire curves, Engine Torque, Engine Sounds, Engine power, Turbo, Brakes, Brake Lights, Steering, etc.)

Version 1.07

- Added copy functions on the car setup for engine, gear and tire values, now you can copy those values from other cars.
- Improved Breaking method, now the AI break earlier on entering curves.
- Added key assignment on the player controls, so you can select the ones you have or want to use.
- The method FlipCar is now public, so you can reset the car by scripting.
- Corrected standings not showing correctly by the end of the race.
- Performance improvements.
- Implemented different race flags (yellow flag, blue flag, drive through pits).
- Improved overtaking on straights.

- Speed restriction with turning the yellow flag on, you can restrict the speed on some parts of the track or the entire track.
- Added a variable that hold the fastest lap time per car.
- Added per car HUD, you can now choose between using the global HUD and use specific HUD for a car.
- Added new methods for applying changes on real time to car engine, gear settings, tire settings and suspension settings while in play mode. You can both, change on the inspector on the fly or by scripting in game. (please see this methods at the bottom of this document).
- Added a menu option for creating predefined tire settings, that you can choose when setting up the cars on the inspector window, so you can have same settings for different cars easily, also those predefined tire settings are saved as prefabs under Tires folder in the resource folder, so you can access them by scripting also in play mode to have those tires as in game tires to choose from.
- Improved Electronic Stability control (ESC).
- Improved Help Steer.
- Improved the air resistance script.
- Added the variable initialPreviewOnRace on the levelload object to be able to disable the preview on race.

Version 1.06.2

- Fixed automatic transmission gear shifting.
- Added Shift Up RPM % and Shift Down RPM % for setting the shift up and down if automatic transmission is enabled.
- Added variable Automatic Transmission on Level Load Object to be able to set up this on level selection.

Version 1.06.1

- Improved Car physics.

Varsion 1.06

- Fixed the camera script, in order to be able to use the preferred camera view.
- Added respawnTime Variable on LeveLoad Object to set the respawn time.
- Added unstuckTime Variable on LeveLoad Object to set the unstuck time.
- Added respawnAtWP Variable on LeveLoad Object to activate respawn at waypoint.

- Added the method SetUnstuckTime on IRDSCarControllerAI Class.
- Car Physics improvements.
- Added the following methods for Force Feedback Support on IRDSWheel Class:
    - GetGrip
    - GetNormalForce
    - GetLocalVelo
    - CalcMz
    - Added the global variable cCoefficients for the Mz calculation
- IDSCarControllerInput – added the method GetCarPilot()
- IRDSDrivetrain added the method GetMaxRPM() of type float
- IRDSRaceLinerRendererUpdate - Fixed first lap race line not changing colors correctly for player car.

Version 1.05

- Included 3 objects with the tags RGT LFT STR.
- Added the options to select to either use re-spawn or not.
- Fixed the ESC Filter to make it work correctly.
- Improved Wheel Script for better Car Physics.

IRDSCarControlInput

- added public void SetRespawnTimer(float time) - Done
- added public bool respawn = true;

IRDSCarCamera

- Added support to the camera follow the CameraViews Objects instead of the car
- You can now add or delete the objects that are children of CameraView object on car, so you can customize those.
- Added on the Class IRDSCarCamera a method to get back to players cars at once, this is helpful when the player just cycled on opponents cars to see them in action and want to get back the camera quickly to his car, see CameraControll for an example (it is assigned there to keyboard key "B") the method is called ChangeToPlayerCar().

IRDSLevelLoadVariables

- Added the public variable bool respawnActive = true;

IRDSCarVisuals

- impproved the Skidmarks and Skid Smoke emissions.

Version 1.04

- Fixed Lap Label and position label GUI not changing the entered text and font type
- Updated overtake method, now it overtakes better
- Improved steering for better counter steering maneuvers
- Fixed Elapsed Time not registering laps
- Fixed camera position change, so cars without cockpit can delete the positions and it wouldn't throw errors
- Fixed the Silence Method in the Sound controller for the turbo and blow off sounds.
- Fixed color change on race start to use the selected by the user.
- Fixed Statistics not loading correctly after loading scene on second time
- Fixed the set of control Digital including two methods, one for setting its value and another to get its value
- Added Obstacle Avoidance.
- Fixed the issue when the car is at full throttle and then suddenly the gear is put in N and the car won't stop going on.
- Added multiple object edit support for setting the Hill Radius on the Race Line waypoints.
- Updated the Traction control method, to prevent using it when the car is on neutral gear.

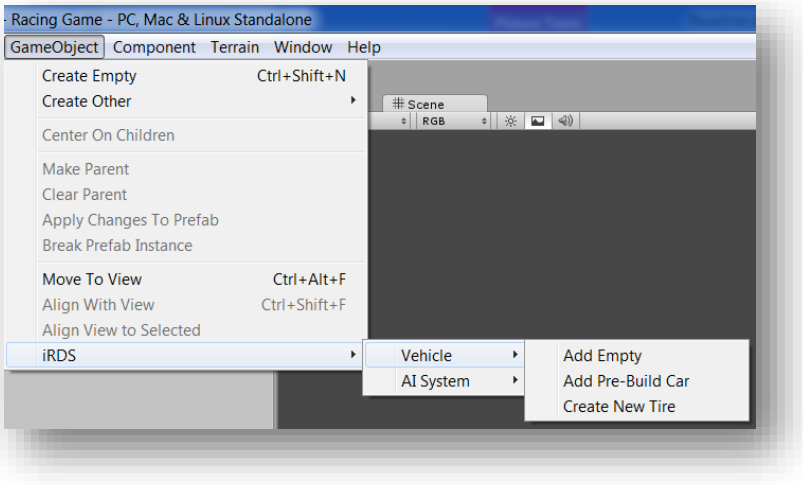
- Added TOE Angle for the wheels and fixed the steering for using front and rear steering vehicles
- Updated the Toe Angle Variable for all the tires.
- Fixed the over reacting issue on collisions.
- Fixed re-spawning to point the car to current waypoint
- Removed all Pits stop stuff if activate pits is not checked on the IRDSManager object.
- Added a feature to assign to the AI Drivers their own cars, so they race with the same car and same color each time they race.

Contact Information

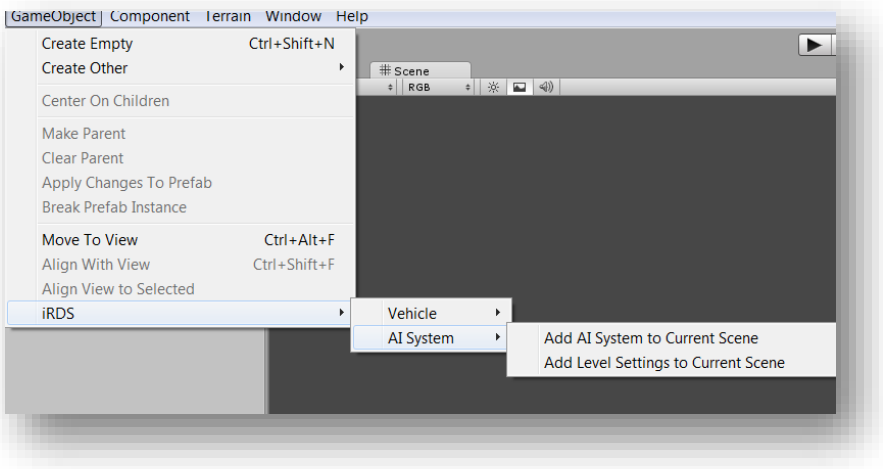
You can contact me by email on josegarrido@dagagames.com / rhodnius@hotmail.com / rhodnius39@gmail.com or by the unity forum user rhodnius with a PM.

IRDS (Unity Menu options)

There are three options available when you import IRDS package into your project:



The first menu option (Vehicle) has three sub menus, Add Empty – This option adds an empty car to the scene and prompts a configuration window to add the car body, tires, etc. (This would be covered on how to Rig a car section). The create new tire, this option would create a new prefab that you can then access on the Tires folder under the iRDS/Resources folder and set the values for that tire, also you can change the prefab name to represent the setting you made(i.e. Sport Tire, Racing Tire, etc.).



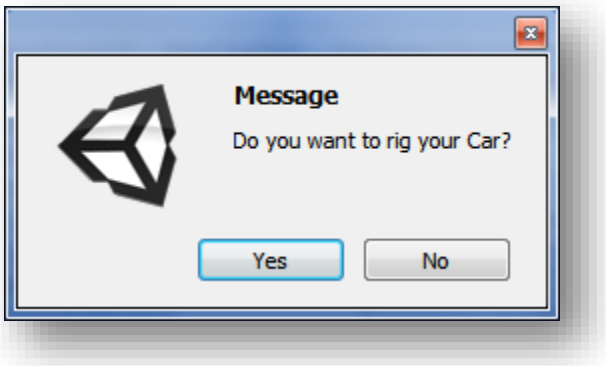
The second option (AI System); these options have two sub options, the first one (Add AI system to current scene) would add a Game Object with all the necessary Game Objects and scripts used by the AI System to work to the current Scene.

The second sub-option (Add Level Settings to Current Scene) would add a Game Object with all the required Level Settings used by the AI System to work on the current Scene, you also could choose not to add this GameObject to each scene or level, and just Add it once in the scene that would let the Player choose the track, car colors, opponents options, and his car and check the option "Destroy" on this GameObject, that would make this game object prevail when you load the track selected, that's why it won't need to be added to each level individually.

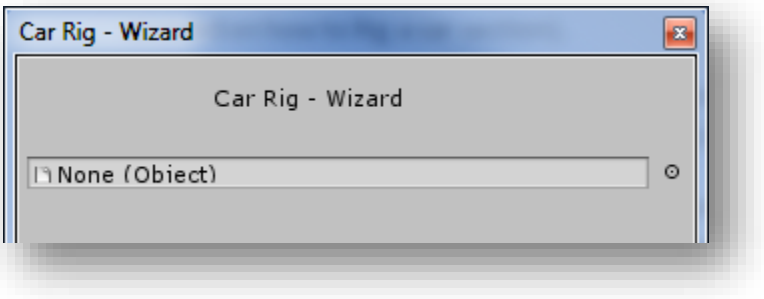
How to Rig a new Car

For rigging a new car you should use the Add Empty sub menu under the Vehicle menu option. This would display a window like the one on figure 1.

Note: When you press this option, the following Dialog box would appear, normally you should answer yes to this question, and a Rig car Wizard would be opened and would guide you through the Rig process. If you answer no, you need to do it all manually

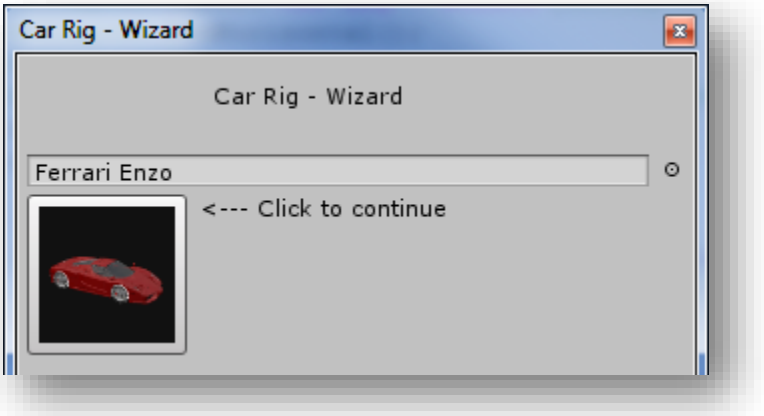


If you press “Yes” the following window would appear:



On this window you need to drag and drop the 3d Model you want to rig (car)

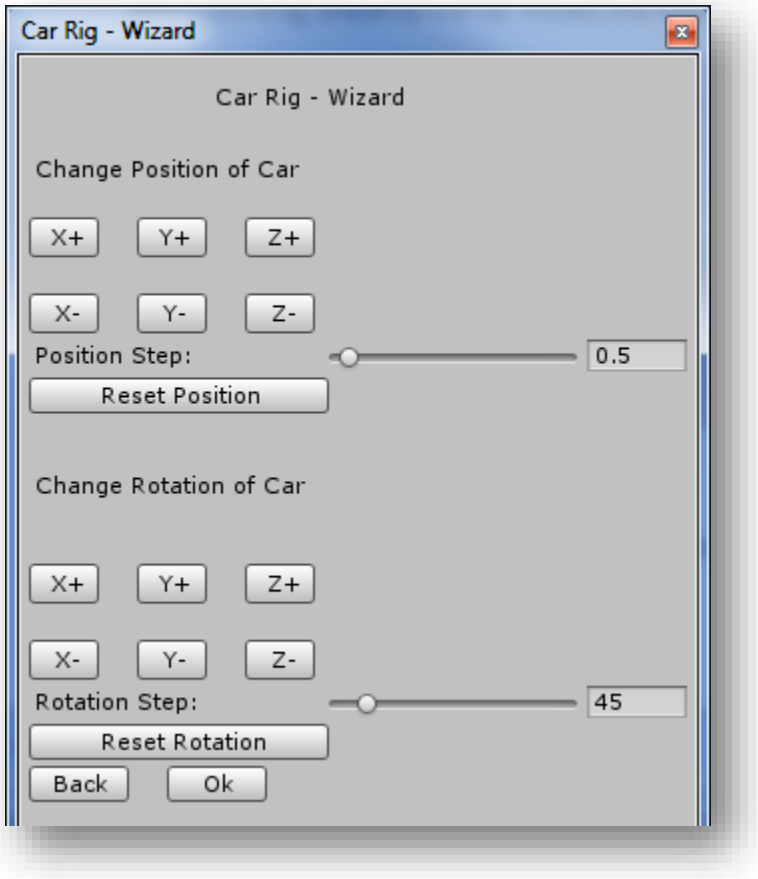
When you drag and drop the car model, the window would look like this:



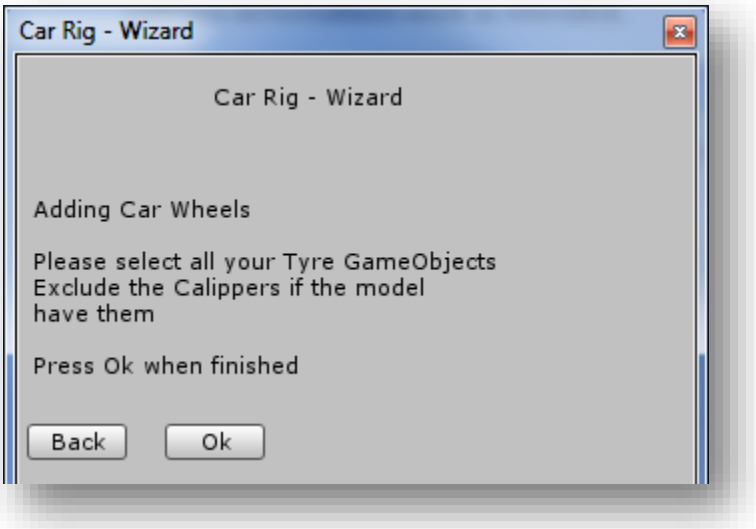
Now just click on the car preview button, if it is the model you want to rig, if it is not the model, Drag and drop the correct model to Rig.

The next step is to adjust the position and rotation of the model if is needed.

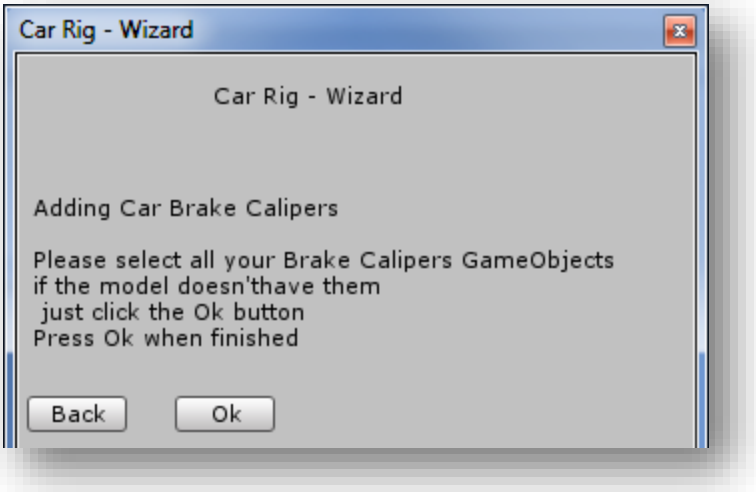
Note: Is better to do these adjustments on the 3D Modeler application (Blender, 3d Max, etc.) in order to set them correct before importing the 3d Model to Unity, this is to ensure that the collision deformations work as intended.



The next step is to assign the Tires, for this step you need to select all the game objects that conforms the tire and wheels and then press on the “OK” button like in the next picture:



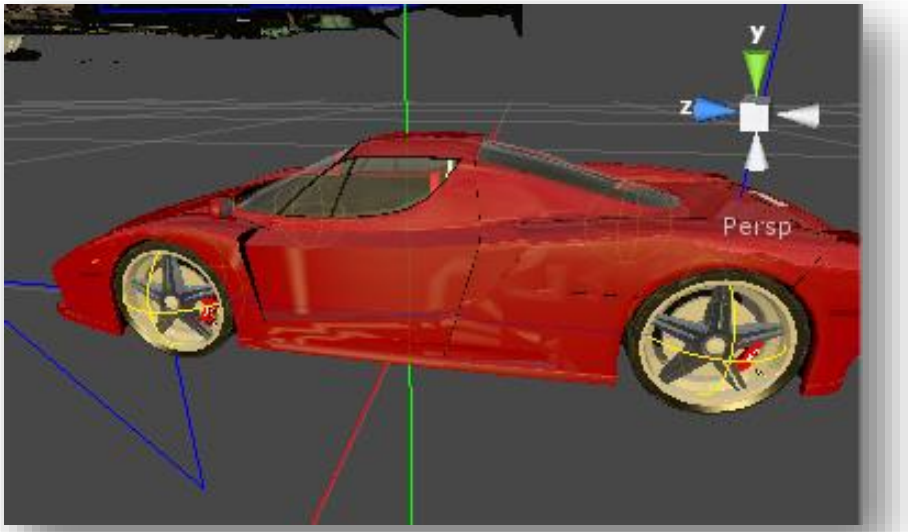
In the following window, you need to select the objects that represent the brake calipers, if the model doesn't have them, just click "Ok" Button.



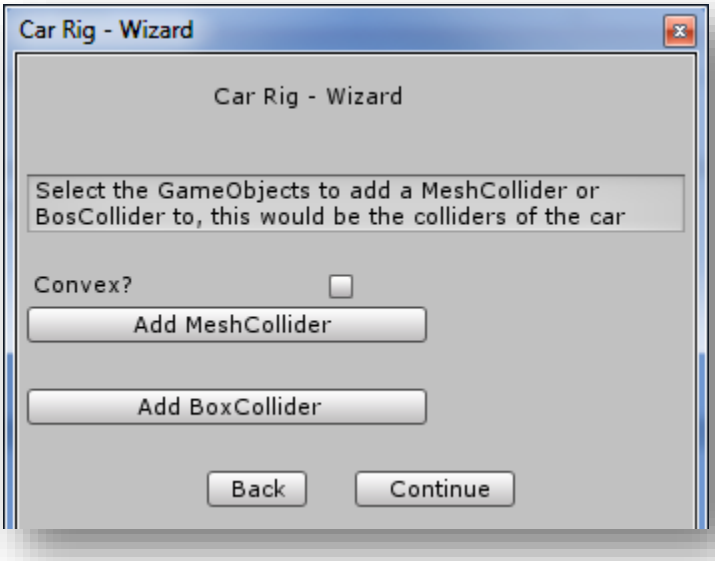
On the following window, you can set the wheel radius independently for the rear and front wheels, please consider that the Wizard already calculates the optimum wheel radius according to the wheel models selected previously.

Also you can select here the type of Drivetrain you want for the car.

Here is Picture of a car when you have selected the Tire Models, you can see that the Wizard marks the Tire radius in yellow spheres.

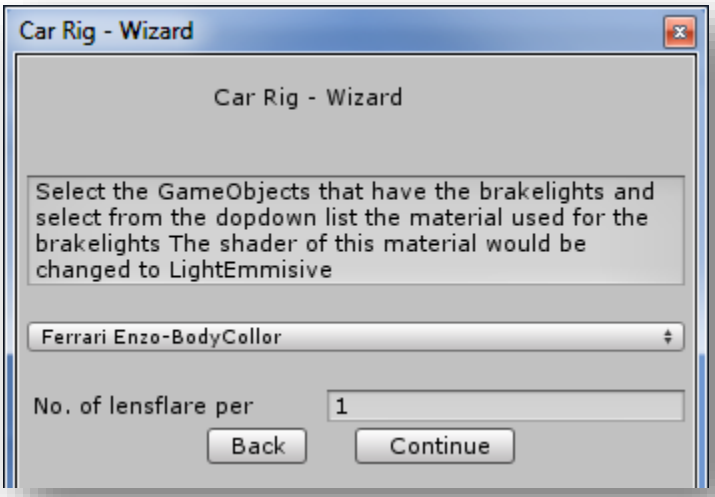


Adding the colliders for the model, to add colliders to the model, you can either add a Meshcollider or a BoxCollider, you just need to select all the objects of the car you want to add a collider to, and press on the desired button. Also you can select in the case of adding Meshcolliders, if you want it to be convex by selecting that option. When finished, press "Continue" Button.

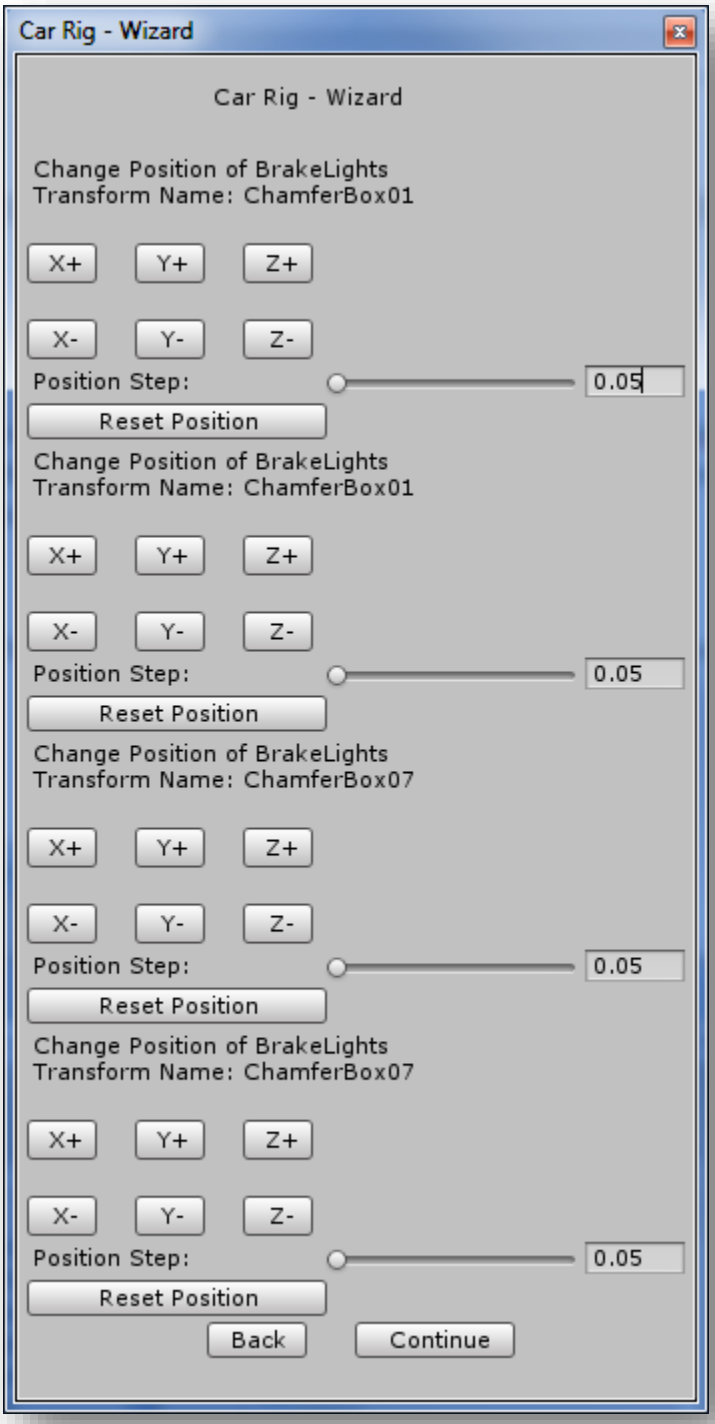


Adding Brake Lights and Flares, to add the brake lights and flares, just select all the gameobjects that represents the brake lights, then select the corresponding material from the dropdown list used for the brake lights in case the object has more than one material.

Then you can add the number of flares you want per object selected, and when finished press “Continue” button.

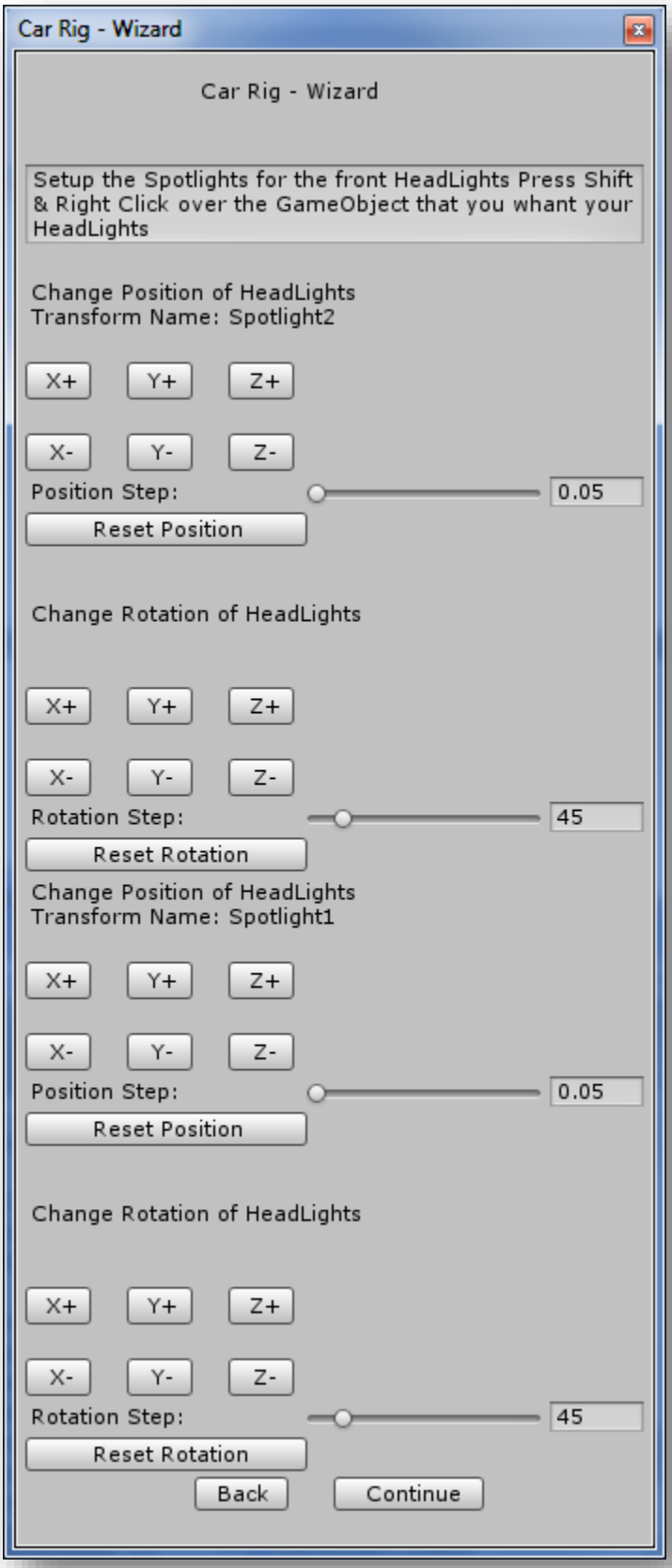


Changing the position and rotation of the brake flares, to change the position and rotation of the brake flares you should use the options presented on the following window, as you can see there is a “+” and “-“ button for each axis for position changing, and the same for rotation changing, use those buttons to change the position and rotation accordingly, each Flare have one position and rotation set as seen below.

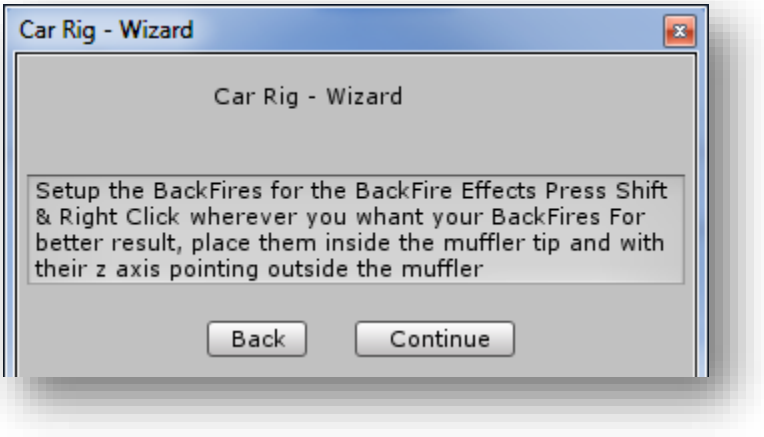

Car Rig - Wizard
Car Rig - Wizard
Change Position of BrakeLights
Transform Name: ChamferBox01
X+
Y+
Z+
X-
Y-
Z-
Position Step:
0.05
Reset Position
Change Position of BrakeLights
Transform Name: ChamferBox01
X+
Y+
Z+
X-
Y-
Z-
Position Step:
0.05
Reset Position
Change Position of BrakeLights
Transform Name: ChamferBox07
X+
Y+
Z+
X-
Y-
Z-
Position Step:
0.05
Reset Position
Change Position of BrakeLights
Transform Name: ChamferBox07
X+
Y+
Z+
X-
Y-
Z-
Position Step:
0.05
Reset Position
Back
Continue

Adding Spotlights to the car, to add spotlights to the car you just need to position on the front of the car and hold the Shift key and press the right mouse button where you want the spotlight to be placed.

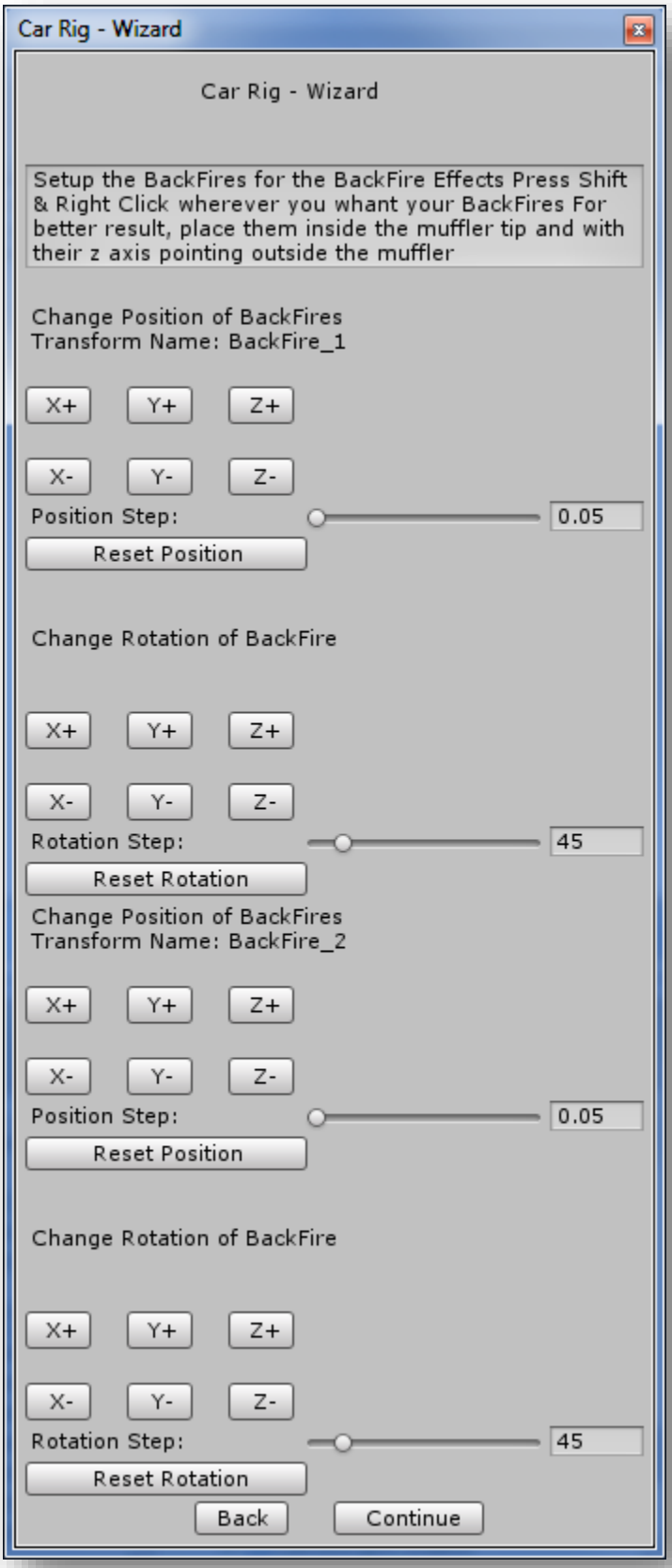
When you add a spotlight you would have some more options as you can see on the following window. The options are for adjusting the spotlight positions and rotation as you did with the brake lights.


Car Rig - Wizard
Car Rig - Wizard
Setup the Spotlights for the front HeadLights Press Shift & Right Click over the GameObject that you whant your HeadLights
Change Position of HeadLights
Transform Name: Spotlight2
X+
Y+
Z+
X-
Y-
Z-
Position Step:
0.05
Reset Position
Change Rotation of HeadLights
X+
Y+
Z+
X-
Y-
Z-
Rotation Step:
45
Reset Rotation
Change Position of HeadLights
Transform Name: Spotlight1
X+
Y+
Z+
X-
Y-
Z-
Position Step:
0.05
Reset Position
Change Rotation of HeadLights
X+
Y+
Z+
X-
Y-
Z-
Rotation Step:
45
Reset Rotation
Back
Continue

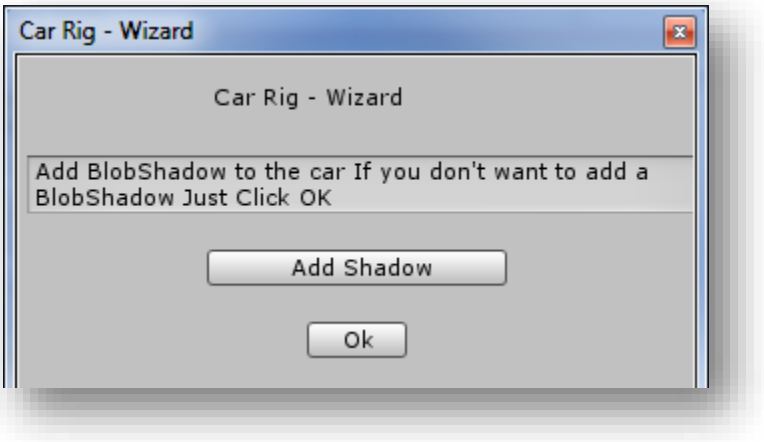
Adding backfires to the car, as you did with the spotlights, you need to hold shift key and press the right mouse button to add the backfire where you want it by pointing it with the mouse pointer.



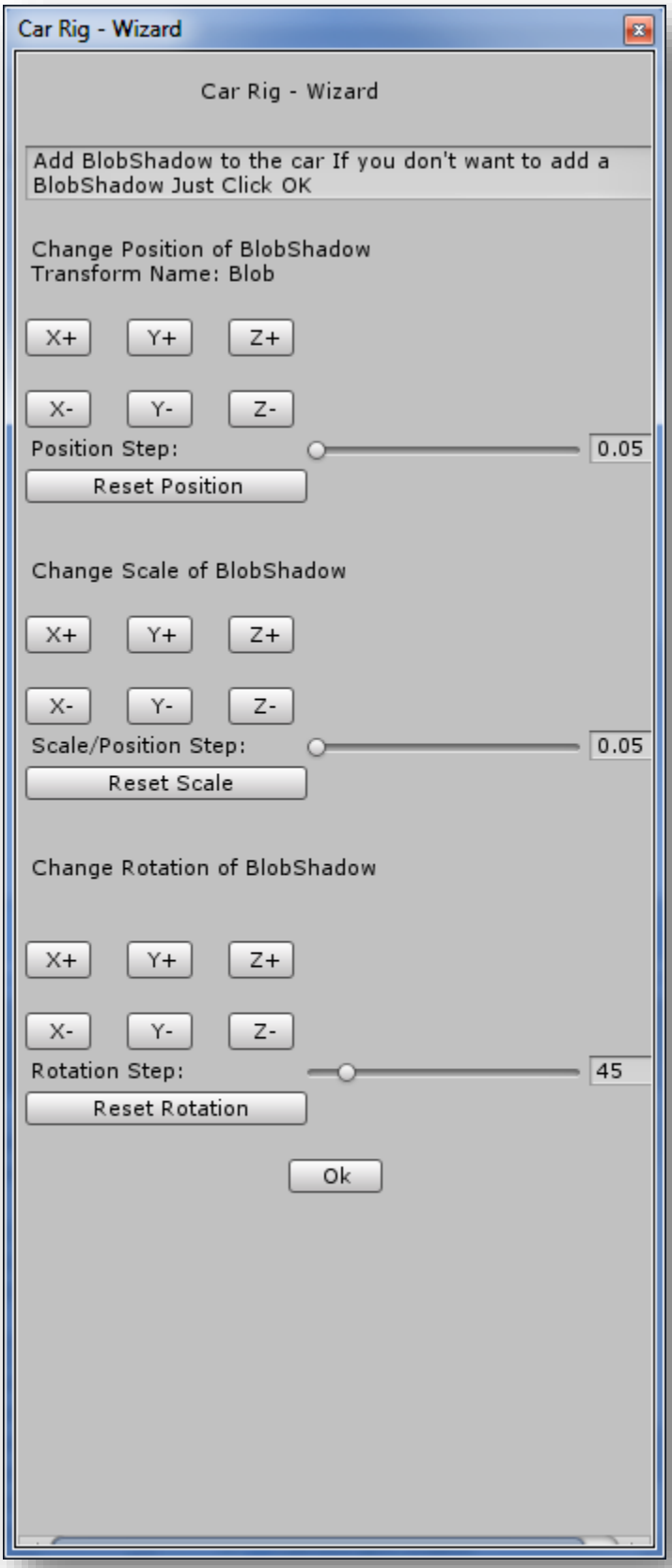
When you add a backfire, the following options would appear on the window, this options lets you adjust the position and rotation of the backfires.


Car Rig - Wizard
Car Rig - Wizard
Setup the BackFires for the BackFire Effects Press Shift & Right Click wherever you whant your BackFires For better result, place them inside the muffler tip and with their z axis pointing outside the muffler
Change Position of BackFires
Transform Name: BackFire_1
X+
Y+
Z+
X-
Y-
Z-
Position Step:
0.05
Reset Position
Change Rotation of BackFire
X+
Y+
Z+
X-
Y-
Z-
Rotation Step:
45
Reset Rotation
Change Position of BackFires
Transform Name: BackFire_2
X+
Y+
Z+
X-
Y-
Z-
Position Step:
0.05
Reset Position
Change Rotation of BackFire
X+
Y+
Z+
X-
Y-
Z-
Rotation Step:
45
Reset Rotation
Back
Continue

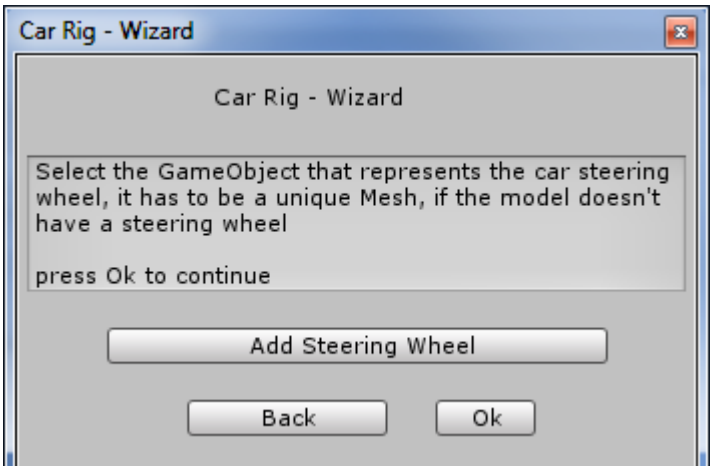
Add blob shadow to the car, just click on the “Add Shadow” button to add a blob shadow to the model.



If you add the blob shadow to the model, the following options would appear on the window; these options let you adjust de position, rotation and size/scale of the shadow.

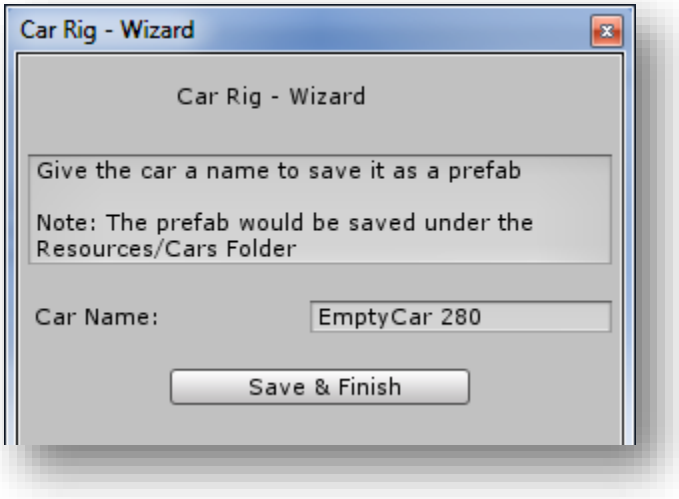

Car Rig - Wizard
Car Rig - Wizard
Add BlobShadow to the car If you don't want to add a BlobShadow Just Click OK
Change Position of BlobShadow
Transform Name: Blob
X+
Y+
Z+
X-
Y-
Z-
Position Step:
0.05
Reset Position
Change Scale of BlobShadow
X+
Y+
Z+
X-
Y-
Z-
Scale/Position Step:
0.05
Reset Scale
Change Rotation of BlobShadow
X+
Y+
Z+
X-
Y-
Z-
Rotation Step:
45
Reset Rotation
Ok

Add steering wheel, at this point you need to select the game object that represents the model of the steering wheel of the car, please note that this object should be independent of any other object, as it would be rotated depending on the steering inputs. When you select the object, click on “Add steering Wheel” button.

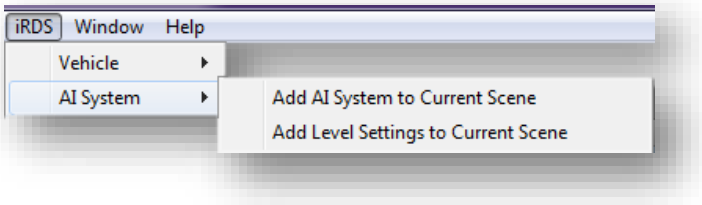


If you added the steering wheel, the following options would appear on the window; these options let you adjust the position and rotation of the steering wheel container in order for it to match the centered axis where the model rotates if you turn the steering wheel left or right.

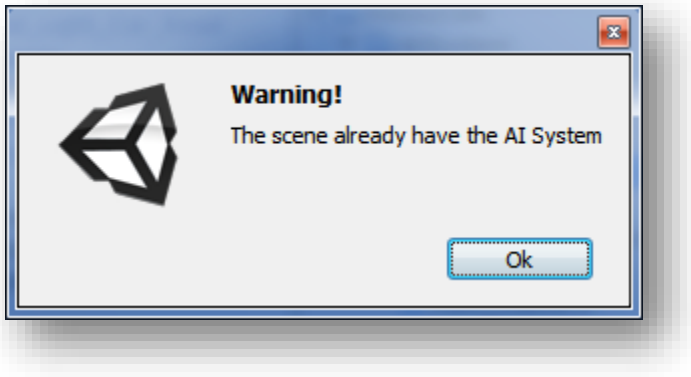
The last step, is just to click on the “Save & Finish” button, this would add a prefab of the model on the Resources/Cars folder on the project.



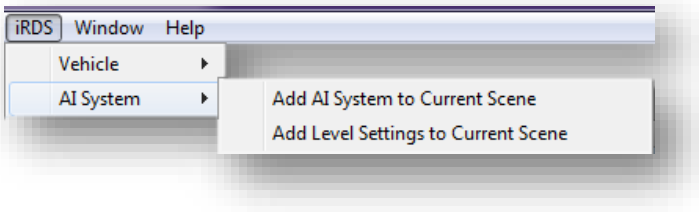
AI System – Add AI System to scene



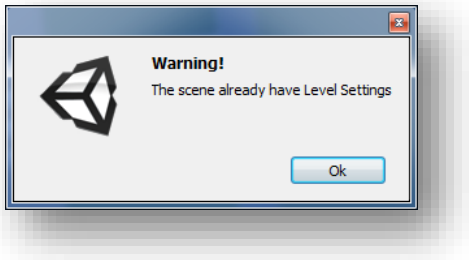
With this option you can add (only once) the GameObject called IRDSManager to the current scene, this is the core object of the system and it is required to be on each scene you want to use the AI system. If you try to add more than once this object, the system would send you this warning message:



AI System – Add Level Settings to scene



With this option you can add (only once) the GameObject called LevelLoad to the current scene, this is the object that have all the Level Settings of the system and it is required to be on each scene you want to use the AI system. If you try to add more than once this object, the system would send you this warning message:



Also you could choose to not add this object to each Level of your game, and instead add it once on a scene used for the Player to choose his options, like for example, the number of opponents, color of his car, his car, which track to race, etc. and you just need to check the option “Destroy” on the Level Settings Object, so when you load the selected track (scene) the player choose, it won’t be destroyed and would be available on the track scene with the settings previously selected by the user. So you could manipulate all this settings with your own script and also load the track selected which could be stored on one of the variables of this object (trackToRace), see section “Level Settings”.

AI System Settings and Configuration

The AI System has some settings and configurations that need to be done before you can use it, the following step by step procedure would guide you through this process.

The first step is to create the track limits line, so you need to select the IRDManagaer Game Object from the Hierarchy list and the following options would appear on the Inspector:

Add new points or markers

Insert new points between two existing markers

Create the track limits

Create the Grid Positions

Create the Grid Positions for rolling starts

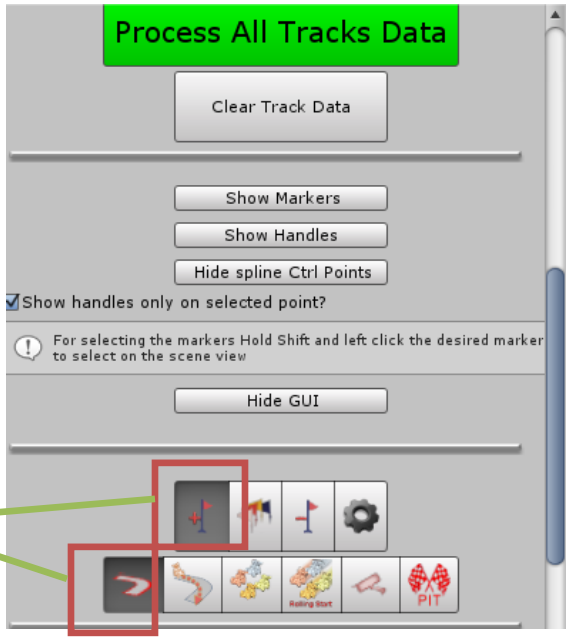
General Settings and HUD Configuration and options

Add the road camera points

Add/insert the waypoints for the pits

Add/insert the waypoints for the racing line

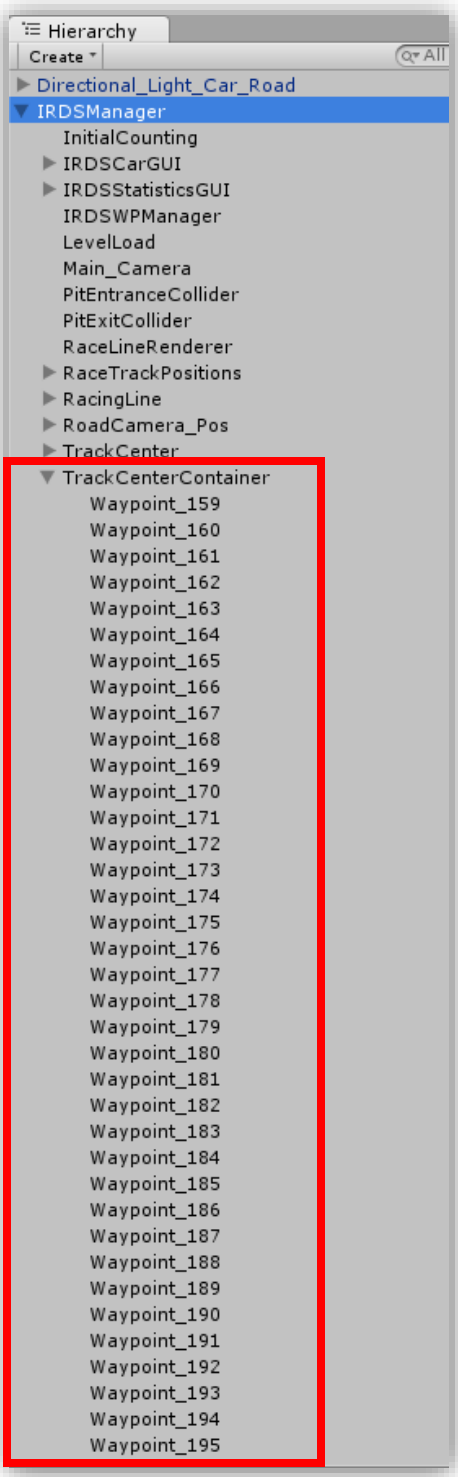
Adding the Track Limits

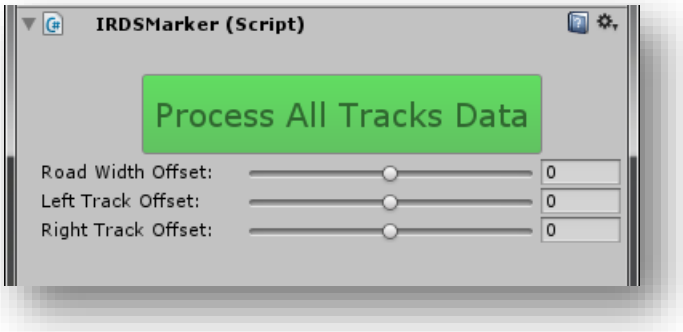


Selected options are Grey

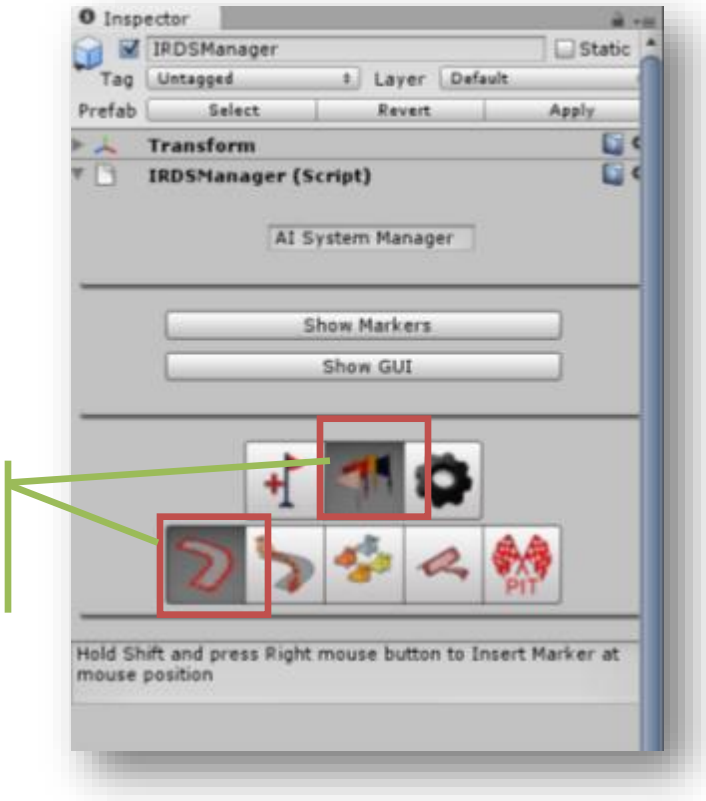
To add the Track Limit, you just need to hold shift and right click on the center of the track where you want the start of the track to be, and then start adding more points advancing through the track until you get again to the start of the track.

You can also select the already added markers to adjust their position on the track and their width.


Hierarchy
Create
All
Directional_Light_Car_Road
IRDSManager
InitialCounting
IRDSCarGUI
IRDSStatisticsGUI
IRDSWPManager
LevelLoad
Main_Camera
PitEntranceCollider
PitExitCollider
RaceLineRenderer
RaceTrackPositions
RacingLine
RoadCamera_Pos
TrackCenter
TrackCenterContainer
Waypoint_159
Waypoint_160
Waypoint_161
Waypoint_162
Waypoint_163
Waypoint_164
Waypoint_165
Waypoint_166
Waypoint_167
Waypoint_168
Waypoint_169
Waypoint_170
Waypoint_171
Waypoint_172
Waypoint_173
Waypoint_174
Waypoint_175
Waypoint_176
Waypoint_177
Waypoint_178
Waypoint_179
Waypoint_180
Waypoint_181
Waypoint_182
Waypoint_183
Waypoint_184
Waypoint_185
Waypoint_186
Waypoint_187
Waypoint_188
Waypoint_189
Waypoint_190
Waypoint_191
Waypoint_192
Waypoint_193
Waypoint_194
Waypoint_195



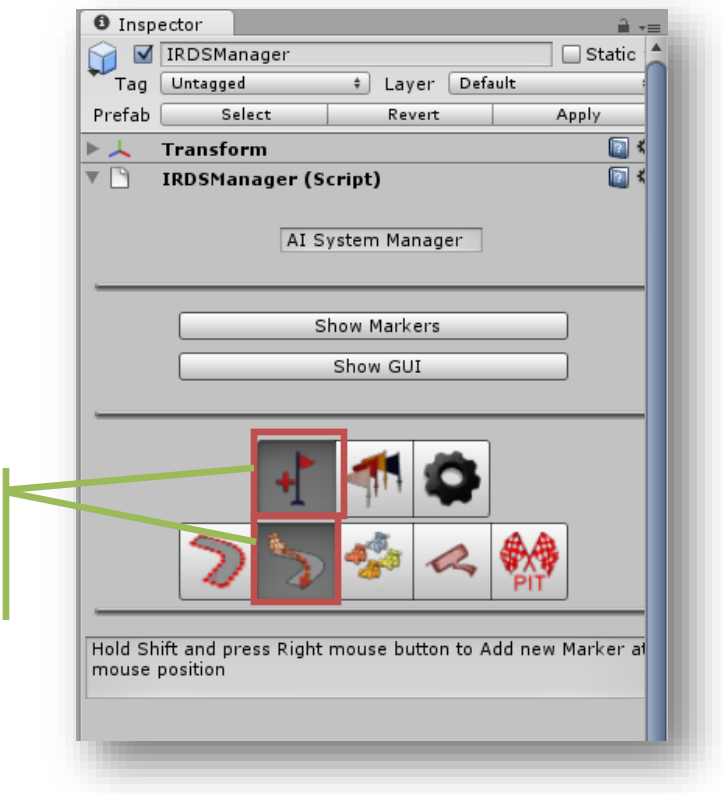
If you want to insert markers between an existing markers, you need to select the option with this icon  while you are on the Track Limits settings (make sure the icon  is selected too) and hold shift and right click where you want to insert the new marker between the existing ones. Here is a screen shot of the options when you want to insert a marker:



Selected options are Grey

As you can see on the picture above, the insert marker option is selected and the Track Limits settings is selected too.

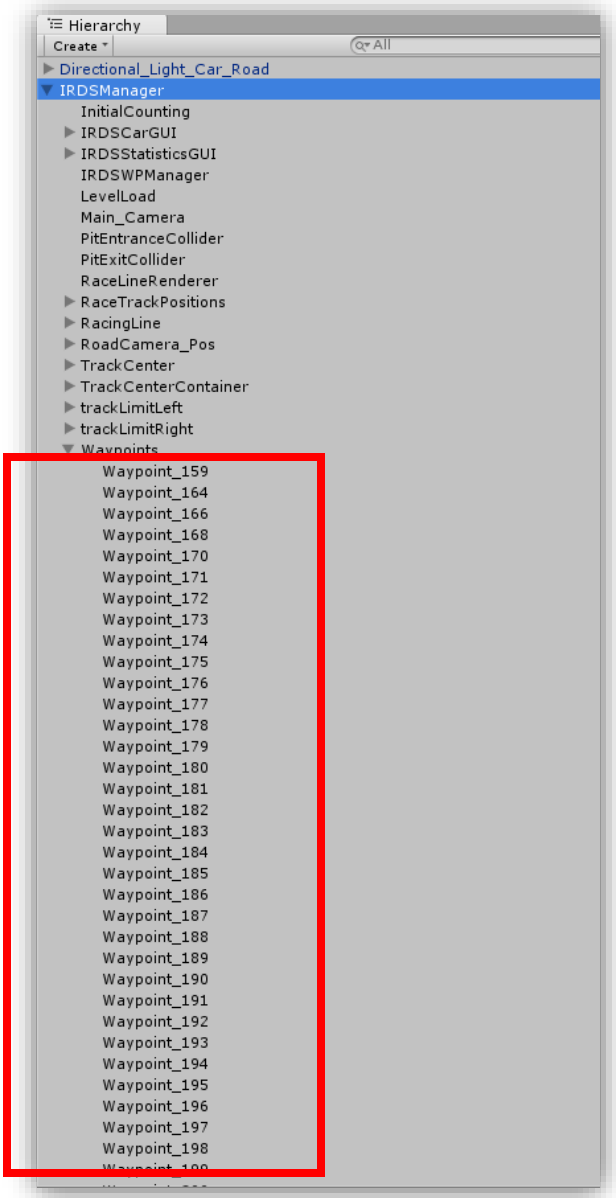
Adding the Race Line



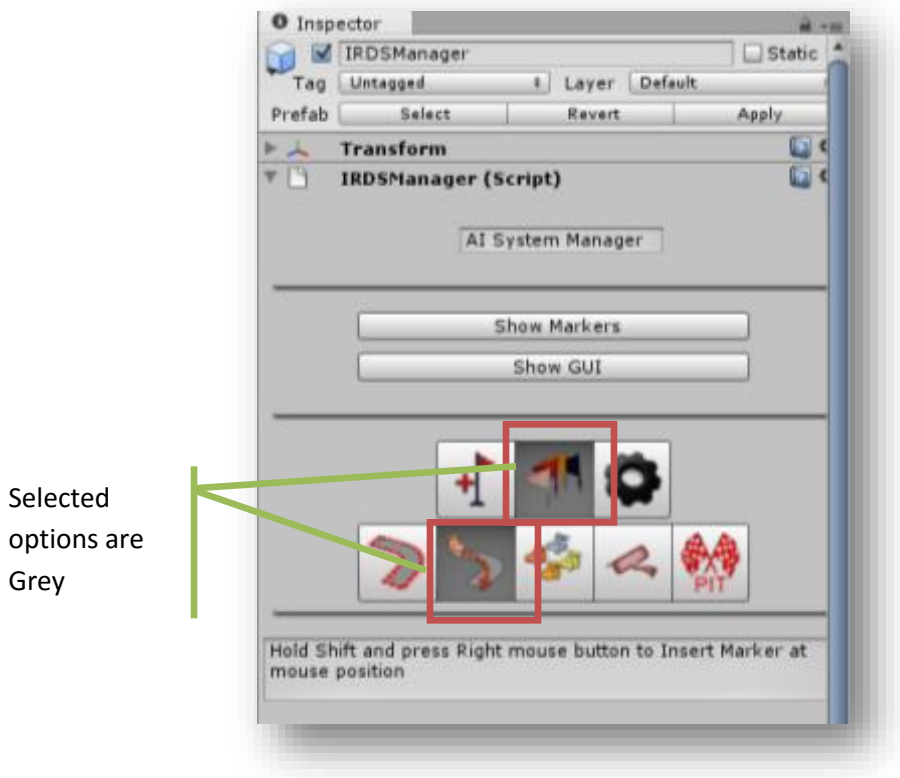
Selected options are Grey

To add the Race Line, you just need to hold shift and right click on the center of the track (or put it more to one of the sides in case of a curve) where you want the start of the track to be, and then start adding more points advancing through the track until you get again to the start of the track.

You can also select the already added markers to adjust their position on the track.

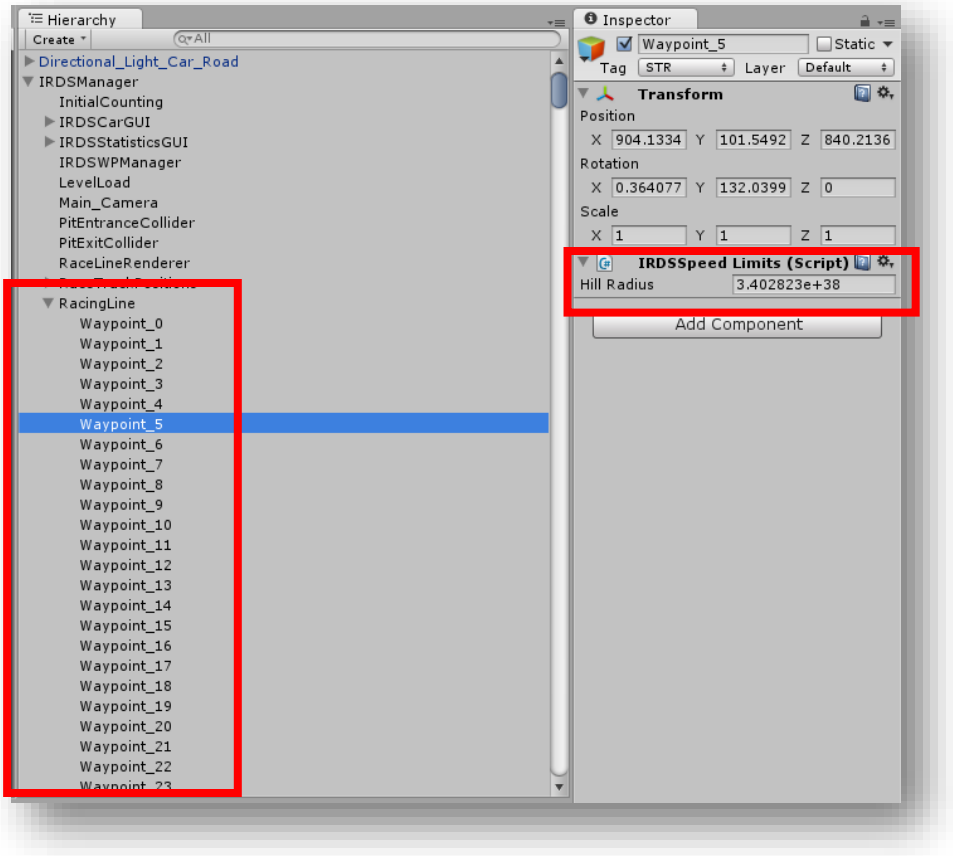


If you want to insert markers between an existing markers, you need to select the option with this icon  while you are on the Race Line settings (make sure the icon  is selected too) and hold shift and right click where you want to insert the new marker between the existing ones. Here is a screen shot of the options when you want to insert a marker:

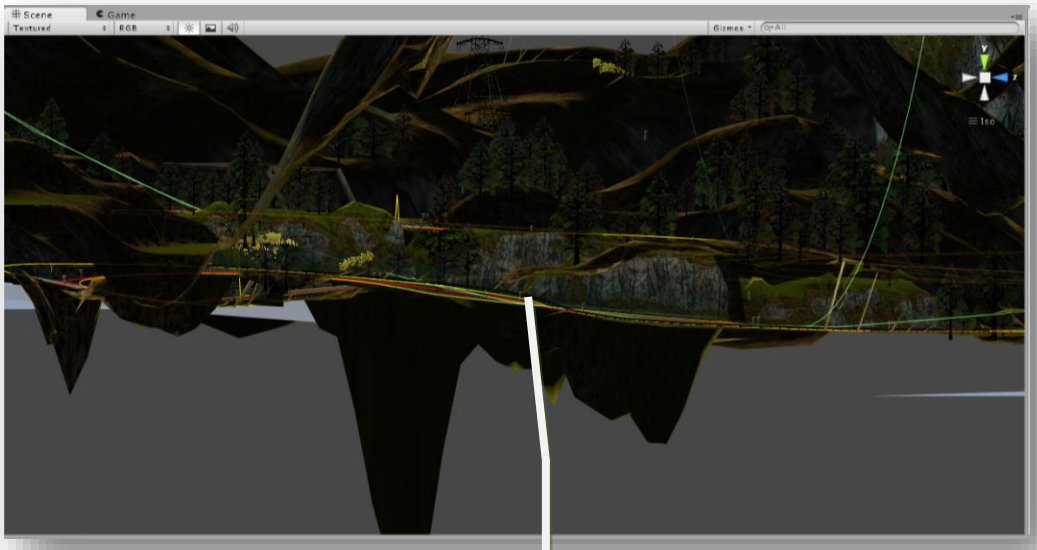


As you can see on the picture above, the insert marker option is selected and the Race Line settings is selected too.

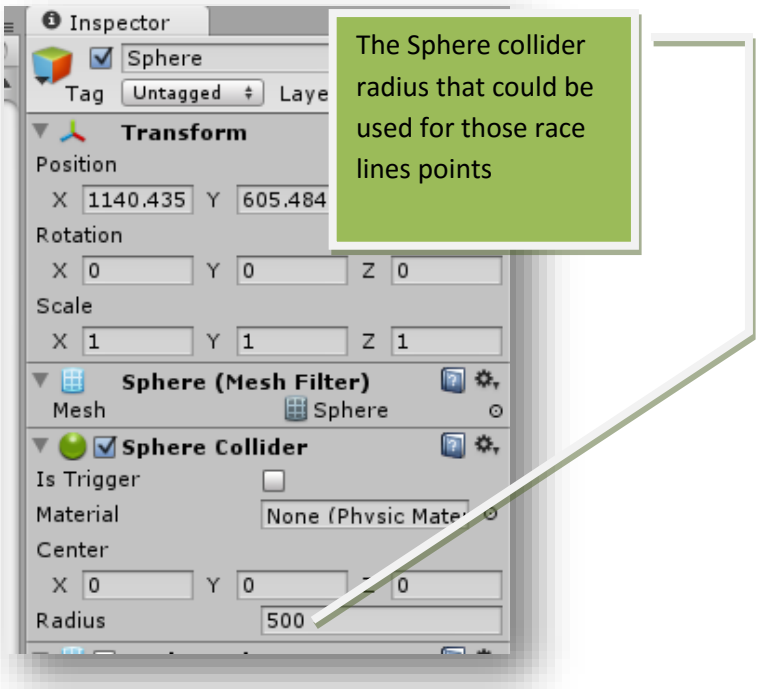
Also, when you finish adding all the waypoints, you can select, individually or multiple, waypoints of the race line itself, this is to set (if required) the hill radius that would affect the maximum speed the AI's would take on that part of the track, this is because right now the AI System can't automatically detect bumps or hills on the track, and they could lose control because they won't slow down to take the hill, in those cases you can select several waypoints on the hill and set the radius of the hill, see picture below to illustrate:



And so, why is it that we used radius instead of putting a maximum speed directly?, well this system calculates the maximum speed for different types of vehicles and tires, so for the same radius, different tires and cars can achieve different max speeds, so that's why we use hill radius and not maximum speed, if you want to know more precisely the radius of a hill, you could use a sphere game object, and disable the renderer, so you could use the radius of the sphere collider, see the following picture:

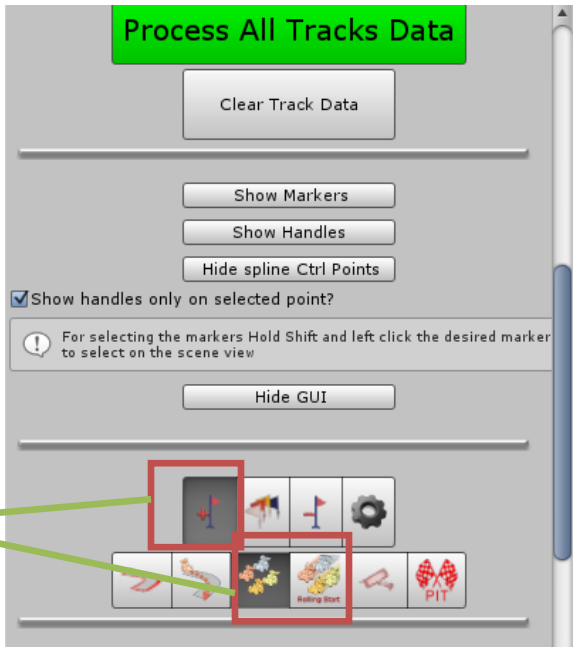


The Sphere collider matches the road hill shape



The Sphere collider radius that could be used for those race lines points

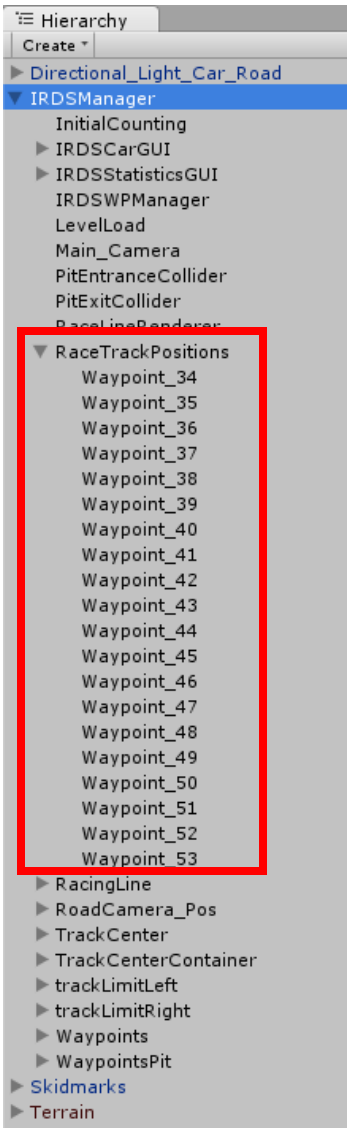
Adding the grid standing and rolling starts



Selected options are Grey

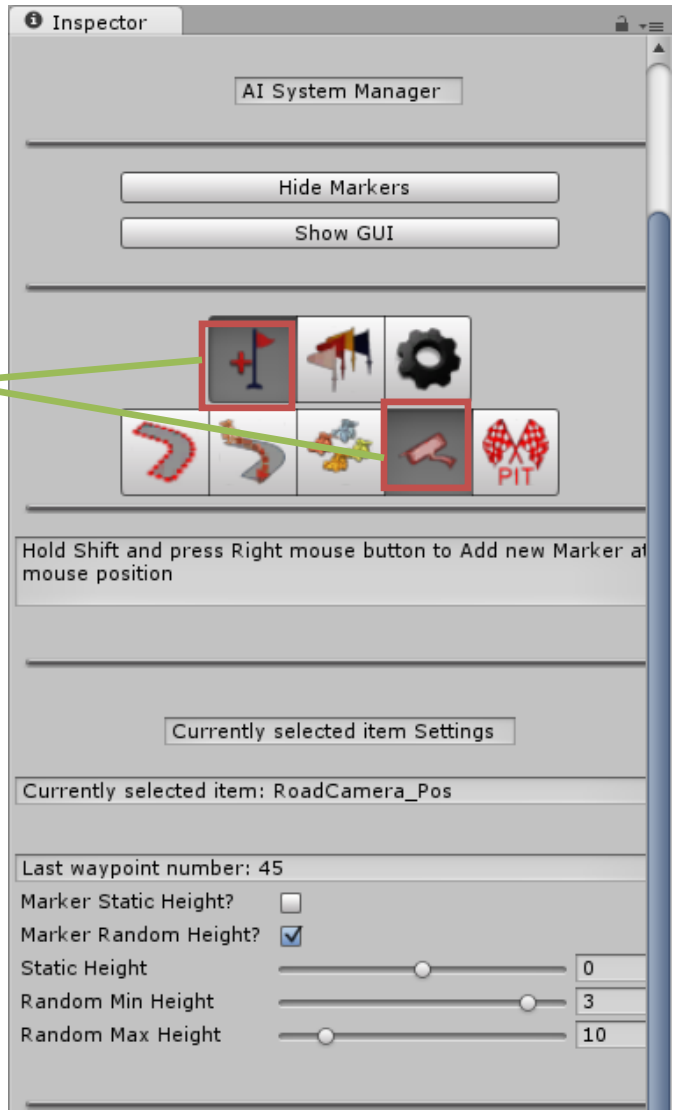
To add the Grid Positions, you just need to hold shift and right click on the spot where you want the cars to start the race, please consider that this should be done behind the start line (waypoint # 0 of the Race Line).

You can also select the already added markers to adjust their position on the track.



You can't insert markers between existing markers as it is done with the Race Line; this is because it is not needed.

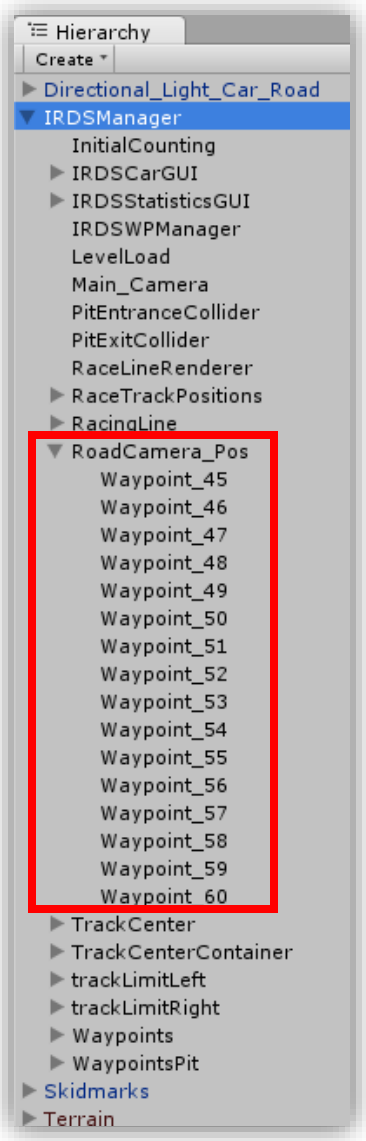
Adding the road cameras



Selected options are Grey

To add the road cameras, you just need to hold shift and right click on the spot where you want the cameras to be.

You can also select the already added markers to adjust their position on the track.



You can't insert markers between existing markers as it is done with the Race Line; this is because it is not needed.

The parameters shown on the above picture are used as follows:

Marker Static Height? : Toggle this feature to set the height of the next added marker as static, using the Static Height value.

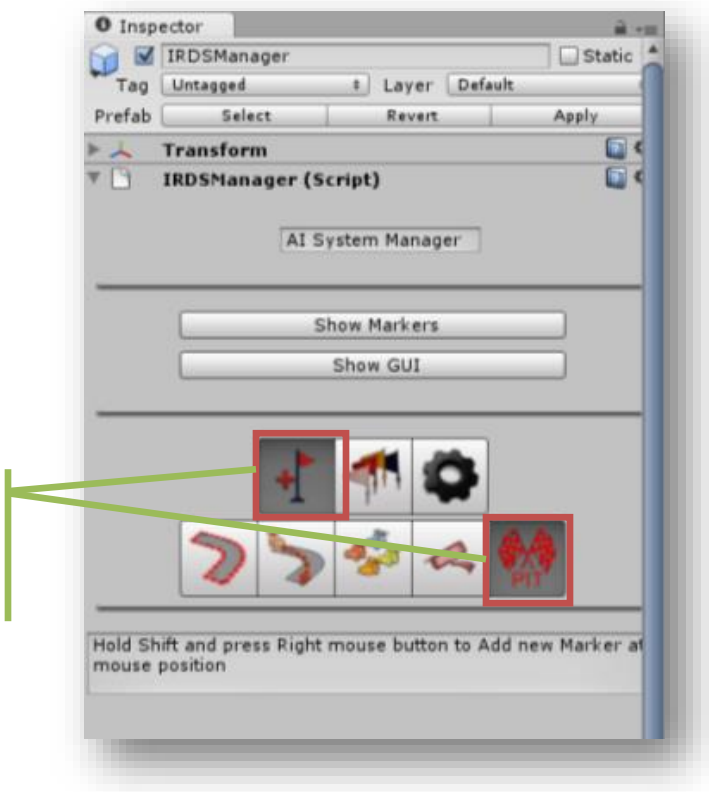
Marker Random Height? : Toggle this feature to set the height of the next added marker as Random, using values from Random min Height to Random Max height.

Static Height: This value represent the static height of the next added marker.

Random Min Height: This value is the min random height.

Random Max Height: This value is the max random height.

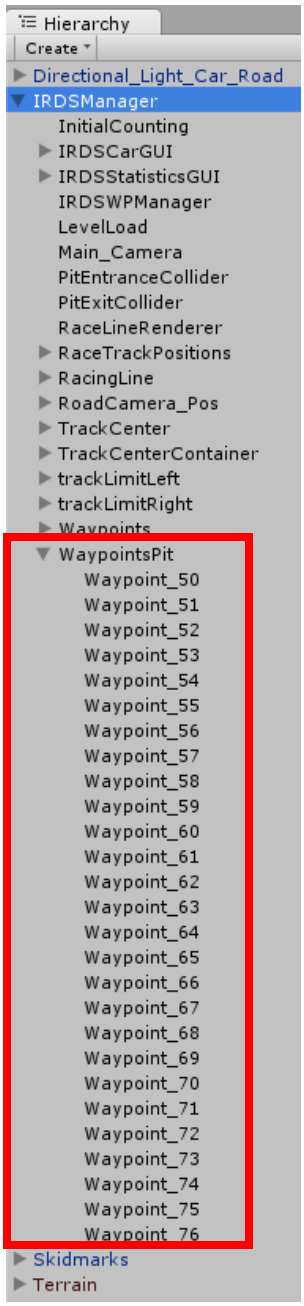
Adding the Pits Line



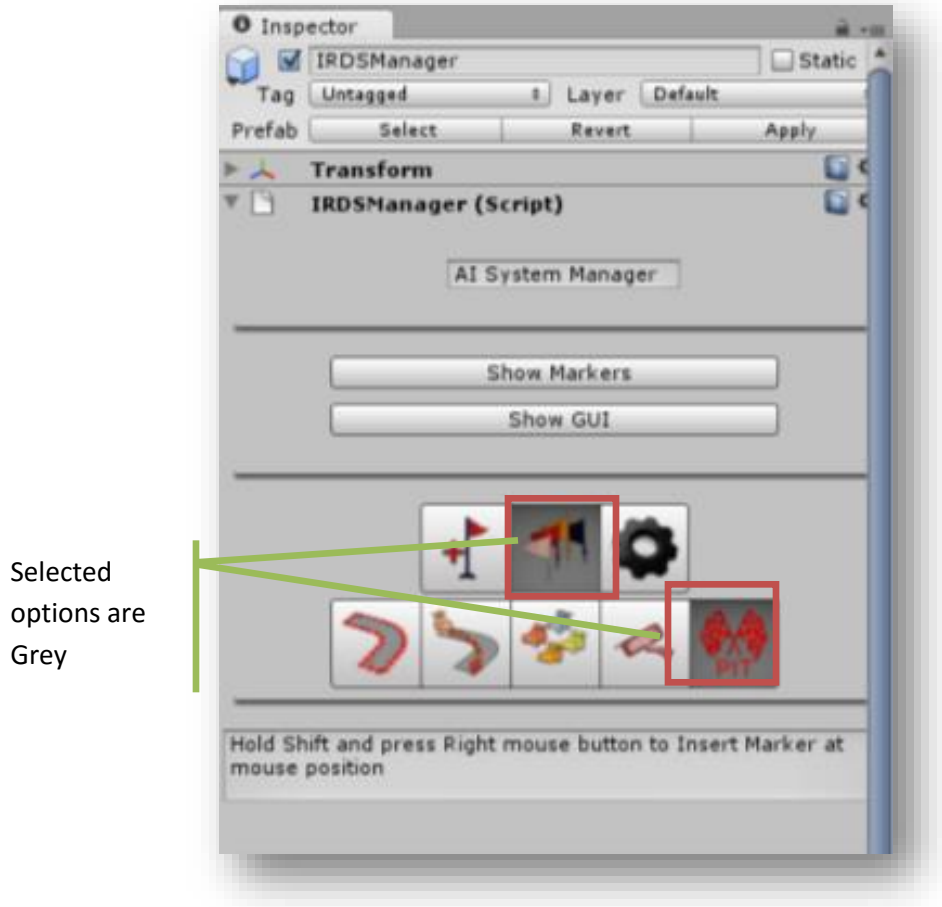
Selected options are Grey

To add the Pit Line, you just need to hold shift and right click on the part of the track where the pit line would start and then start adding more points advancing through the pit line until you get again to the track when the pit line ends.

You can also select the already added markers to adjust their position on the pit line.



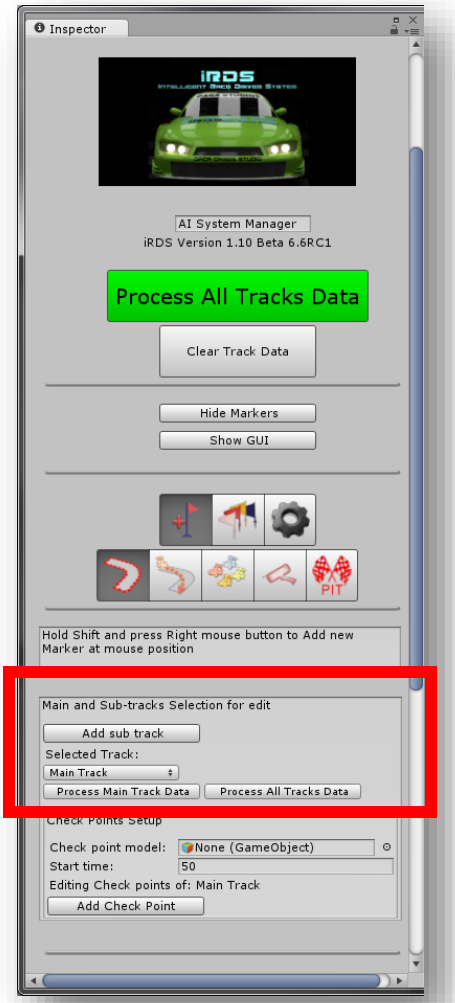
If you want to insert markers between an existing markers, you need to select the option with this icon  while you are on the Race Line settings (make sure the icon  is selected too) and hold shift and right click where you want to insert the new marker between the existing ones. Here is a screen shot of the options when you want to insert a marker:



As you can see on the picture above, the insert marker option is selected and the Pit Line settings is selected too.

IRDS Sub Track Settings

On this section we would cover how to add new subtracks to our existing main track, let's see first where are the options for adding subtracks located on the IRDSManager, for this we just select the IRDSManager object as in the picture below:



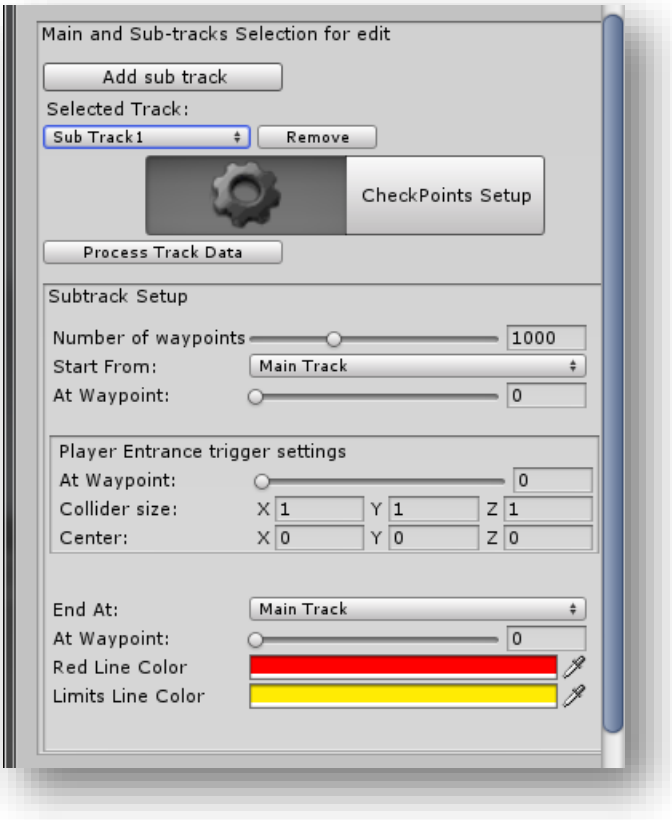
You would be able to see at the middle of the inspector a new section called "Main and Sub-Tracks Selection for edit" this would be the options that are available for adding, editing and removing sub tracks on your scenes.

Let's take a closer look now to the available options:

- **Add Sub Track:** By pressing this button, you would add a new sub track to your scene.
- **Selected Track:** Here you can select each of the tracks you have on your scene, for adjusting their settings, by default the main track is selected, if you have added a new sub track it would be on this drop down list.

- **Process Main Track Data:** By pressing this button, the system will process all the track data of the main track only.
- **Process All Track Data:** By pressing this button, the system will process all the track data of the main trackand all created sub tracks.

This options would change if you select a sub track from the drop down box list, as the following picture:



You can notice that now there are more options available than when the main track is selected, we are going to explain each of this options.

- **Number of waypoints:** The amount of waypoints this Sub Track would have.
- **Start From:** Select here the track that this sub track would start from, in this example we selected the main track.
- **At Waypoint:** You most select here the waypoint from the selected “Start From” track, this would be the starting point for the new sub track.
- **Player Entrance Trigger Settings:** This section is for setting up the trigger that would be used by the system to know when a player have entered into this sub track, the options in this section are the followings:
    - **At Waypoint:** This is the waypoint from this subtrack that the trigger would be placed on, normally you need to

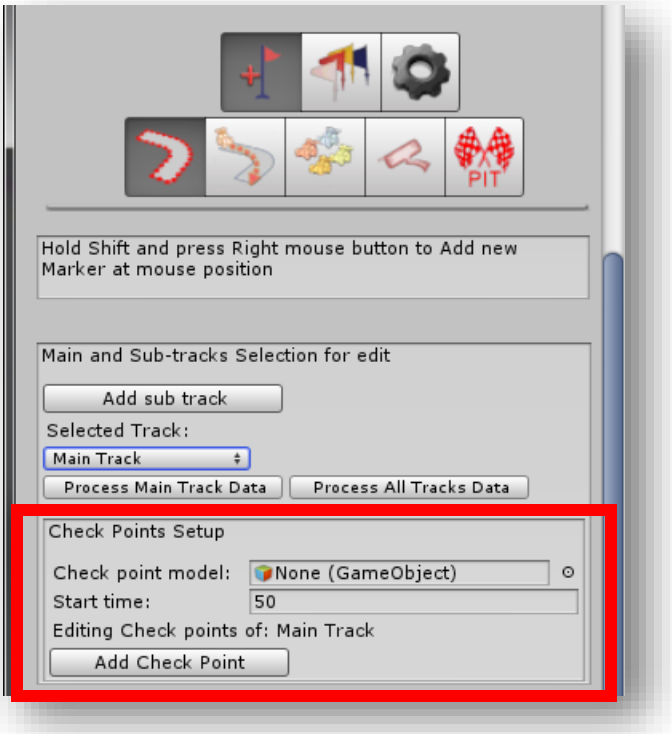
    - **Collider Size:** This is the size of the collider, it should match the size of this subtrtack width and height, normally the depth is set to just 1 unit.
    - **Center:** The center of this trigger relative to the waypoint it is at. Use it to adjust the trigger position accordinghly.
- **End At:** Select here the Main track or sub track you want this sub track to end at.
- **At waypoint:** The waypoint from the selected "End At" track this sub track would have its ending point at.
- **Red Line Color:** The color of the "racing line" of this sub track, this is useful to distinguish this racing line from the other tracks.
- **Limits Line Color:** The same as the Red Line Color, but for the limits lines.

IRDS Check Points Settings

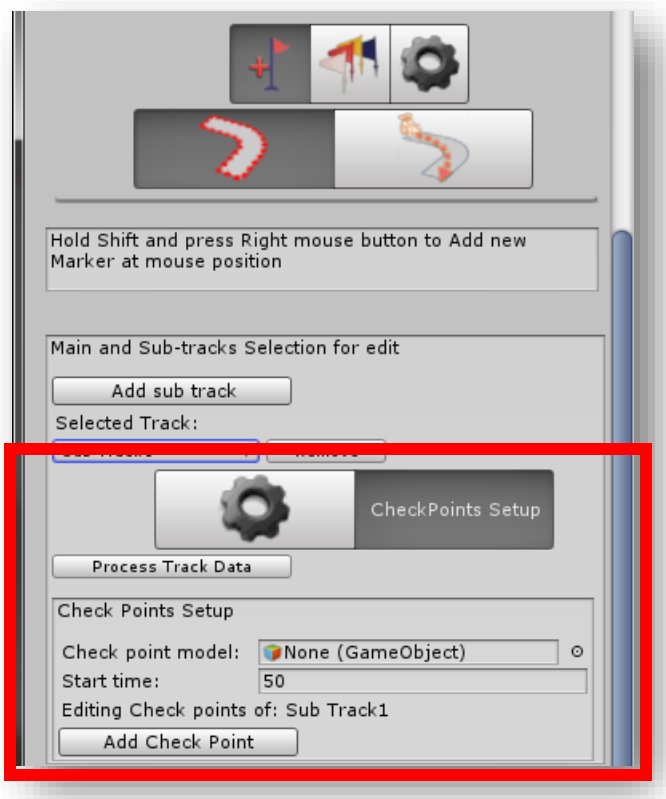
On this section we are going to learn how to setup the check points, for check point based races.

First we are going to take a look at where this options are on the IRDSManager, they are in 2 different places, depending on if you have the main track selected, or a sub track.

For the main track, the check points options are as seen on the following picture:



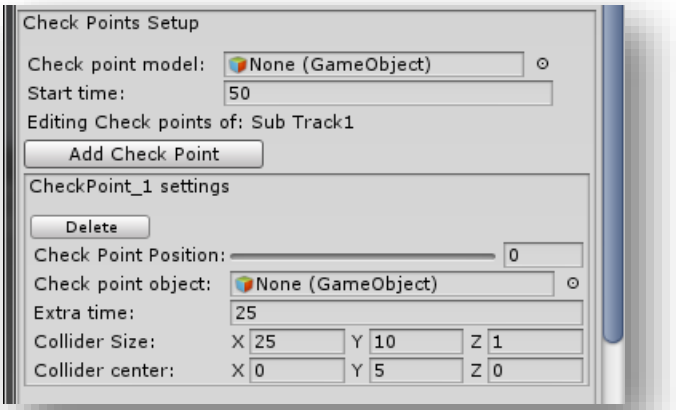
And for the Sub tracks are as in the following picture:



The options in both cases are the same, so we are going to review them just one time, the options are as follows:

- **Check Point model:** you can drop here the renderer that would be considered the check point if you want, if not you can leave this empty.
- **Start Time:** The amount of time every racer would have to get to the first check point.
- **Editing Check Points of:** This label indicates which track is currently edited.
- **Add Check Point:** Use this button to add a new check point to the selected track.

When we add a new check point, there are some option available on each check point, see the picture bellow:



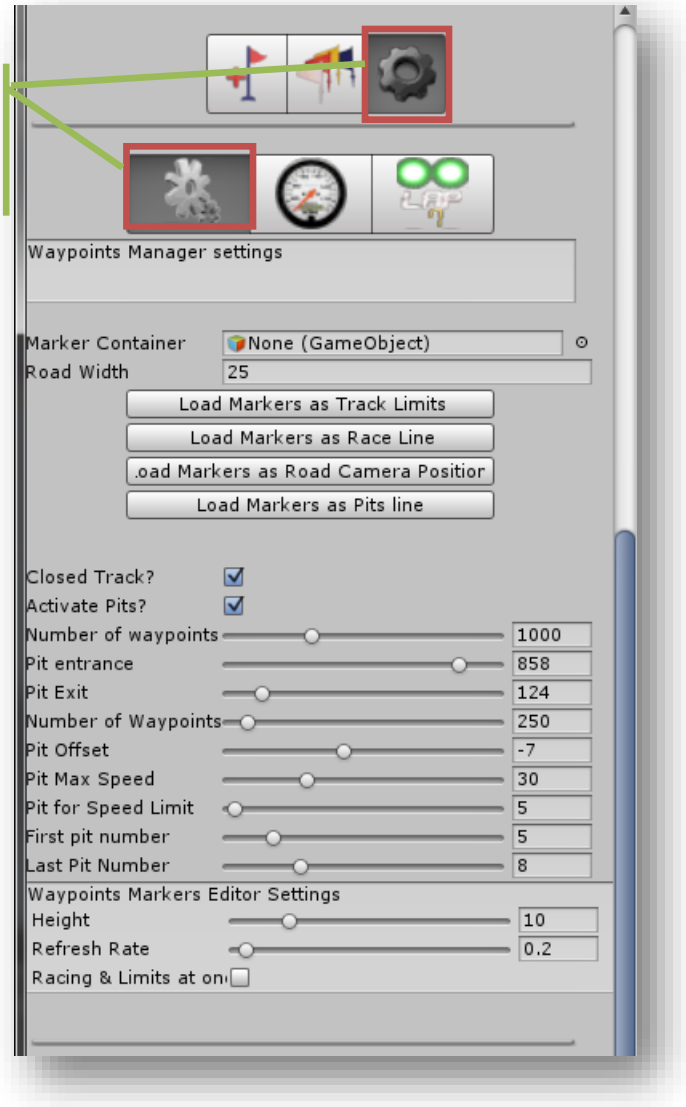
These options are the same for every checkpoint you may add, and each of these options are going to be explained as follows:

- **Delete:** This button would delete this checkpoint.
- **Check Point Position:** This option sets at which Waypoint from the currently edited track this check point would be at.
- **Check Point Object:** The Game Object (or prefab) that would represent graphically this check point.
- **Extra Time:** The time that would be added to each player/AI if they cross this checkpoint.
- **Collider Size:** The size of the collider for this checkpoint, it should match the width of the track, by default it have the track width on the X axis.
- **Collider Center:** The position relative to the waypoint this check point is at, by default it is centered at Y = collider size Y devided by 2.

IRDS Manager General Settings

On this section, we are going to review the general settings of th IRDS AI System, in the picture below you can see all the general parameters that can be configured

Selected options are Grey



Marker Container: Put here the game object that contains all the markers of EasyRoads, or also all the gameobject that represents the markers, this is needed only if you are using EasyRoads for making the road or if you already have al the gameobject you may want to use as the waypoints and just want the system to load them.

Road Width: Width of the road, this should the same value of the easy road width if easy road is used, if not, use the desired value here.

Load Markers as Track Limits: Press this button to process the Markers and load them as the track limits

Load Markers as Race Line: Press this button to process the Markers and load them as the Racing line.

Load Markers as Road Camera Positions: Press this button to process the Markers as the points for the road camera view mode.

Load Markers as Pits Line: Press this button to process the Markers as the pits stop line.

Closed Track?: Is this a circuit or sprint race?

Activate Pits?: Is this track with a pit line? Do you want the pits to be active for this track?.

Number of waypoints: Number of total points that would conform the racing line.

Pit entrance: Number of racing line point nearest to the first pit point.

Pit Exit: Number of racing line point nearest to the last pit point.

Number of Waypoints pits: Number of total points that would conform the pit line.

Pit Offset: Distance from pit points to where the cars would stop for pitting (this is an offset to the x local axis of the pit points).

Pit Max Speed: Max speed limiter on the pits.

Pit for Speed Limit: Number of the pit waypoint to enforce speed limit.

First pit number: Number of the pit waypoint were the actual pit stop for cars starts.

Last Pit Number: Number of the pit waypoint were the actual pit stop for cars ends.

Height: This value is for placing the waypoints just over the surface, what it does is that the System would cast a ray from the heigh specified, that way all the waypoints generated are placed exactly above the track collider.

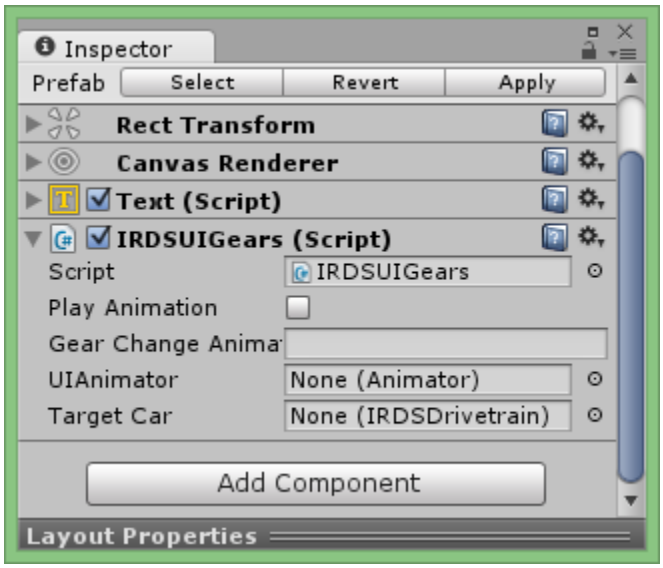
Refresh Rate: This ooption is for setting the refresh rate that would be used when changing the markers positions and the system will update the racing line and track limits lines shape.

Racing and Limits at once: Enable this option if you want to add the Track limits and the Racing line markers at the same time.

Car HUD Settings (available only on full version with own car physics)

You can access the add menu by right clicking on the hierarchy view and following iRDS->UI->Car stats and select the component you want to add.

Gear GUI Settings



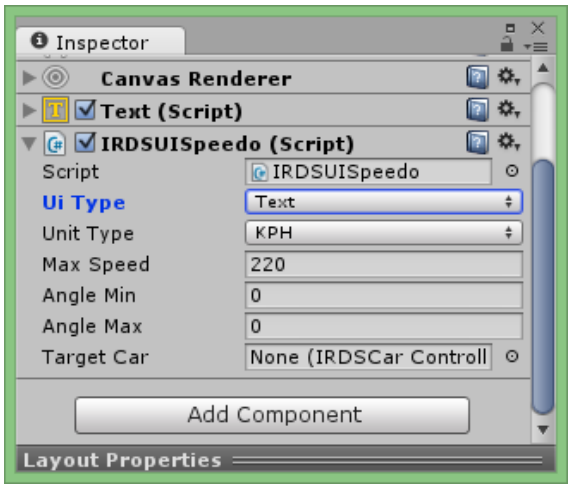
Play Animation (Toggle): This would enable an animation that would be player each time the gears are changed.

Gear Change Animation: The name of the animation state that would be triggered when the gears changes.

UIAnimator: The Animator from which the animation state would be played from.

Target Car: (Used optional) This is for assigning the target car this label would read the information from, if used with the AI System, is not needed to assign this manually. (this information applies to all the Car HUD Settings and would be mentioned only on this one to avoid repeating text).

Speedometer Settings



UI Type: The type of component you which to use, you can choose from Text, Filled and Gauge. Text would show the information on a uGUI Text component, Filled would use a uGUI Image component and would use the Image Type Filled, to increase and decrease the fill amount of the image, and the Gauge would get the transform of the object this script is attached to, to get rotated along the Z axis.

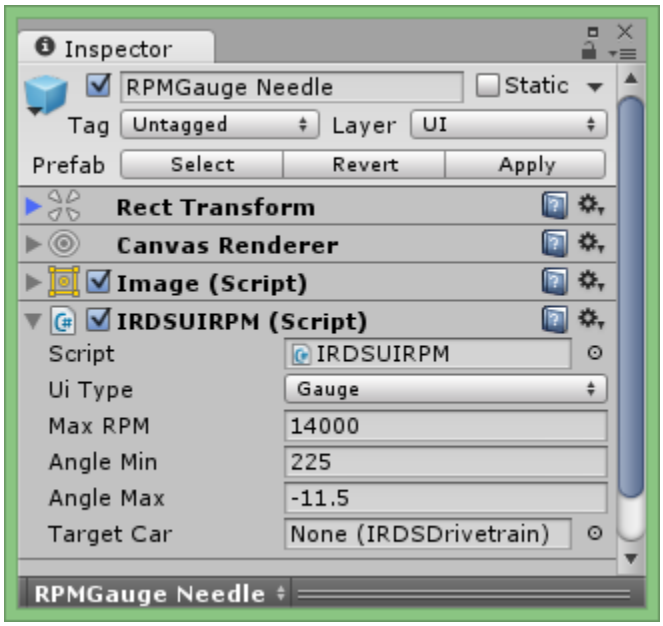
Unit Type: Can be KPH or MPH.

Max Speed: The maximum speed of this speedometer.

Angle Min: (for Gauge type only) this specifies the transform angle on the Z axis at which the gauge needle is at the actual gauge 0 value.

Angle Max: (for Gauge type only) this specifies the transform angle on the Z axis at which the gauge needle is at the actual gauge Max value (i.e. if the gauge graphics max speed is 120MPH or 200 KPH it would be the angle of the transform for it to be pointing at that value on the gauge).

Tachometer settings



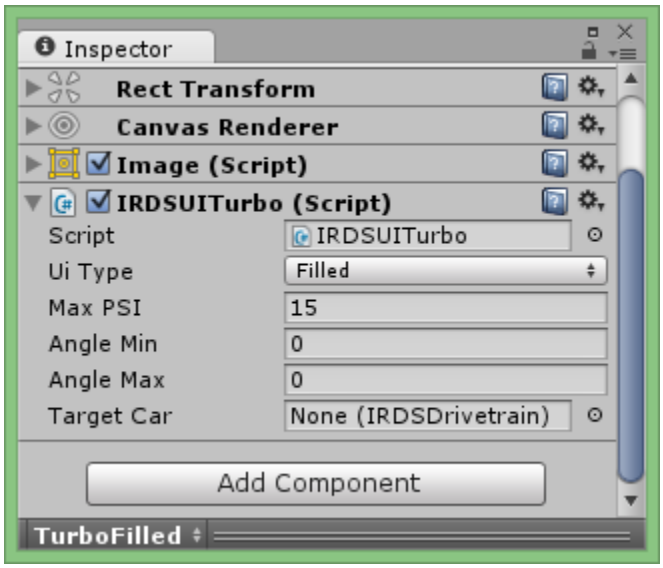
UI Type: The type of component you which to use, you can choose from Text, Filled and Gauge. Text would show the information on a uGUI Text component, Filled would use a uGUI Image component and would use the Image Type Filled, to increase and decrease the fill amount of the image, and the Gauge would get the transform of the object this script is attached to, to get rotated along the Z axis.

Max RPM: The maximum RPM of this tachometer.

Angle Min: (for Gauge type only) this specifies the transform angle on the Z axis at which the gauge needle is at the actual gauge 0 value.

Angle Max: (for Gauge type only) this specifies the transform angle on the Z axis at which the gauge needle is at the actual gauge Max value (i.e. if the gauge graphics max RPM is 9000 it would be the angle of the transform for it to be pointing at that value on the gauge).

Turbo GUI Settings



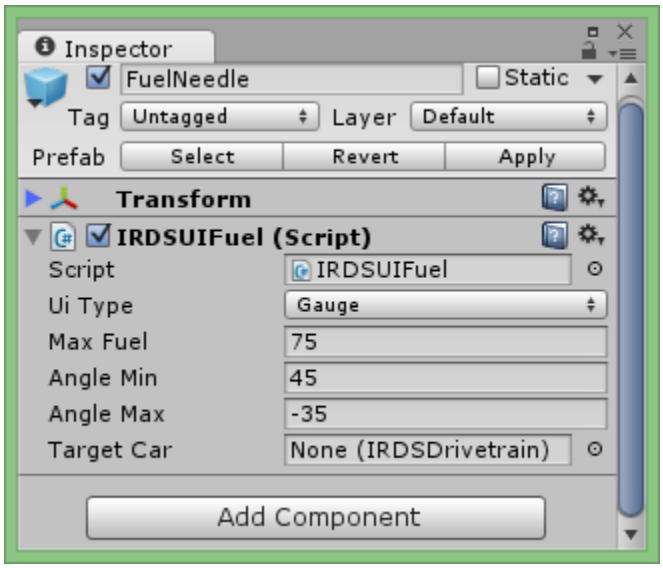
UI Type: The type of component you which to use, you can choose from Text, Filled and Gauge. Text would show the information on a uGUI Text component, Filled would use a uGUI Image component and would use the Image Type Filled, to increase and decrease the fill amount of the image, and the Gauge would get the transform of the object this script is attached to, to get rotated along the Z axis.

Max PSI: The maximum PSI of this turbo gauge.

Angle Min: (for Gauge type only) this specifies the transform angle on the Z axis at which the gauge needle is at the actual gauge 0 value.

Angle Max: (for Gauge type only) this specifies the transform angle on the Z axis at which the gauge needle is at the actual gauge Max value (i.e. if the gauge graphics max PSI is 25 it would be the angle of the transform for it to be pointing at that value on the gauge).

Fuel Settings



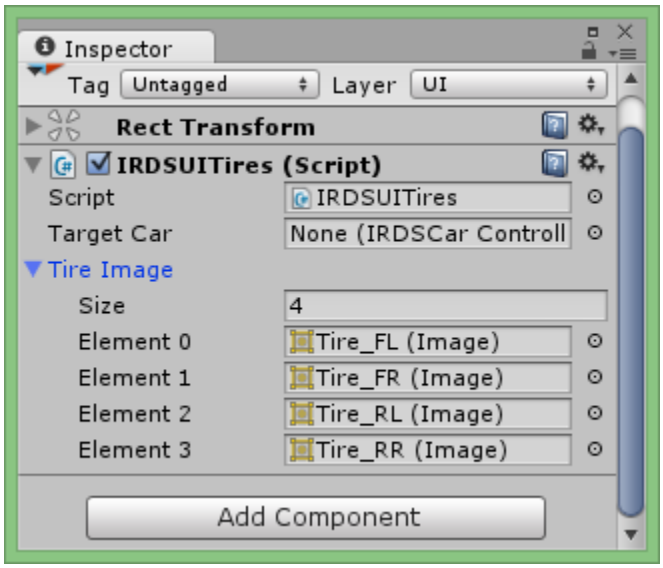
UI Type: The type of component you which to use, you can choose from Text, Filled and Gauge. Text would show the information on a uGUI Text component, Filled would use a uGUI Image component and would use the Image Type Filled, to increase and decrease the fill amount of the image, and the Gauge would get the transform of the object this script is attached to, to get rotated along the Z axis.

Max Fuel: The maximum Fuel of this Gauge.

Angle Min: (for Gauge type only) this specifies the transform angle on the Z axis at which the gauge needle is at the actual gauge 0 value.

Angle Max: (for Gauge type only) this specifies the transform angle on the Z axis at which the gauge needle is at the actual gauge Max value (i.e. if the gauge graphics max fuel is 75Kg it would be the angle of the transform for it to be pointing at that value on the gauge).

Tire GUI Settings

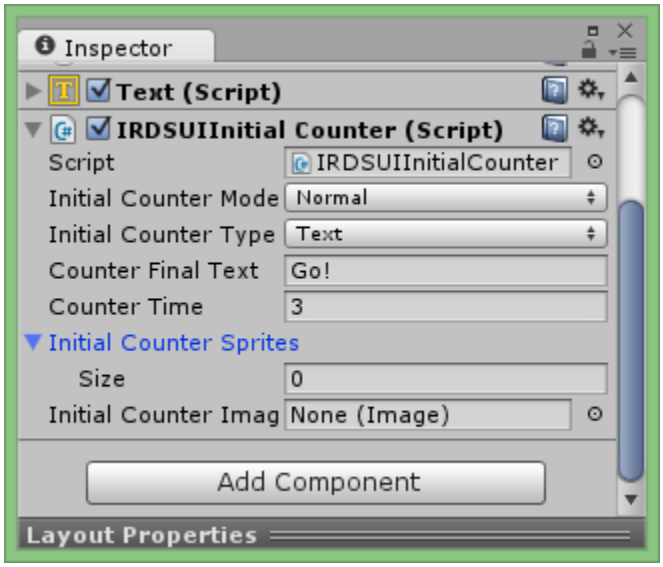


No settings to explain on this component

General HUD Settings

You can access the add menu by right clicking on the hierarchy view and following iRDS->UI->Stats and select the component you want to add.

Initial Counter Settings



Initial Counter mode: This can be normal or reverse, so the counter starts counting from 0 to the specified value, or from the specified value to 0.

Initial counter type: this can be Text or sprite array, with text, the uGUI Text component attached would be used to display the count down, if Sprite array is selected, then the counter would be switching from the first sprite till the last one on the array on each count.

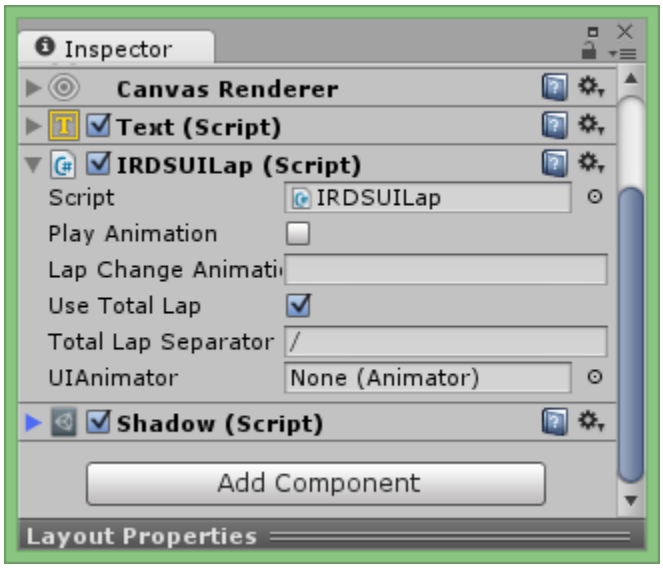
Counter final text: (used only on Text type) This would be the final text displayed when the counter ends counting.

Counter Time: The time is seconds the counter would count.

Initial Counter Sprites: This is the array of sprites that would be used to do the counting.

Initial Counter Image: This would be the uGUI Image component that would be used to display the sprites on the Initial Counter Sprites array.

Lap GUI Settings

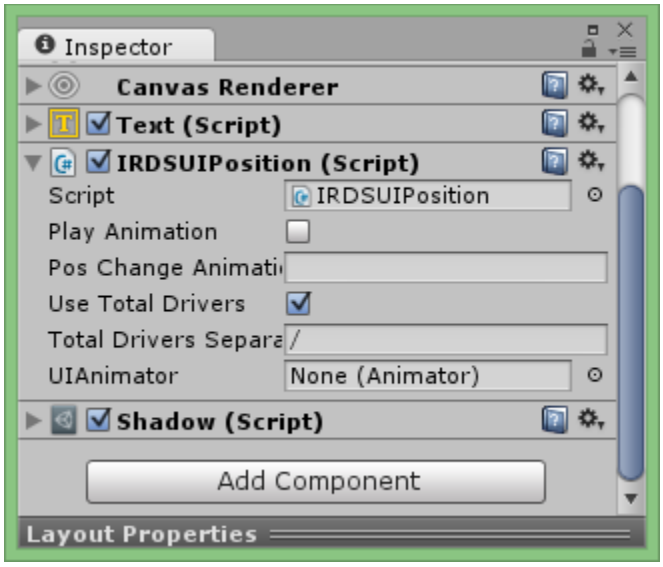


Play animation, lap change animation and UIAnimator are used the same as explained on Gear Settings.

Use Total Lap: If enabled, this component would display the current lap and the total laps this race have (i.e. 1/3)

Total lap separator: This is the string (character or any number of characters) that would be used to separate the current lap from the total laps (i.e. if the string is " of " it would display "1 of 10").

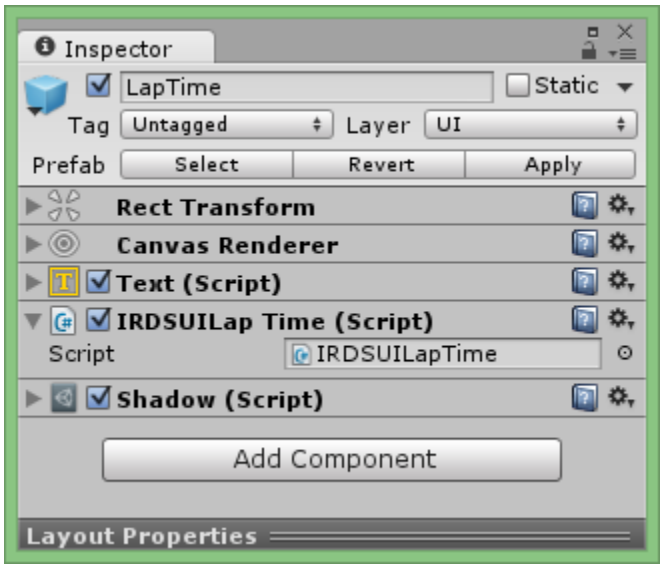
Position GUI Settings



Play animation, lap change animation and UIAnimator are sued the same as explained on Gear Settings.

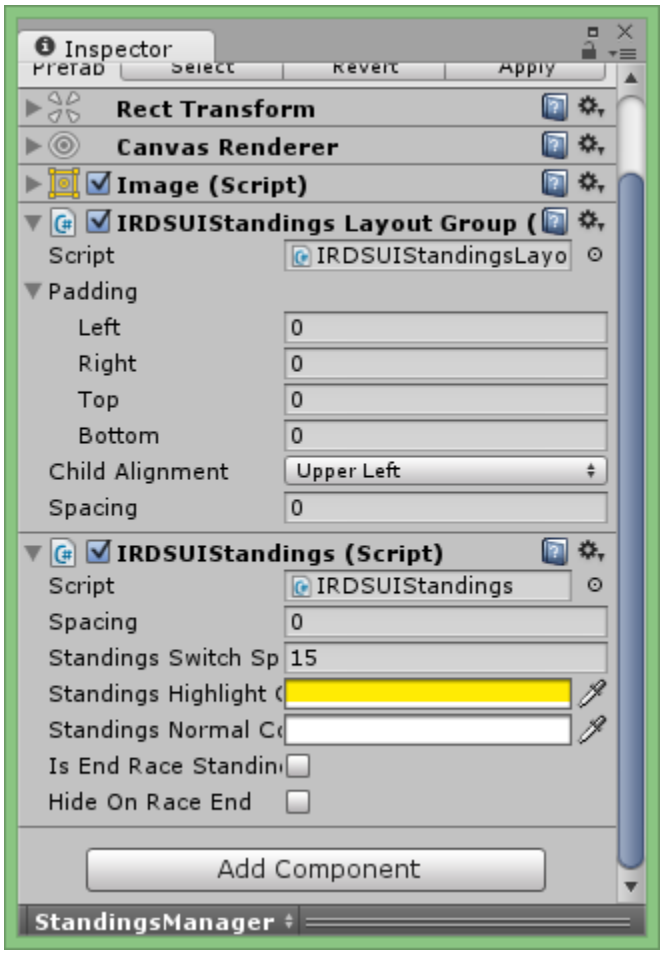
Use total drivers and Total drivers separator have the exact same functionality as explained on Lap GUI Settings section.

LapTime GUI Settings



No settings required.

Statistics GUI Settings



IRDSUIStandings Layout Group, on this component only the Spacing can be used, and it is for separating each of the items of the list of drivers this standing component is displaying.

IRDSUIStandings component:

Spacing: Not used, it would take the value of the spacing of the IRDSUIStandings Layout Group.

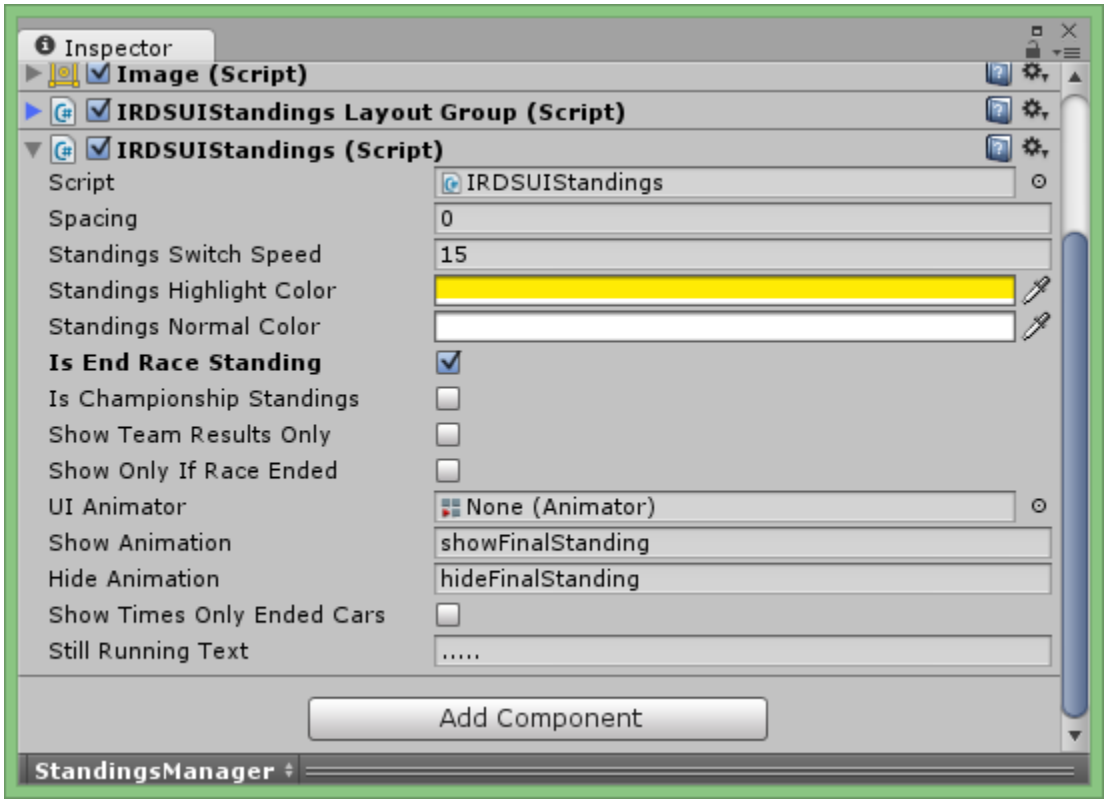
Standings Switch Speed: This is how fast the each of the items of the list would move from its place up or down on the list.

Standings Highlight color: The color an item would turn to when it matches the current car followed by the main camera of iRDS.

Standings Normal Color: The color of the rest of the items of the list.

Is End race standings(Toggle): Enable this if this is the final race standing.

Hide on Race end (toggle): Enable this if you want this standing to get hide when the race ends (requires a valid animation state and an Animator component).



Is championship Standings (toggle): Enable this if this would be used for showing the results of the championship system.

Show team results only: Enable this if you want to just show the team results (valid only if used with championship mode)

Show only if race ended (toggle): enable this if this standings should be shown only if the current car followed by the camera has ended the race.

UIAnimator: The animator that would be used to play the animations from.

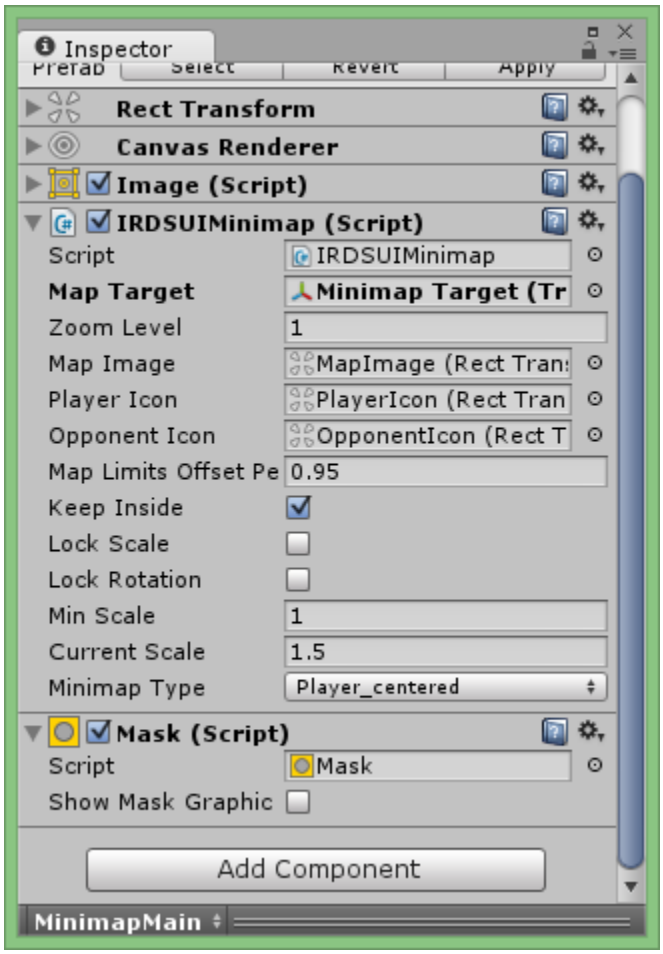
Show Animation: The state name of the animation to show this component.

Hide Animation: The state name of the animation to hide this component.

Show time only ended cars(toggle): Enable this if you want to show the times only of the cars that have ended the race.

Still Running Text: This text would be shown instead of the drivers time, if the previous option is enabled and the car have not finished the race.

Minimap GUI Settings



Map Target: This is the transform this minimap is targeting at, usually it should be an empty game object placed more or less in the middle of your track.

Zoom level: The level of zoom for this minimap, take into account that the Image that would have the actual Minimap Sprite/Texture needs to have its scale set to the real size in unity units of your track, so they match each other.

Map Image: This is the Image component that would be used to display the Minimap texture/sprite, this is automatically added.

Player/Opponent Icon: This are the images of the icons that would represent the player and the opponents on the minimap, these are added automatically (the components).

Map Limits Offset Percentage: How much to keep inside the icons from the map border.

Keep inside (Toggle): Enable if you want the icons to always be visible on the minimap.

Lock scale (toggle): Lock the scale of the minimap.

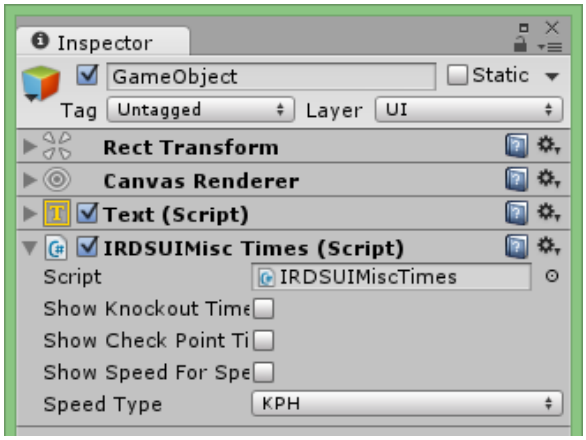
Lock Rotation(Toggle): Lock the rotation of the minimap.

Min Scale: The minimum scale of the Icons on the minimap.

Current Scale: The scale of the icon of the current car followed by the camera.

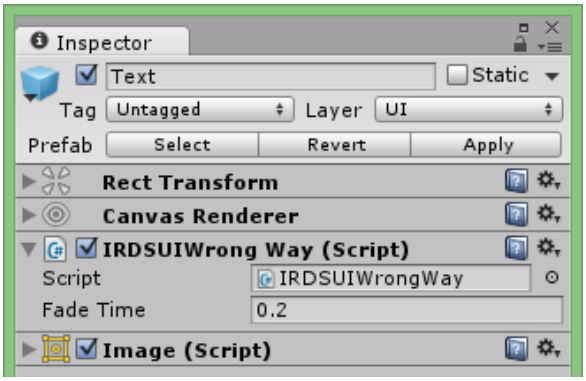
Minimap type: The type of this minimap, can be Player centered (the map would rotate around the player and would move, the player would always be on the center of the minimap. Static, the minimap wont move or rotate, all the icons would move and rotate. Line Horizontal/Vertical the icons would move horizontally or vertically on the minimap, this is for Bar kind of minimaps.

Other textures and text GUI Settings



No further explanation required.

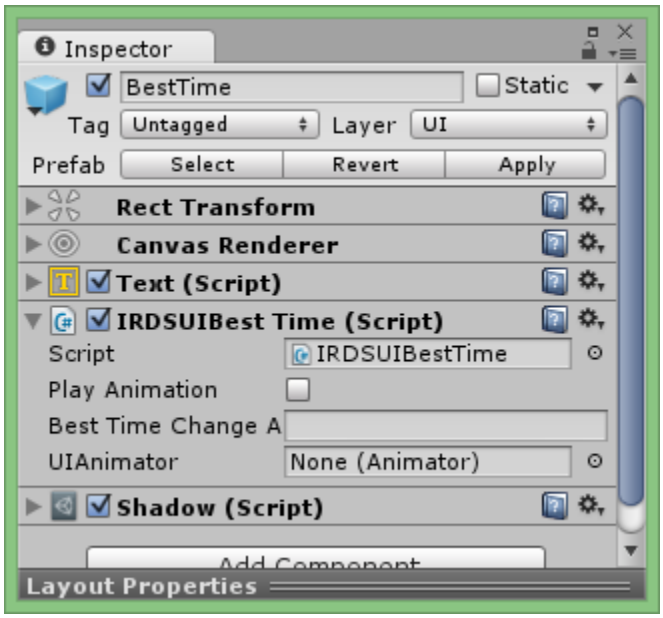
Wrong way GUI Settings

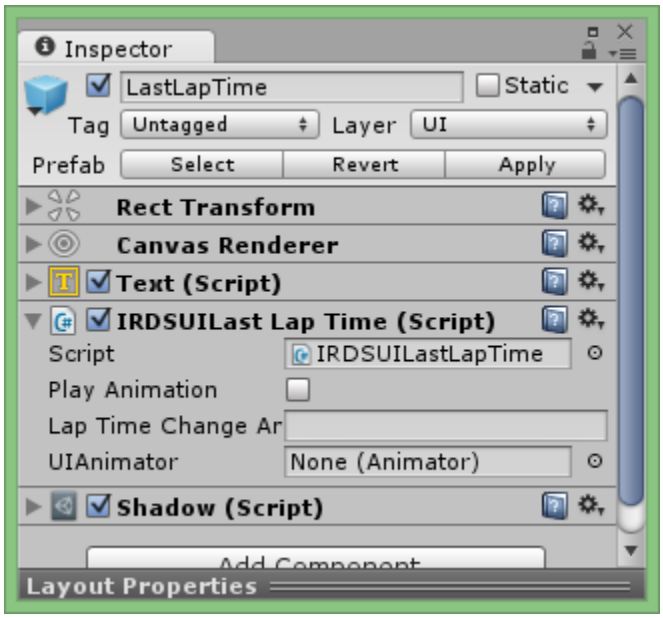


This can be attached to an Image or Text uGUI component

No further explanation required.

Best Time and Last Lap Time GUI Settings





Play animation, lap change animation and UIAnimator are used the same as explained on Gear Settings.

Level Settings

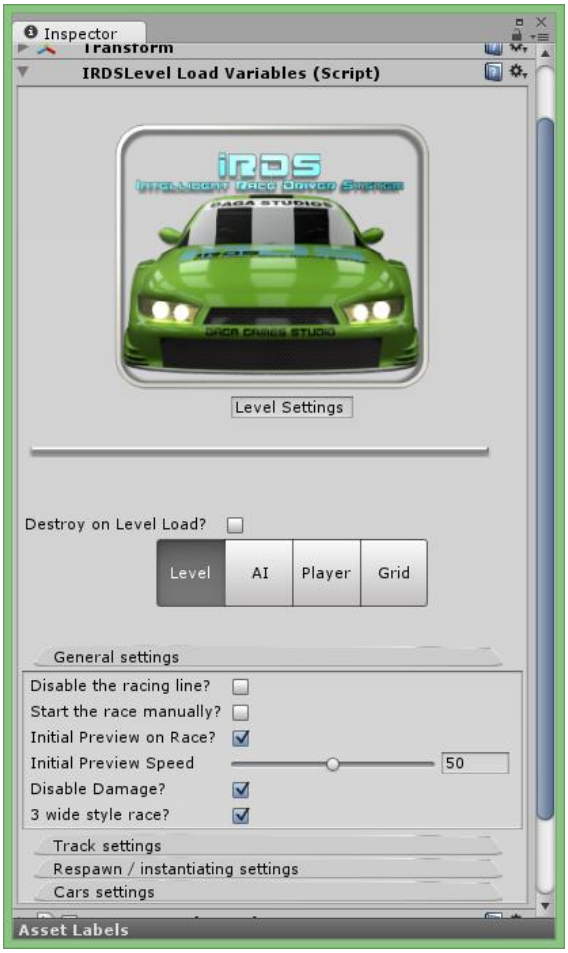
The level settings are set to tell the AI system the colors of the cars, the AI level, number of AI cars to race on the track, the track, number of laps, the car selected by the human player, if the control of the human player is digital or analog, the preferred camera view (cockpit, bumper, roof, etc.)

the shaders used on the car for the car body, so the AI system could change their color to have cars of same model but with different colors.

As you could see in the hierarchy tab, the object that contains the level settings script is called LevelLoad, this object could also be placed alone on the scene where you would let the player to choose the track to race, the car, and the general race settings, like number of opponents and laps and assign does values to this object by script using the class reference called IRDSLevelLoadVariables and keep the Destroy option unchecked, then on the scenes were you have the tracks, you won't need that object (you have to delete it) because it would remain from the scene were the player make his choices, the script for loading the scenes need to be made by you, it is not included on the system and is very easy to implement.

Level options

General Settings



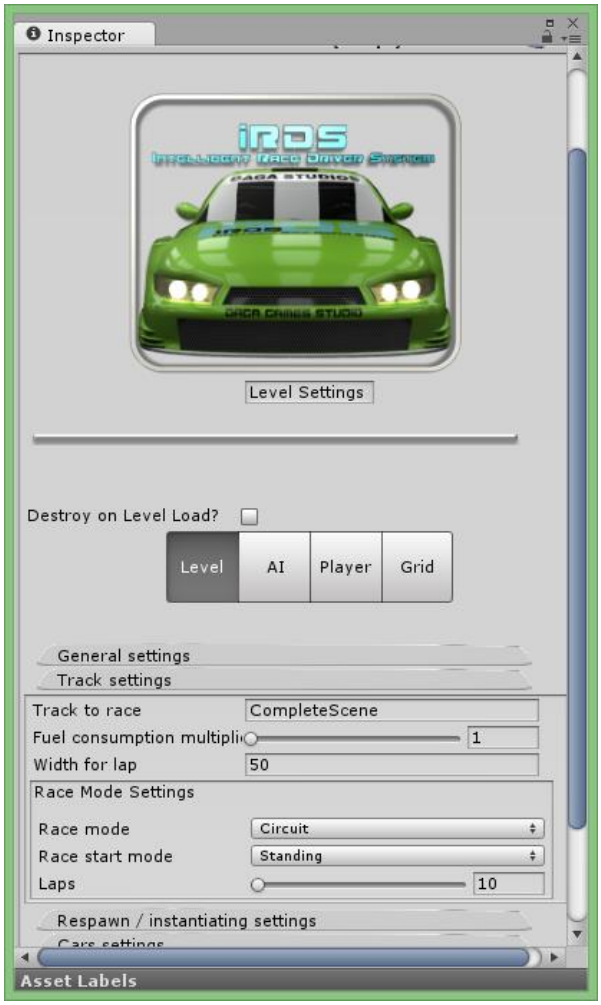
Initial Preview On Race: This allows to enable or disable the initial preview of the race.

Initial Preview Speed: The speed for the initial preview.

Disable Damage?: Check this if you want to Deactivate the damage of the cars.

Three Wide Style Race?: Check this if you want to activate 3 wide style race, this would make the AI's stay of the racing line for longer period of time.

Track Settings



Track to race: This holds the name of the scene

Fuel Consumption Multiplier: X times the normal fuel consumption for the cars.

Width for lap: The width that would be used to know if a player is out of the range for continue counting the laps (to avoid cheating).

Race Mode settings: These are the options for selecting the race mode.

- **Race Mode:** Select here the type of race (Circuit, Sprint, Last man Standing, capture the flag, check points, speed trap).
- **Laps:** Number of laps for this race. (available for circuit, Check Points and speed trap type of races only)

- **By Time:** Enable this option if you want the race be by time (Only for last man standing)
- **Time interval:** Set here the time between DNF. (Only for last man standing)
- **DNF Cars Change layer?**: Enable this option if the DNF cars should have the physics layer changed to avoid them to collide with the non DNF cars. (Only for last man standing)
- **Flag:** Drag here the flag object (this is only available for Capture the flag type of race) The flag needs to have attached to it the IRDSCarControllerAI script
- **Flag Position on waypoint?**: Should the flag be when the race starts at a waypoint?
- **Waypoint number:** The waypoint the flag would be placed at when the race starts.
- **Position:** The position at which the flag would be if the Flag position on waypoint is not selected.
- **Rotation:** The rotation at which the flag would be if the Flag position on waypoint is not selected.

Respawn / Instantiating Settings

Activate Respawn?: Check this if you want to activate Respawn of the cars.

Respawn At last WP: Check this if you want to activate respawn the car at current Waypoint Position, so i.e. if the car got off the track and is upside down, it would get respawn at the current Waypoint on the Track.

Respawn Time: Set here the respawn time. How much time the car it's upside down to respawn it.

Always respawn AI: Activate this option if you want the AI to get respawn if they get stuck.

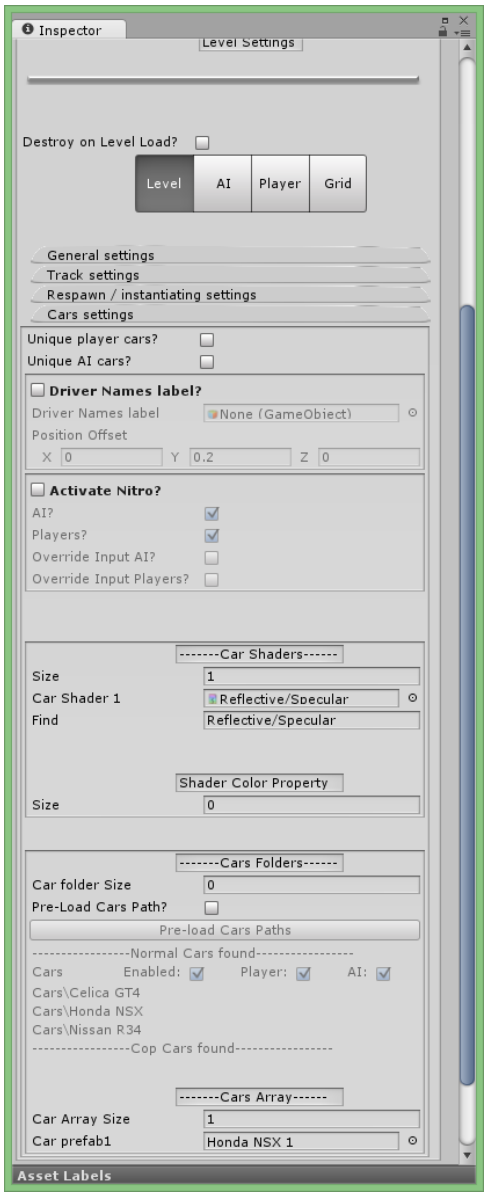
Always respawn players: Activate this option if you want the players to get respawn if they get stuck.

Respawn speed: The speed the cars would get when they are respawned.

Respawn Height: The height at which the cars would get respawned.

Instantiate Car Manually: If this is enabled, the cars must be instantiated manually, this is useful for creating Networking multiplayer games, were the networking framework requires to instantiate the network objects using the networking framework API, also the cars that gets instantiated manually, needs to get registered on the iRDS system by using the method AddNewPlayerCar and AddNewAICar from the class IRDSPlaceCars.

Car Settings



Unique Player cars: Activate this option if you want the player cars to be unique (AI won't choose this car)

Unique AI cars: Activate this option if you want the AI cars to be unique (AI won't repeat the cars)

Driver Names label?: Activate this option if you want to add the label object on top of each car.

- **Driver names label:** Drag here the game object that would be used as the car labels.
- **Position offset**: Set here the offset for positioning the car labels on each car.

Activate Nitro?: Enable this option if you want to use nitro on the race.

- **AI?:** Activate this option if you want the AI to use nitro.

- **Players?:** Activate this option if you want the players to use nitro.
- **Override Input AI?**: Activate this option if you don't want the AI to directly use the nitro, this is useful for creating kind of speed boost items on some race games like Mario kart.
- **Override Input Player?**: Activate this option if you don't want the players to directly use the nitro, this is useful for creating kind of speed boost items on some race games like Mario kart.

Car Shader: These are all the shaders that the AI would change their color at will.

Shader Color Property Name: Use this to specify the color property names of the sahders that you assigned on **Car Shader** Option, since some custom made shaders won't use _Color as default shader color.

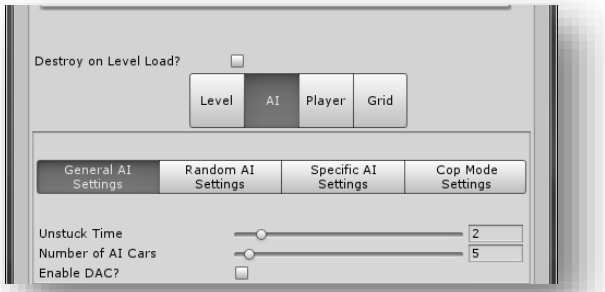
CarsFolders: This could be left in 0 size, or you could add here different folders name (should be exactly as the folders names) that should be inside a Resource folder on your project, this allows you to separate cars for example, you could have a folder named Offroaders, to put there all the 4x4 cars, another named sports cars, and put there all the coupes, etc. All the folders assigned to this variable would be used by the AI system to select them randomly from every folder added to this array.

Preload cars path?: Activate this option to pre-read all the cars folders on the resources folder, this option prevents loading all the cars when the track scene is loaded, since instead all the cars are pre-cached its name and path and this info is used to load each specific car that would race on the specific track, and avoid loading all the cars objects on the specified resources folders.

> **Pre-Load Cars Path (button):** When the preload cars path option is enabled, you must click this button in order to cache all the current cars specified on the cars folder array.
>
> **Enable:** You can choose to enable or disable this folder on the cached folders array
>
> **Player:** Enable this if you want the player been able to choose cars from this folder, otherwise disable it.
>
> **AI:** Enable this if you want the AI been able to use the cars from this folder.

Car Array: Use this if you want to manually assign all the cars that the system could choose from for the AI and the player, this is also useful for using asset bundles.

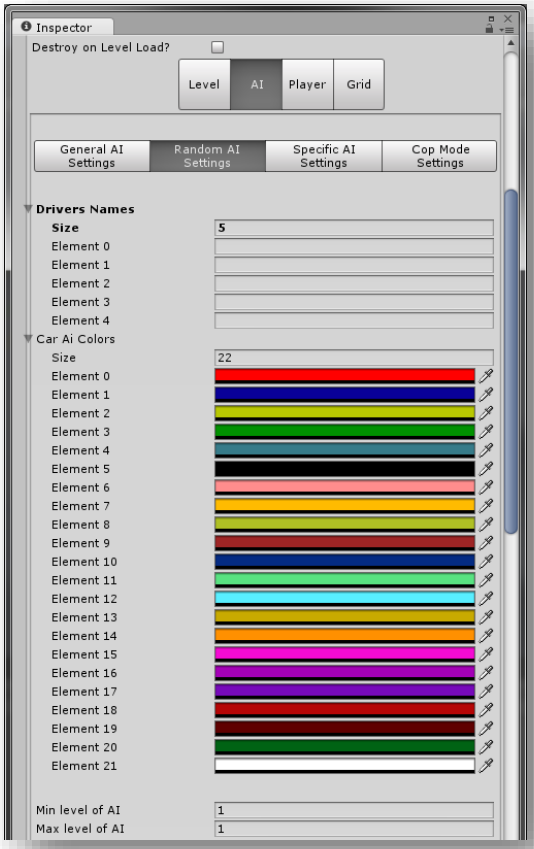
AI Options



Unstuck Time: Set here the unstuck time. How much time the AI car are stuck in order to make unstuck maneuver.

Number of AI Cars: This is the quantity of AI Drivers that would be racing against the human player.

Enable DAC: This option enables the Dynamic Aggressiveness Change for the AI's, this would make the AI to try to be at a certain distance from the player car, by slowing down on curves or speeding up.



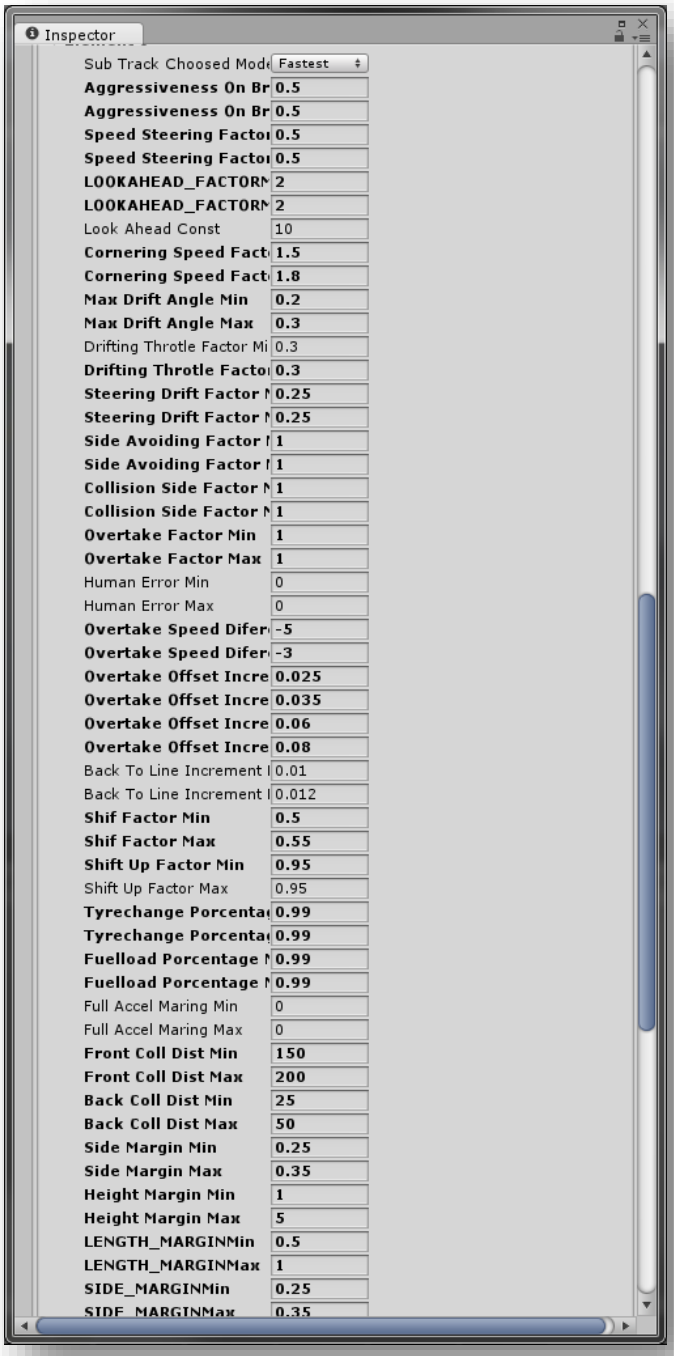
Driver Names: If you are using the AI system on any platform different from PC or Mac standalone you need to put in here all the AI racer drivers names you would like to be used by the AI system, this are just the AI Drivers names, and the values of the settings for the AI drivers would be random, rather than the ones covered early on the AI Drivers section.

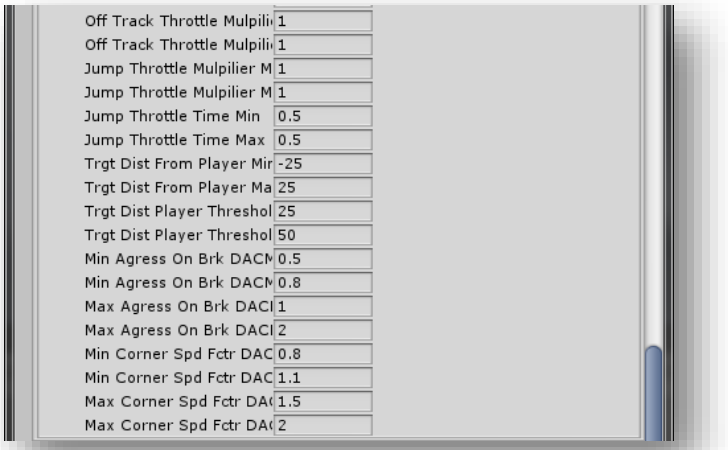
Car AI Colors: These are all the available colors for the AI Cars that would be used by the AI system to randomly chosen for the AI Cars.

For reference, this value could be used by your loading scene script to load the chosen track.

Level of AI Cars min: This is the lowest AI level you want for this race. This is related with the index number of the last parameter "Drivers Settings", so for example, you can create as many AI Custom levels ranges, let's put this way, if I create on the Drivers Settings 3 different settings, then I have 3 different AI Levels, and in this case the index 0 could be with lower values on cornering speed setting (0.85), so the AI drivers race slowly, the second index (1) could be with a little higher cornering speed setting (0.90) and on the last index you could put a cornering speed setting of 1 or above to make those AI drivers more difficult to race against them (they would do laps in less time, don't assign values to high above 1 to the cornering speed, or the AI drivers could begin to skid too much on corner and lose control).

Level of AI Cars max: This is the maximum AI level you want for this race. See previous parameter for a more detailed explanation.


Inspector
Sub Track Choosed Mode
Fastest
Aggressiveness On Br
0.5
Aggressiveness On Br
0.5
Speed Steering Factor
0.5
Speed Steering Factor
0.5
LOOKAHEAD_FACTORM
2
LOOKAHEAD_FACTORM
2
Look Ahead Const
10
Cornering Speed Fact
1.5
Cornering Speed Fact
1.8
Max Drift Angle Min
0.2
Max Drift Angle Max
0.3
Drifting Throtle Factor Mi
0.3
Drifting Throtle Factor
0.3
Steering Drift Factor M
0.25
Steering Drift Factor M
0.25
Side Avoiding Factor M
1
Side Avoiding Factor M
1
Collision Side Factor M
1
Collision Side Factor M
1
Overtake Factor Min
1
Overtake Factor Max
1
Human Error Min
0
Human Error Max
0
Overtake Speed Difer
-5
Overtake Speed Difer
-3
Overtake Offset Incre
0.025
Overtake Offset Incre
0.035
Overtake Offset Incre
0.06
Overtake Offset Incre
0.08
Back To Line Increment M
0.01
Back To Line Increment M
0.012
Shif Factor Min
0.5
Shif Factor Max
0.55
Shift Up Factor Min
0.95
Shift Up Factor Max
0.95
Tyrechange Porcentage
0.99
Tyrechange Porcentage
0.99
Fuelload Porcentage M
0.99
Fuelload Porcentage M
0.99
Full Accel Maring Min
0
Full Accel Maring Max
0
Front Coll Dist Min
150
Front Coll Dist Max
200
Back Coll Dist Min
25
Back Coll Dist Max
50
Side Margin Min
0.25
Side Margin Max
0.35
Height Margin Min
1
Height Margin Max
5
LENGTH_MARGINMin
0.5
LENGTH_MARGINMax
1
SIDE_MARGINMin
0.25
SIDE_MARGINMax
0.35



Drivers Settings: You need to configure the min and max values, in order for the AI system to use them for randomly assigning them to the AI Drivers, always the min value has to be less or equal to the max value, here are the variables included on this array:

Aggressiveness on Brake: Values greater than 1 would make the AI Driver to brake earlier and values lower than 1 would make the AI brake later and Harder. (The best values are between 1 and 1.5)

Speed steering Factor: This value determines the minimum speed for the AI to let go of the throttle if the Gforce is greater than the force the tires can exert.

Look Ahead Factor: This value would affect how the AI would anticipate for turning on curves, lower values would make the AI less anticipated and higher values would make the AI more anticipated. (Best values are between 1 and 4, excepting edy's VP version, best values are between 0.2 amd 0.5).

Look Ahead Const: This parameter sets the minimum look ahead distance for the AI to follow on the waypoints, the default value is 10.

Cornering Speed Factor: This values controls how fast the AI would take the curves, values below 1 would make the AI take the curves slower, and values above 1 would take curves faster. Be aware that values above 1 tend to over exceed the maximum corner speed, and would tend to make the AI lose control of the car. (Best values are between 1.3 and 1.6, depending on the track could be higher till 2).

Max Drift Angle: This value set the maximum angle between the car yaw direction and the car movement direction in order to make counter drifting maneuvers. (Best values are between 1 and 5).

Drifting throttle factor: This parameter would make the AI to loose on the throttle if the car is drifting, the lower this value the more the AI would stop throttling, values of 0.3 and lower would make this to happen.

Steering Drift Factor: This value influences how much the AI would counter steer, on a counter steering maneuver. (Best values are between 0.2 and 0.5)

Side Avoiding Factor: This value set how much the AI would react to getting out of the track, values greater than 1 would make the AI react faster, and less than 1 would react slower. (Best values are between 0.9 and 1.2, preferable a value of 1)

Collision Side Factor: This value set how much the AI would react to getting near to the side of another car, values greater than 1 would make the AI react more, and less than 1 would react less.(Best values are between 0.9 and 1.2, preferable a value of 1)

Overtake Factor: This value set how much the AI would react to overtaking an opponent, values greater than 1 would make the AI react more, and less than 1 would react less. (Best values are between 0.9 and 1.2, preferable a value of 1)

Human Error: How many errors would the AI do, bigger values would make the AI do more and bigger mistakes, and lower would make the AI do less mistakes. (Best values are between 0.001 and 0.05, preferable 0.0)

Overtake Speed Difference: This value sets how much speed difference should be between this AI Driver and another opponent in order to make an overtake maneuver. (Best values are between -3 and 0)

Overtake offset increment Min: This is the overtake step that would be used for the AI's to make their maneuvers (overtake and avoid). (Best values are between 0.015 and 0.03)

Back To line increment: This is the back to the racing line step that would be used by the AI when they have finished a maneuver to get back to the racing line. (Best values are between 0.09 and 0.3)

Shift Factor: This value set on a percentage basis, when to make shifts on the rpms range, for example, using a 0.5 value, would make the AI shifting gears at the middle of the Rpm range. (Best values are between 0.4 and 0.6)

Shift Up Factor: This value set on a percentage basis, when to make shifts Up on the rpms range, for example, using a 0.95 value, would make the AI shifting gears at the higher of the Rpm range. (Best values are between 0.4 and 0.6)

Tire change %: This value sets when the AI would enter the pits to change tires, it is a percentage of the tire wearing, and higher value would enter pits early. (Best values are between 0.1 and 0.15)

Fuel Load %: This value sets when the AI would enter the pits to load fuel, it is a percentage of the fuel tank capacity, and higher value would enter pits early. (Best values are between 0.05 and 0.15)

Full Accel Margin: Experimental value, it is intended to keep a margin between full acceleration and none depending on the distance between the AI Driver and a curve or another opponent. (Best values are between 1 and 2)

Front coll distance: The front distance the AI would search for opponents. (Best values are between 100 and 250, also could be less, but with values less than 100, if a car is not moving and in the middle of the racing line, the possibility that the AI would avoid it is less, since it won't see the car until it is too close)

Back coll distance: The back distance the AI would search for opponents. (Best values are between 25 and 50)

Side Margin (track sides): How near the track limit the AI should be in order to react to get back on the track. (Best values are between 0.5 and 2)

Height Distance: The height distance the AI would search for opponents. (Best values are between 5 and 10, this value depends more if the track has some parts with bridges, like in the CarTutorial scene, so the AI won't search for opponents higher than that value)

Length Margin: How near the AI should be to an opponent to fully press the breaks to avoid collision. (Best values are between 0.5 and 2)

Side Margin (opponents): How near the AI should be to an opponent side in order to make an avoiding maneuver. (Best values are between 0.1 and 0.5)

Off track throttle multiplier: This parameter is multiplied to the throttle if the AI gets out of the track, this is to avoid the AI to start drifting outside of the track if the terrain is sand, for example.

Jump throttle multiplier: This parameter is multiplied to the throttle if the AI makes a jump; this is to avoid the AI to lose control after the jump.

Target Distance from player: This parameter is used if the DAC option is enabled, and this would be the target distance the AI would try to keep from the player.

Target Distance player threshold: This parameter is used if the DAC option is enabled, and this would be the threshold distance the AI would use to lerp from the min aggressiveness to the max.

Min Aggressiveness on brake DAC: This is the minimum aggressiveness on brake used when the AI is out of the range from the player (the AI is behind of the player).

Max Aggressiveness on brake DAC: This is the maximum aggressiveness on brake used when the AI is out of the range from the player (the AI is ahead of the player).

Min cornering speed factor DAC: This is the minimum cornering speed factor used when the AI is out of the range from the player (the AI is ahead of the player).

Max cornering speed factor DAC: This is the maximum cornering speed factor used when the AI is out of the range from the player (the AI is behind of the player).

Assign Cars to AI Drivers: This option will enable the use of the next array, which would let you specify which drivers with its car and car color would race on the scene, if you enable this option you have to fill the next array “Cars Drivers”.

Add: Add a new AI driver to the list.

Remove: Remove this AI driver from the list.

Car: This is a string so you could write here the name of the car (Prefab) you want to assign to this driver, the name have to be exactly as the prefab on the “Cars” folder inside the resource folders (iRDS/Resources/Cars) or any other folder you assigned on the “Cars Folders” Array (Remember this folder most exits inside the Resources folder).

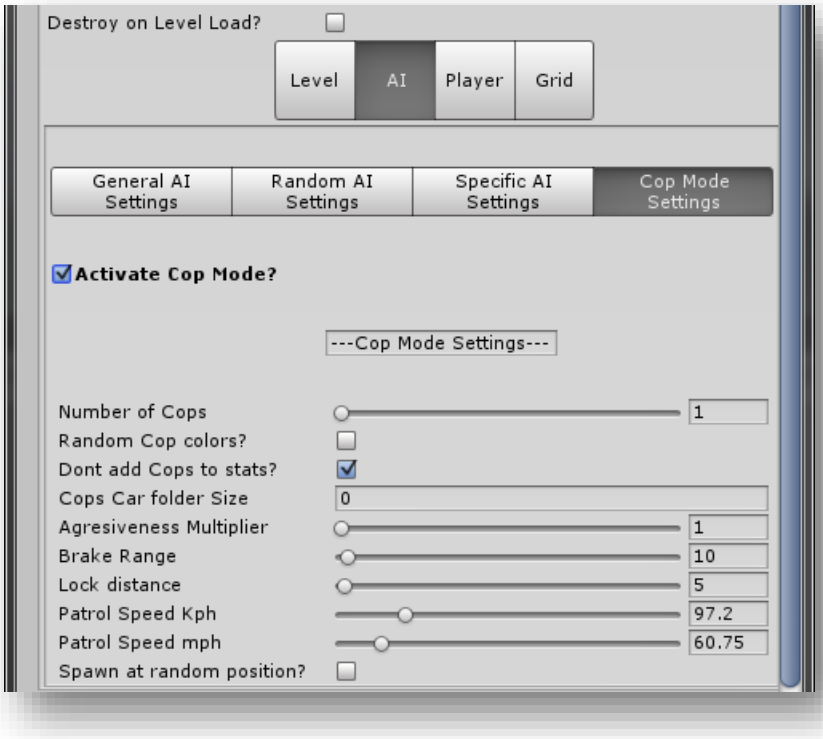
Driver Name: This is a String and you should write here the name of the Driver, if you have selected the “Random AI Drivers” bool to true then this names are the names of the drivers that would race against the player, if the “Random AI Drivers” bool is set to false (unchecked) then you

have to create the AI Drivers (See “AI Driver – Setting up new drivers and modifying the existing ones”) and use those names (exactly as you created them) to assign them here in this array in this variable.

Car Color: This would be the color of the AI’s Car.

Grid Position Number: The actual grid position of the AI car. (should not be modified here).

Own Settings?: Enable this option if you want this driver to have its own settings. If it is not enabled this driver would get random settings from the random drivers settings Tab.



Activate Cop Mode?: This option would activate the cop mode.

Number of cops: The number of cops for this race:

Random Cop colors?: Enable this option if you want the cop cars to have random colors.

Don’t add cops to stats?: Enable this option, if you don’t want the cops to be added to the standings.

Cops car folder size: This is the size of the array to hold the names of the cops cars folders, the system would get the cars for the cops from these folders (these folders need to be under the resources folder).

Aggressiveness multiplier: This value would be multiplied by the cornering speed factor, to make the cops try to catch the players or non cops AI's.

Brake Range: This is the distance in unity units that the cops would start to brake if they are ahead of the target car.

Lock Distance: Range in unity units for the cops directly follow the target.

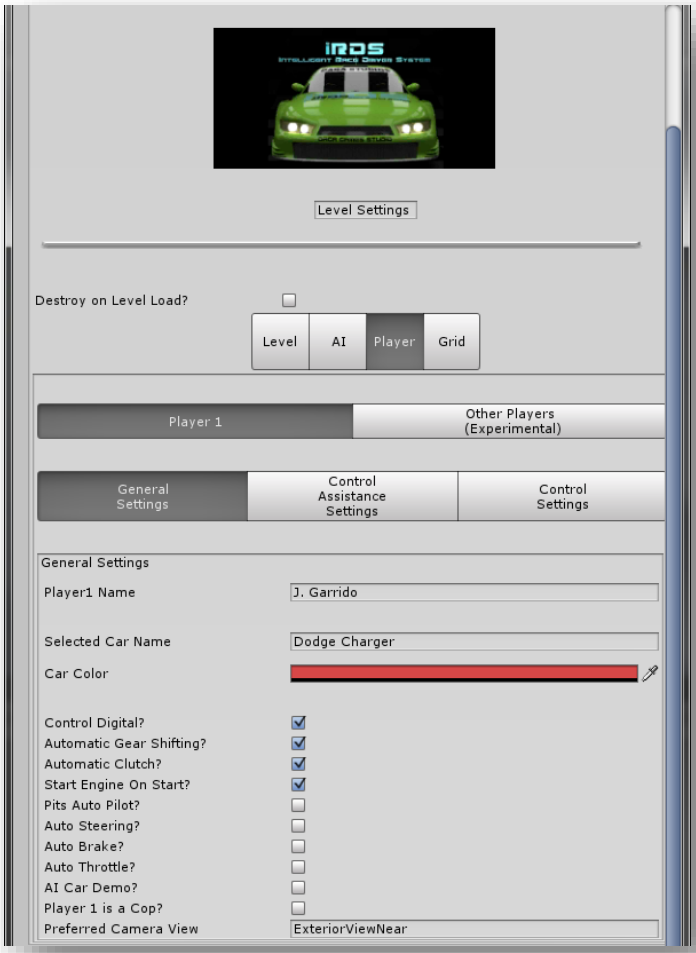
Patrol Speed KPH: The patrolling mode speed of the cops in kph.

Patrol Speed MPH: The patrolling mode speed of the cops in mph.

Spawn at random positions?: This option, if enabled, would make the cops to be spawned along the track, instead of spawning them on the grid.

Player Options

General settings



Player Name: Name of the first player.

Selected Car: This is the exact prefab name of the car selected by the player and that would be loaded by the System.

Selected Car Color: This is the color for the selected car of the player.

Control Digital: To be selected by the player, if checked (true) the control is digital, if not it is analog.

Automatic Gear Shifting: enable this option to have the player car in automatic transmission mode.

Automatic Clutch: enable this option to have the player car in automatic clutch mode.

Automatic Start Engine on Start: enable this option to have the player car start its engine automatically before the race starts.

Pits Auto Pilot: enable this option to have the player car in automatic pits driving mode, the AI would take control of the car when the player enters the pits.

Auto Steer: enable this option to have the player car in automatic Steer mode.

Auto Brake: enable this option to have the player car in automatic Brake mode.

Auto Throttle: enable this option to have the player car in automatic Throttle mode.

AI Car Demo: This option would make the player car to be driven by an AI Driver for testing purpose.

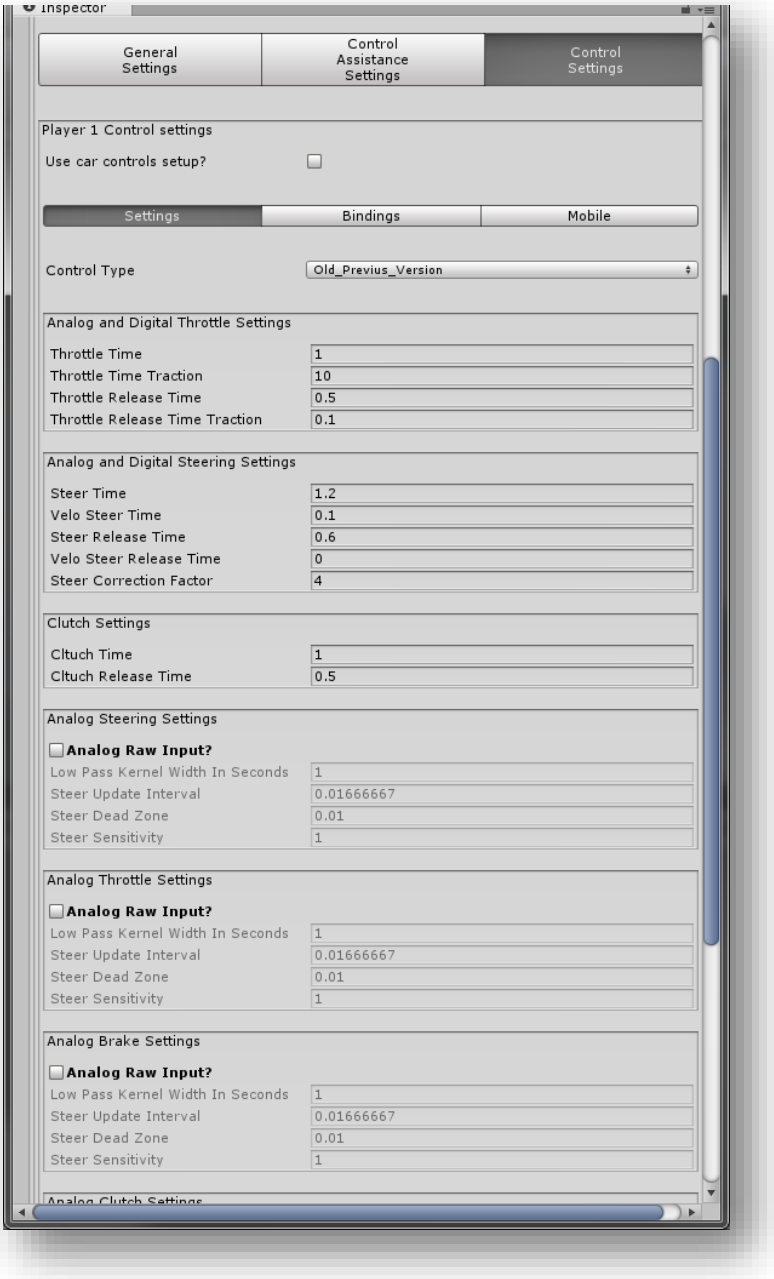
Player 1 is cop?: This option would make the player car to be a cop.

Preferred camera view: The initial camera view for the player (cockpit, bumper, roof, exterior, etc.)

Control Assistance Settings

This options allows to set the assistance settings of the player car, allowing to setup the ABS, Traction control, ESC and steer help values of the car.

Control Settings

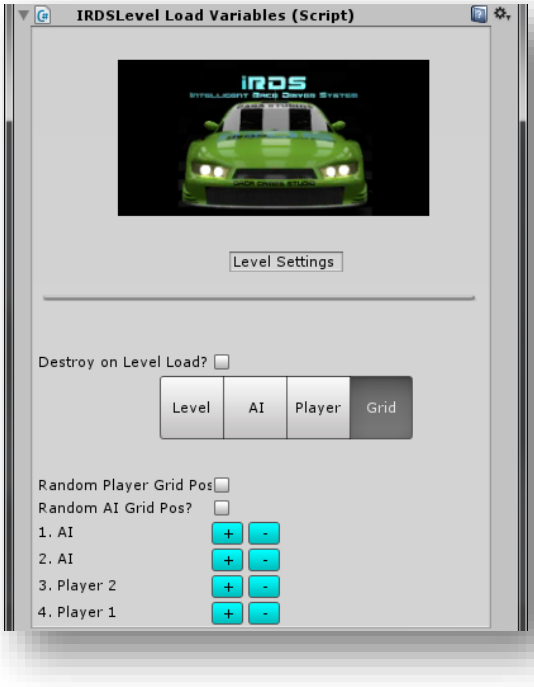


This section allows to setup the car control settings as in the own car Control settings, see car control settings section for details.

Other Players: On this array you can setup as many players as you want, be aware this is experimental, since the system doesn't handle yet Split Screen, nor multi audio sources, use it for experimenting with it. The parameters used here are the same as explained before for the first player. All the players included here start on the second player and so on, since the first player is defined on the first part of Player Options.

Grid Options

This section is about the grid positions options. On this version of iRDS is possible to assign the player grid position and AI if needed.

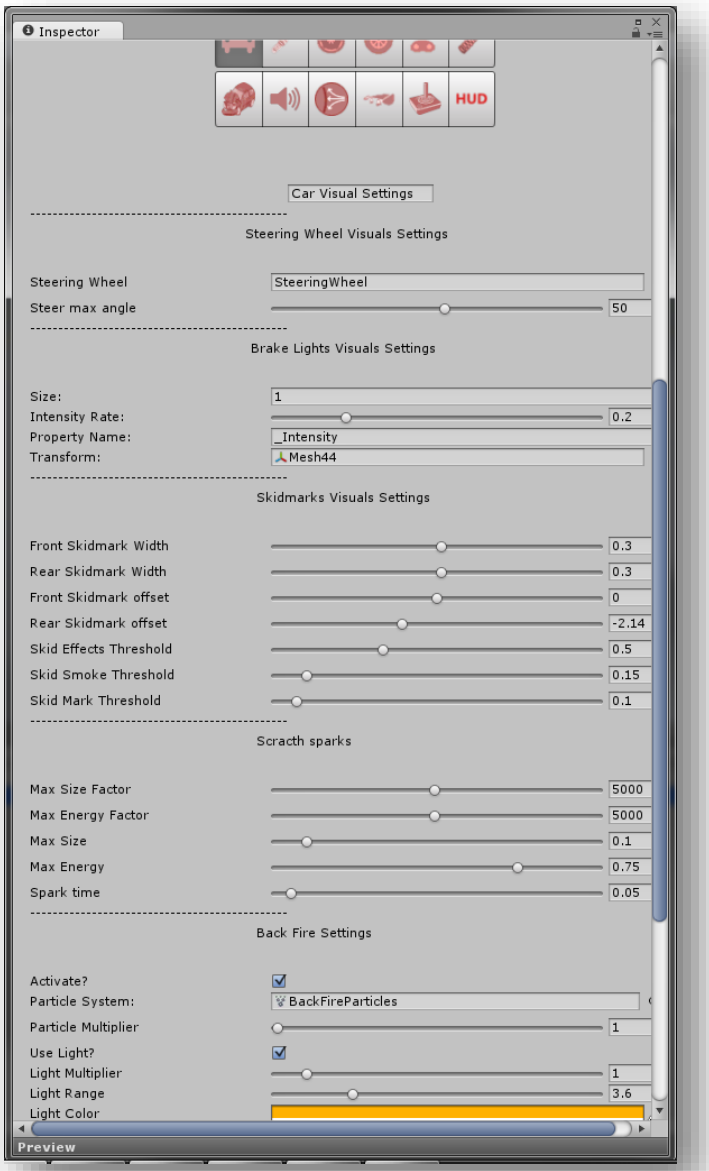


Random player grid position: If enabled, the system would assign the players grid positions randomly.

Random AI grid positions: If enabled, the system would assign the AI grid positions randomly. (this option is only available if the Assign AI drivers to car is enabled on the AI Tab.

Car Physics Settings

Car Visual Settings



Steering Wheel Visuals Settings

Steering Wheel: Assing here the GameObject that contains the Steering Wheel Mesh.

Steer max angle: Set the max angle for the steering wheel mesh to be rotated when the user apply steer inputs.

Brake Lights Visuals Settings

Size: The number of brake lights objects this car have

Intensity rate: Here you can adjust the intensity rate of the brake light, to adjust how quick the lights fade on and off.

Property name: The property name of the shader that would be used for the light intensity.

Transform: Here you can assign all the Game Objects that represents the brake lights.

Skidmarks Visuals Settings

Front Skidmark Width: Front tires skidmark's width.

Rear Skidmark Width: Rear tires skidmark's width.

Front Skidmark offset: Front tires skidmarks offset, with this setting the skidmark position relative to the tire can be corrected.

Rear Skidmark offset: Rear tires skidmarks offset, with this setting the skidmark position relative to the tire can be corrected.

Skid Effects Threshold: The greater this value, the more wheel spin and skid is needed to show skid smoke and skid marks effects.

Skid Smoke Threshold: The greater this value, the more wheel spin and skid is needed to show skid smoke effects.

Skid Mark Threshold: The greater this value, the more wheel spin and skid is needed to show skid marks effects

Scratch Sparks

Max Size Factor: If this value is higher the sparks would be smaller.

Max Energy Factor: Higher values makes the sparks disappear early.

Max Size: Maximum size of the sparks.

Max Energy: Maximum energy of the sparks particles.

Spark time: time between each spark effect.

Back Fire Settings

Activate?: Enable back fire effect?

Particle System: You can set here your custom particle system (it has to be Shuriken type))

Particle Multiplier: Multiplier for the particle start size

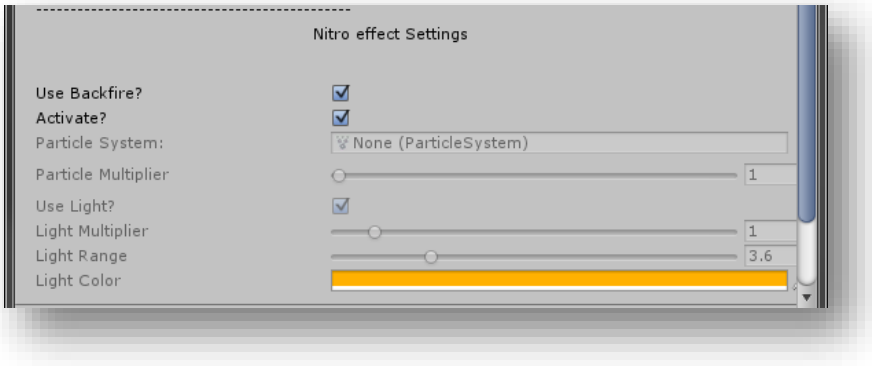
Use Light?: Enable light for back fire effect

Light Multiplier: Light Intensity Multiplier

Light Range: Light Range

Light Color: Color of the light.

Nitro Effect Settings



Use BackFire?: Use for the nitro the same backfire particle?

Activate?: Enable nitro effect?

Particle System: You can set here your custom particle system (it has to be Shuriken type))

Particle Multiplier: Multiplier for the particle start size

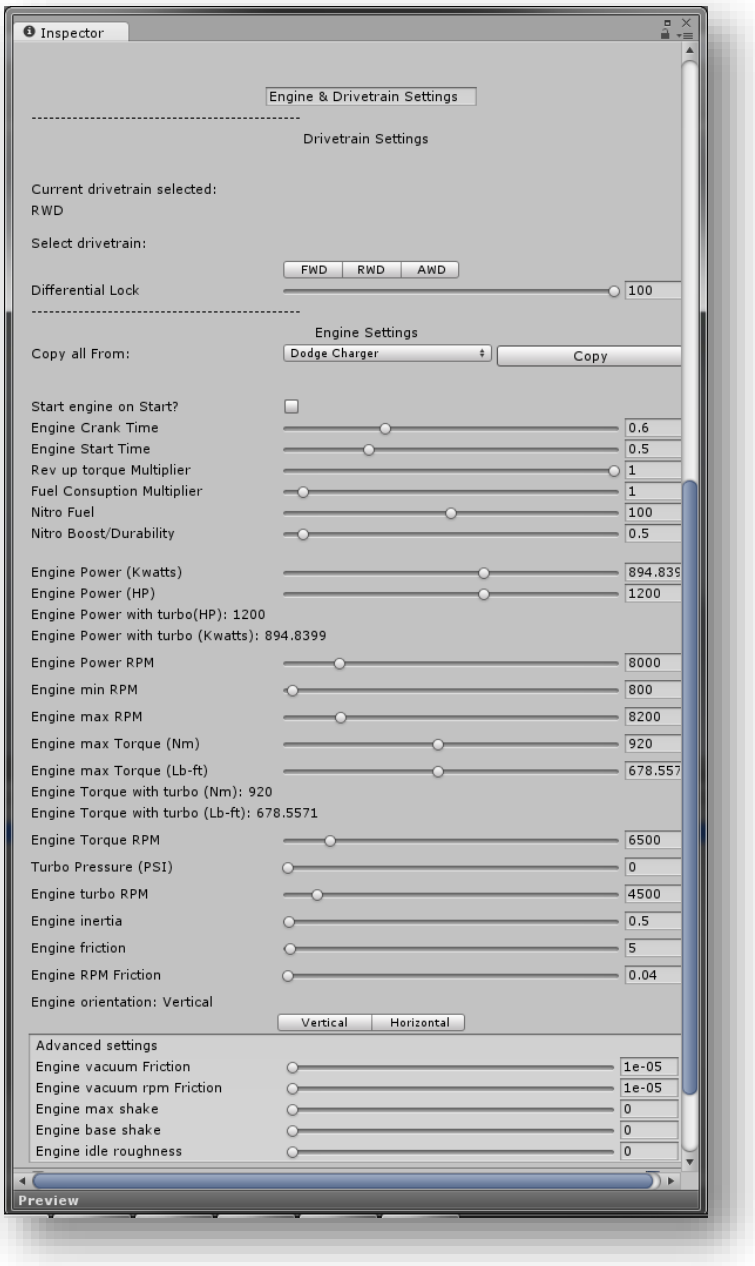
Use Light?: Enable light for back fire effect

Light Multiplier: Light Intensity Multiplier

Light Range: Light Range

Light Color: Color of the light.

Engine & Drivetrain Settings



Drivetrain Settings

Current drivetrain selected: Here you can see the current selected drivetrain for this car.

Select drivetrain: Here you can select the desired drivetrain for this car. FWD, RWD, AWD.

Differential Lock: Set the differential lock coefficient, this is to simulate the differential lock that would keep tires spinning with the same velocity or not, set it to 100 if you want to activate full differential lock and to 0 to deactivate it.

Engine Settings

Copy All From: Here you can select another car to copy the engine values from, please note that there is no Undo option, so if you copy the values from another cars, the older ones would be lost.

Start engine on Start?: Auto start the engine?.

Engine Crank Time: Set the engine time to crank.

Engine Start Time: Set the engine time to start.

Rev up torque multiplier: This value would be multiplied by the torque of the car that is applied to the rigidbody, this is useful to make the torque smaller on smaller cars to avoid the car to roll.

Fuel Consuption Multiplier: Set the fuel consuption multiplier, greater values would make engine consume the fuel faster.

Nitro Fuel: Set the amount of Nitro.

Nitro Boost/Durability: Lower values makes less boost and enhance the Nitro Durability, higher values increase boost, and lowers the nitro durability.

Engine Power (Kwatts): Set the engine power of the car in Kilowatts.

Engine Power (HP): Set the engine of the car in HP (Horse Power).

Engine Power RPM: Set the engine RPM at which would have its maximum power, this is used to simulate the engine power curves.

Engine min RPM: Set the engine min RPM (this would be the idle rpm).

Engine max RPM: Set the engine max RPM.

Engine max Torque (Nm): Set the engine max torque in Newton's meters.

Engine max Torque (Lb-ft): Set the engine max torque in pounds per feet.

Engine Torque RPM: Set the engine RPM at which would have its maximum torque, this is used to simulate the engine torque curves.

Turbo Pressure (PSI): Set the turbo air pressure in PSI.

Engine turbo RPM: Set the engine RPM where the turbo would be generating its maximum air pressure.

Engine inertia: engine inertia (how fast the engine spins up), in kg * m^2, lower values would make the engine spin up faster.

Engine friction: constant friction coefficient, these cause the engine to slow down, and cause engine braking.

Engine RPM Friction: Linear friction coefficient (higher friction when engine spins higher).

Engine orientation: The position of the engine, it could be vertical or horizontal.

Advanced settings

Engine Vacuum friction: Linear friction coefficient for simulating the vacuum brake effect on the engine

Engine vacuum rpm Friction: Linear friction coefficient for simulating the vacuum brake effect on the engine depending on rpm

Engine max shake: The maximum shake torque of the engine on idle

Engine base shake: The base shake torque of the engine on idle

Engine idle roughness: How rough the idle should be, 0 smooth and 1 rough

Gear Settings

Copy All From: Here you can select another car to copy the Gear values from, please note that there is no Undo option, so if you copy the values from another cars, the older ones would be lost.

Automatic Transmission: This toggles automatic gear shifting.

Automatic Clutch?: Select this option if you want automatic clutch, unselect it to use manual clutch.

Shift Up RPM %: This sets the RPM % at which gear would be shifted up if automatic transmission is enabled.

Shift Down RPM %: This sets the RPM % at which gear would be shifted down if automatic transmission is enabled.

Number of gears: Set this to be the amount of gears the transmission would have, normal cars have 1 gear for reverse, 1 for neutral and five for forward, making a total of 7 gears.

Gears: Adjust each gear values to match the gears of the real car you want to simulate.

Final gear ratio: This is the final drive gear ratio.

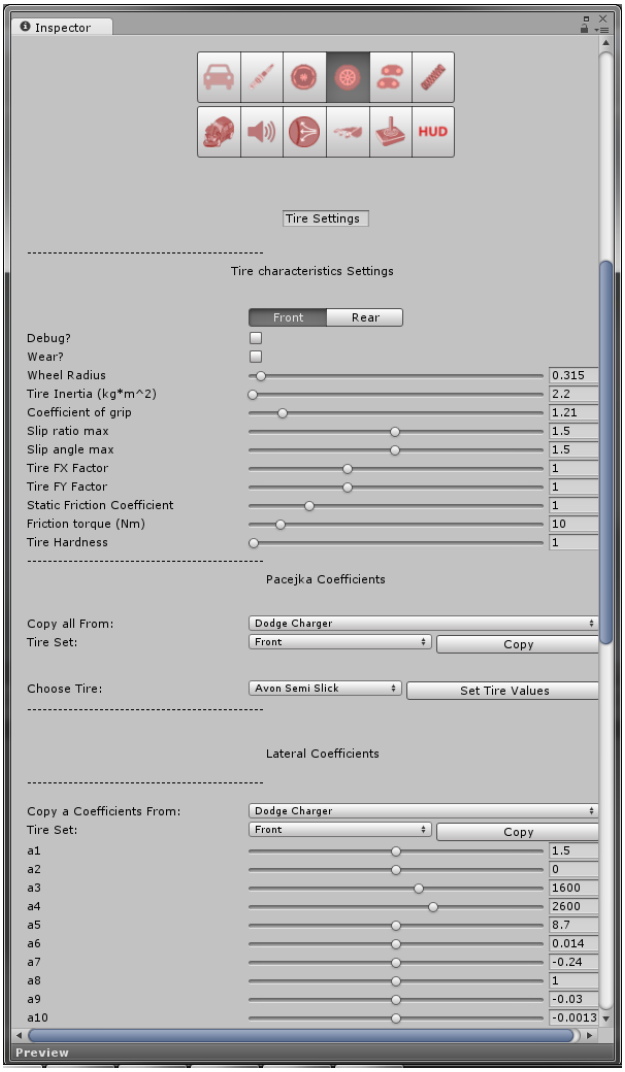
Automatic Shift Speed: This would set the amount of throttle that would be released when shifting gears in automatic mode, greater values would make more throttle release, resulting in slower gear shift or accel.

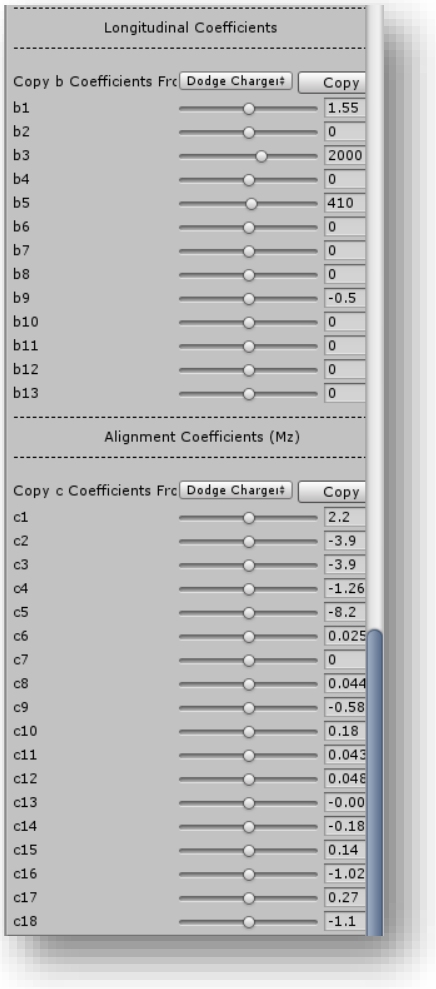
Transmission Shift Speed: This would set the speed or time between each gear shift

Is Top Speed Limited?: This car would have its top speed manually limited?.

Top Speed (KPH / MPH): The current top speed of the car, using the current gears and final gear ratio settings.

Tire Settings





Copy All From: Here you can select another car to copy the Gear values from, please note that there is no Undo option, so if you copy the values from another cars, the older ones would be lost.

Choose Tire: Here you can select from pre-created tire settings, this tire settings are located under the Tires Folder in the Resource Folder, please note that there is no Undo option, so if you copy the values from another cars, the older ones would be lost.

Copy a coefficients From: Here you can select another car to copy the a coefficients values from, please note that there is no Undo option, so if you copy the values from another cars, the older ones would be lost.

Copy b coefficients From: Here you can select another car to copy the b coefficients values from, please note that there is no Undo option, so if you copy the values from another cars, the older ones would be lost.

Copy c coefficients From: Here you can select another car to copy the c coefficients values from, please note that there is no Undo option, so if you copy the values from another cars, the older ones would be lost.

Debug?: Enable this to see the forces of the tire.

Wear?: This option is for enabling disabling the tire wear when using the physics alone.

Wheel Radius: Set the radius of the wheel to match the radius of the wheel mesh, be aware that this is set automatically by the car import wizard, normally this should not be set manually.

Tire Inertia (kg*m^2): Set the Tire Angular Inertia.

Coefficient of grip: This is simply multiplied to the resulting forces, an easy way to quickly change handling characteristics, increase this number in order to get more Grip (higher would act as arcade physics).

Slip ratio max: This value would be used to clamp the slip ratio of the tire

Slip angle max: This value would be used to clamp the slip angle of the tire

Tire FX Factor: This value would be used to multiply by the resulting longitudinal force

Tire FY Factor: This value would be used to multiply by the resulting lateral force

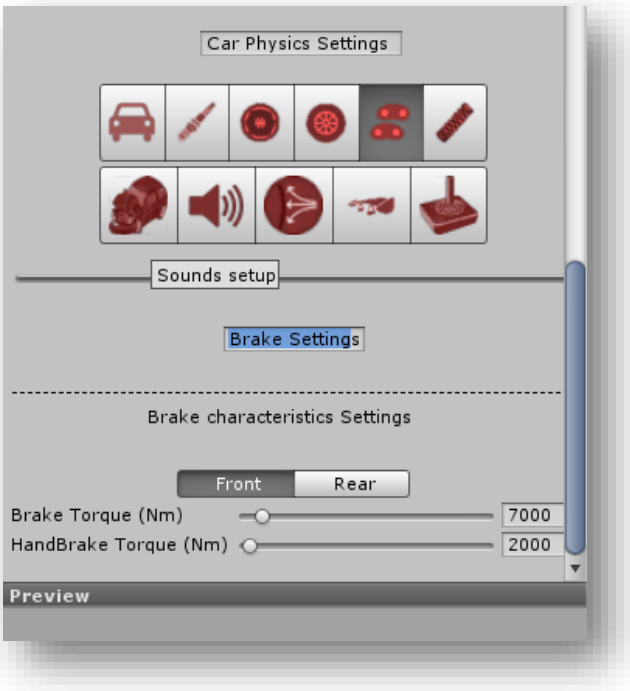
Static Friction Coefficient: This is the static coefficient of grip when the car either at full stop or at low speed to avoid side slipping.

Friction torque (Nm): This value would act in opposite direction to the wheel spin direction, making the wheel to stop, higher values, would make this force higher.

Tire Hardness: This value would make the Tire more Hard or soft, higher values would make the Tire harder.

Pacejka Coefficients: This are the pacejka coefficients used for Lateral and Longitudinal simulation of the tire forces, please get more information about them on books like the physics of racing, by Brian Beckman.

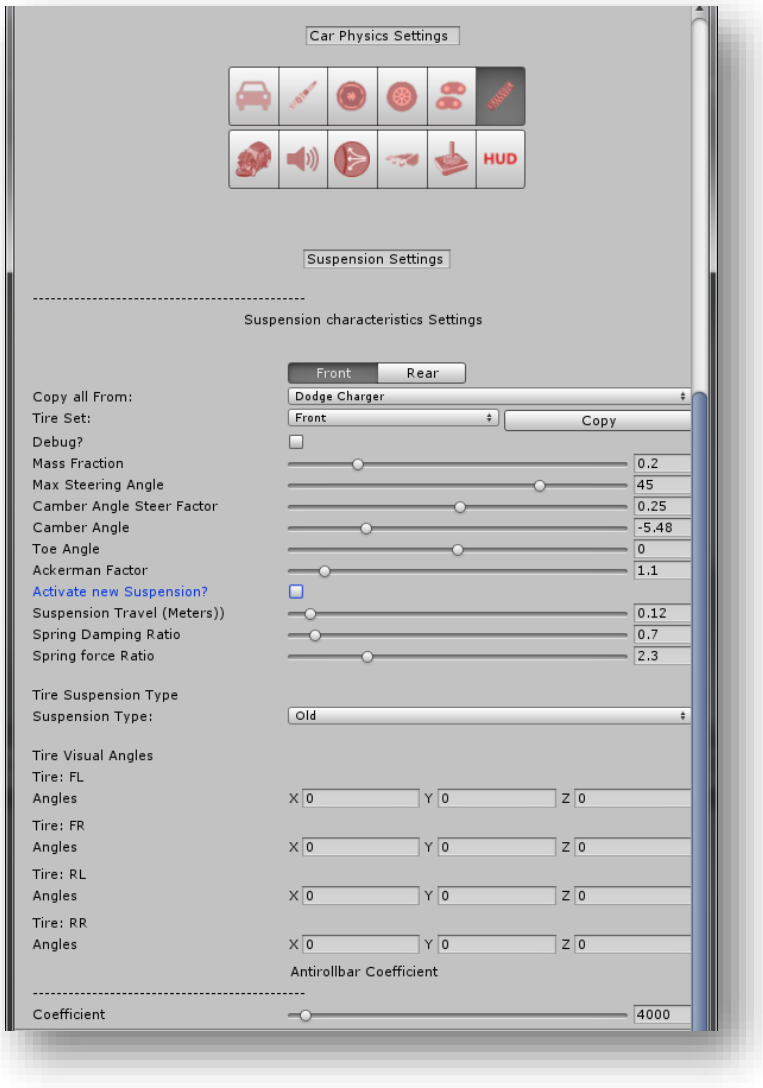
Brake Settings



Brake Torque (Nm): The torque applied to this tire when breaking.

HandBrake Torque (Nm): The torque applied to this tire when Handbreaking.

Suspension Settings



Mass Fraction: This is the % of the mass that each front tire holds of the total car mass.

Max Steering Angle: The max turn angle of the Tires.

Camber Angle Steer factor: How much the Camber is affected when steering.

Camber Angle: The camber angle of the Rear/Front Tires.

Toe Angle: The Toe angle of the Rear/Front Tires, positive values would make the front Tires separate from each other.

Ackerman Factor: The Ackerman Factor for the Rear/Front Tires, positive values would make the inner Tire in a turn, turn more than the outer tire.

Suspension Travel (Meters)): The suspension distance.

Spring Damping Ratio: Damper ratio of the spring force, it would be ratio * spring force.

Spring force Ratio: Ratio of the suspension, it qould be ratio * normal force.

Antiroll bar Coefficient: Antiroll Bar coefficient for the Front/Rear Tires.

New settings:

Activate new Suspension?: This option activates the new suspension spring system

Damper Rebound: The damper force applied on suspension rebound

Damper Bounce: The damper force applied on suspension Bounce

Damper Factor min: The min factor used for the damper

Damper Factor max: The max factor used for the damper

Spring force: The force of the spring

Spring Factor min: The min factor for this spring

Spring Factor max: The max factor for this spring }

Suspension Type: You can select here from 4 different types of suspension:

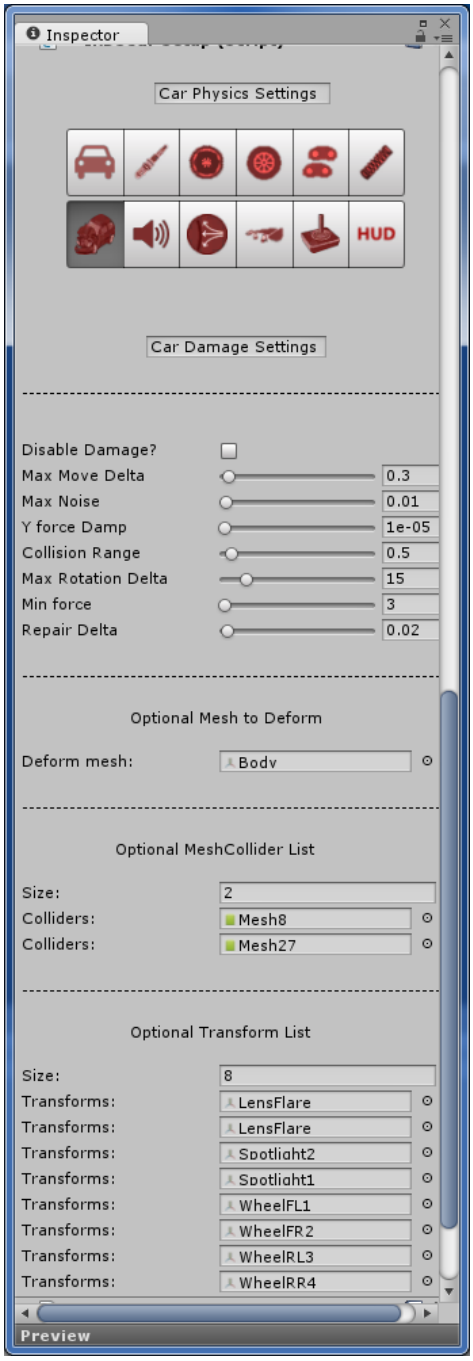
- **Old:** This would make the suspension to be as in the older versions of iRDS.
- **Basic:** This type of suspension would emulate a single simple arm suspension
    - **Hinge anchor pos:** The position of the hinge anchor with respect to the car (this is where you want the arm to start from the car).
    - **Wheel Hub Pos:** The position of the wheel hub with respect to the car (this is where you want the hub of the wheel to be).
- **Macpherson:** This would enable the MacPherson suspension.
    - **Strut Top:** The position with respect to the car of the Strut top.
    - **Strut End:** The position with respect to the car of the Strut End (this is also where the hub of the wheel would be)
    - **Strut Hinge:** The position of the hinge anchor with respect to the car (this is where you want the arm to start from the car).
- **Double Wishbone:** This enables the double wishbone suspension.

- **Upper arm anchor:** The position of the upper hinge anchor with respect to the car (this is where you want the arm to start from the car).
- **Upper arm hub:** The position of the wheel hub (upper) with respect to the car (this is where you want the hub of the wheel to be).
- **Lower arm anchor:** The position of the lower hinge anchor with respect to the car (this is where you want the arm to start from the car).
- **Lower arm hub:** The position of the wheel hub (lower) with respect to the car (this is where you want the hub of the wheel to be).

Tire Visual angles:

These are the angles of the tire for ta visual effect, this could be useful when making Bikes and want to have the tire not to go up and down, but instead in an angle of 10 degrees on the x axis, simulating the inclination that normally the bikes have on the front wheel.

Car Damage Settings



Disable Damage?: Disable car damage.

Repair now?: Repair the vehicle now?.

Max Move Delta: Maximum distance one vertices moves per explosion (in meters).

Max Noise: Maximum noise to be applied when deforming the mesh.

Y force Damp: Damp for the Y axis, to avoid deformation when hitting ground after a jump, lower values would prevent Y axis forces to deform the mesh.

Collision Range: Max distance between vertices and point of contact to be affected by the collision.

Max Rotation Delta: Maximum amount in degrees a Transform affected by collision would be rotated.

Min force: Minimum generated by a collision in order to start deforming the mesh.

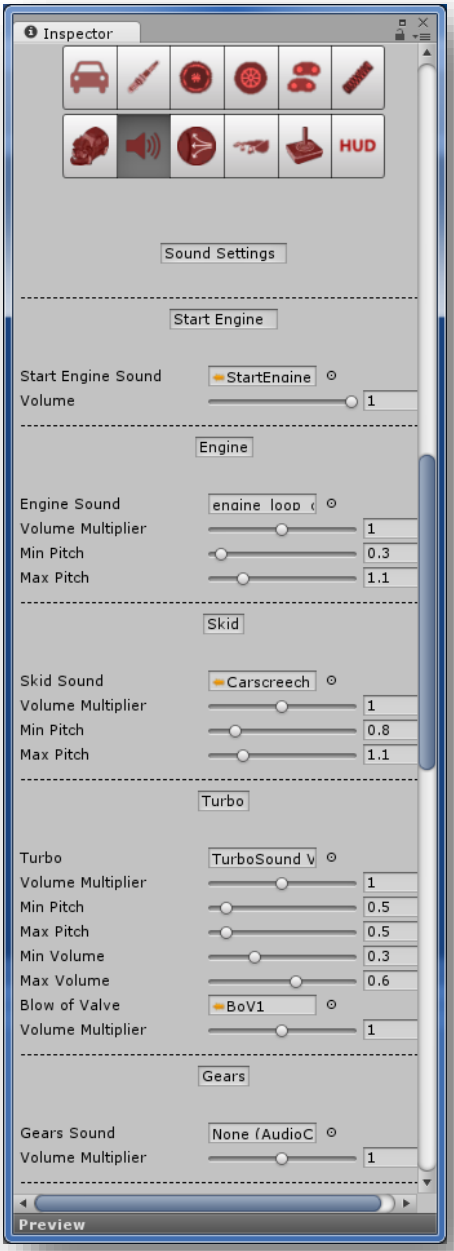
Repair Delta: Percentage of repairing by timestep.

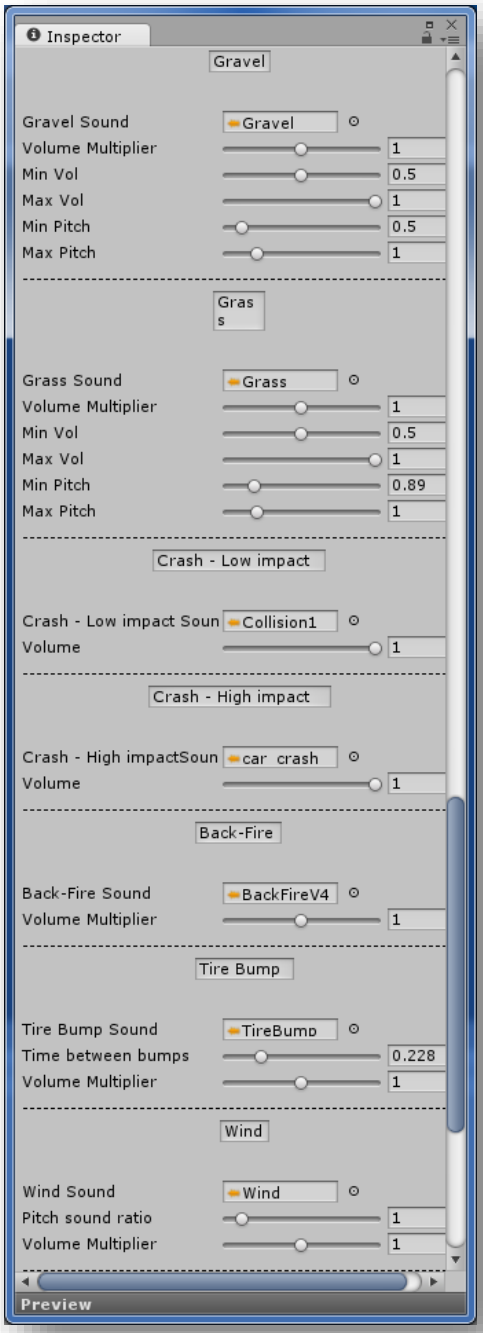
optionalMeshList: Add here all the Meshes you want to be deformed on collisions, usually put here everything except the Tires of the car.

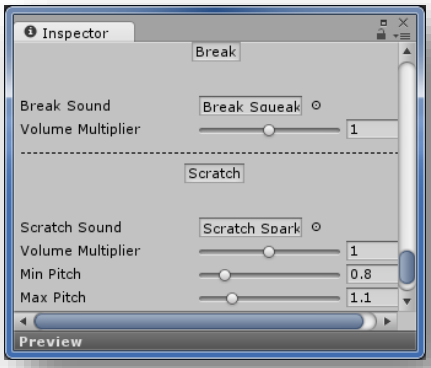
optionalColliderList: Add here all the colliders you want to be used to retrieve collision detections when hit.

optionalTransformList: Add here al the transforms you want to be moved and rotated due to the collision impact and force.

Sound Settings




Inspector
Gravel
Gravel Sound
Gravel
Volume Multiplier
1
Min Vol
0.5
Max Vol
1
Min Pitch
0.5
Max Pitch
1
Gras
s
Grass Sound
Grass
Volume Multiplier
1
Min Vol
0.5
Max Vol
1
Min Pitch
0.89
Max Pitch
1
Crash - Low impact
Crash - Low impact Soun
Collision1
Volume
1
Crash - High impact
Crash - High impactSoun
car_crash
Volume
1
Back-Fire
Back-Fire Sound
BackFireV4
Volume Multiplier
1
Tire Bump
Tire Bump Sound
TireBump
Time between bumps
0.228
Volume Multiplier
1
Wind
Wind Sound
Wind
Pitch sound ratio
1
Volume Multiplier
1
Preview



Start Engine Sound: Assign here the start engine sound.

Volume: Start engine Sound volume.

Engine Sound: Assign here the engine sound.

Volume Multiplier: Sound volume Multiplier.

Min Pitch: Sound min pitch.

Max Pitch: Sound max pitch.

Skid Sound: Assign here the skid sound.

Volume Multiplier: Sound volume Multiplier.

Min Pitch: Sound min pitch.

Max Pitch: Sound max pitch.

Turbo Sound: Assign here the turbo sound.

Volume Multiplier: Sound volume Multiplier.

Min Pitch: Sound min pitch.

Max Pitch: Sound max pitch.

Min Volume: Sound min Volume.

Max Volume: Sound max Volume.

Blow off Valve Sound: Assign here the blow off valve sound.

Volume Multiplier: Sound volume Multiplier.

Gears Sound: Assign here the gear sound.

Volume Multiplier: Sound volume Multiplier.

Gravel Sound: Assign here the gravel sound.

Volume Multiplier: Sound volume Multiplier.

Min Pitch: Sound min pitch.

Max Pitch: Sound max pitch.

Min Volume: Sound min Volume.

Max Volume: Sound max Volume.

Grass Sound: Assign here the Grass sound.

Volume Multiplier: Sound volume Multiplier.

Min Pitch: Sound min pitch.

Max Pitch: Sound max pitch.

Min Volume: Sound min Volume.

Max Volume: Sound max Volume.

Crash - Low impact Sound: Assign here the crash low impact sound.

Volume: Sound volume.

Crash - High impact Sound: Assign here the Crash high impact sound.

Volume: Sound volume.

Back-Fire Sound: Assign here the Back fire sound.

Volume Multiplier: Sound volume Multiplier.

Tire Bump Sound: Assign here the Tire Bump sound.

Time between bumps: Time between each bump on a tire.

Volume Multiplier: Sound volume Multiplier.

Wind Sound: Assign here the Wind sound.

Pitch sound ratio: lower values would make the pitch lower.

Volume Multiplier: Sound volume Multiplier.

Brake Sound: Assign here the Brake sound.

Volume Multiplier: Sound volume Multiplier.

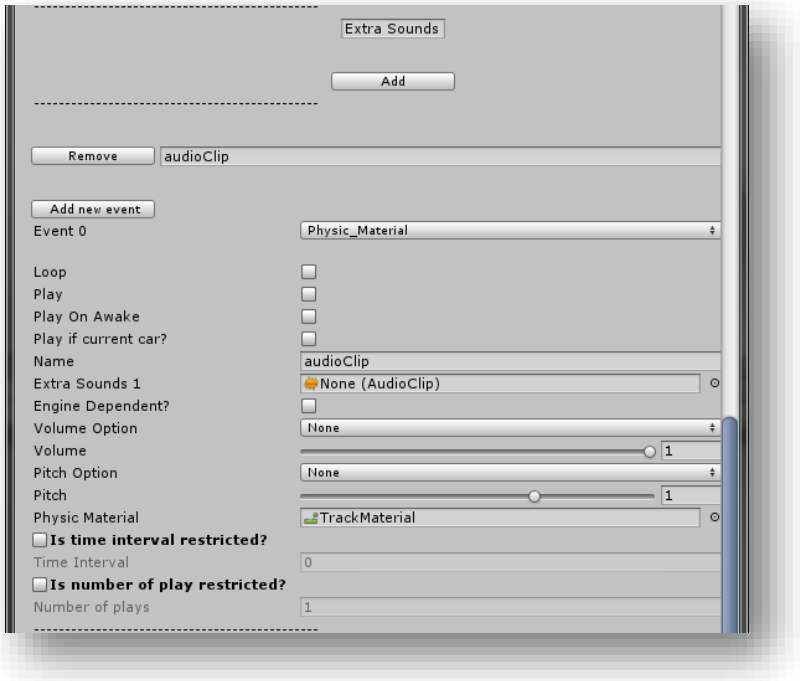
Scratch Sound: Assign here the Scratch sound.

Volume Multiplier: Sound volume Multiplier.

Min Pitch: Sound min pitch.

Max Pitch: Sound max pitch.

Extra Sounds



Add: Add new extra sound

Remove: Remove this extra sound

Add new event: Add a new event to this extra sound

Event: The event that would activate this sound, the types of events are:

- None: No event selected.
- Overtake: Would be activated when the AI is overtaking
- OvertakeLeft: Would be activated when the AI is overtaking on the left side
- OvertakeRight: Would be activated when the AI is overtaking on the right side
- Braking: Would be activated when braking
- Avoiding: Would be activated when the AI is avoiding
- AvoidingLeft: Would be activated when the AI is avoiding on the left side
- AvoidingRight: Would be activated when the AI is avoiding on the right side
- Backfire: Would be activated when the back fire is activated
- Scratch: Would be activated when the car is scratching something
- TurboBlowoff: Would be activated when the turbo blow off is activated
- StartEngine: Would be activated when the AI engine is starting
- Physic_Material: Would be activated when the tires are in contact with the physics material
- Crash: Would be activated when the car crashes

- Shifting: Would be activated when the car is shifting gears
- WinningRace: Would be activated when the car won the race
- LossingRace: Would be activated when the car lose the race
- PlaceOnRace: Would be activated when by the place the car ended on the race
- Nitro: Would be activated when the nitro is activated

Loop: Is this a loop clip?

Play: Should this clip be always played?

Play On awake: Play this on awake?

Play if current car: Only plays this sound if it is the current target of the camera?

Name: The name of this sound

Extra sounds: The actual audio clip for this sound

Engine dependent: Is this an engine dependent sound, i.e. if the engine is turned off, this sound should not be active?

Volume option:

- RPM dependent: The volume would depend on the RPM
    - Volume multiplier: The multiplier of this volume sound
    - Min Volume: The minimum volume to use at lower rpm
    - Max Volume: The maximum volume to use at higher rpm
- Speed dependent:
    - Volume multiplier: The multiplier of this volume sound
    - Min Volume: The minimum volume to use at lower speed
    - Max Volume: The maximum volume to use at higher speed
    - Min Speed: The minimum speed to use the min volume
    - Max Speed: The maximum speed to use the max volume
- Throttle Dependent:
    - Volume multiplier: The multiplier of this volume sound
    - Min Volume: The minimum volume to use at lower throttle
    - Max Volume: The maximum volume to use at higher throttle

Volume: The volume of this sound

Pitch Option:

- RPM dependent:
    - Min Pitch: The minimum Pitch to use
    - Max Pitch: The maximum pitch to use
- Speed dependent:

    - Min Pitch: The minimum Pitch to use
    - Max Pitch: The maximum pitch to use
    - Min Speed: The minimum speed to use the min pitch
    - Max Speed: The maximum speed to use the max pitch
- Throttle Dependent:
    - Min Pitch: The minimum Pitch to use
    - Max Pitch: The maximum pitch to use
- Randomized:
    - Min Pitch: The minimum Pitch to use
    - Max Pitch: The maximum pitch to use

Pitch: The pitch of this sound

Physics Material: The physic material assigned for the event “Physics material” only

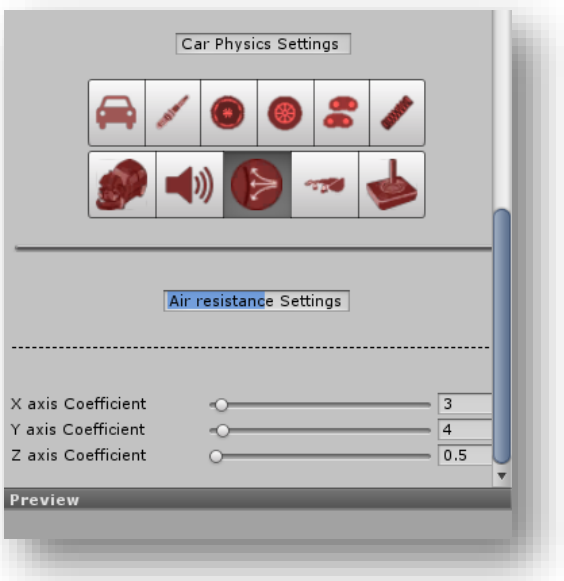
Is time interval restricted?: Is this sound restricted by an amount of time to be played per event?

Time interval: Time to play this sound per event

Is Number of play restricted?: Is this sound restricted to be played a certain amount per event?

Number of plays: The number of plays this sound would be used per event

Air Resistance Settings

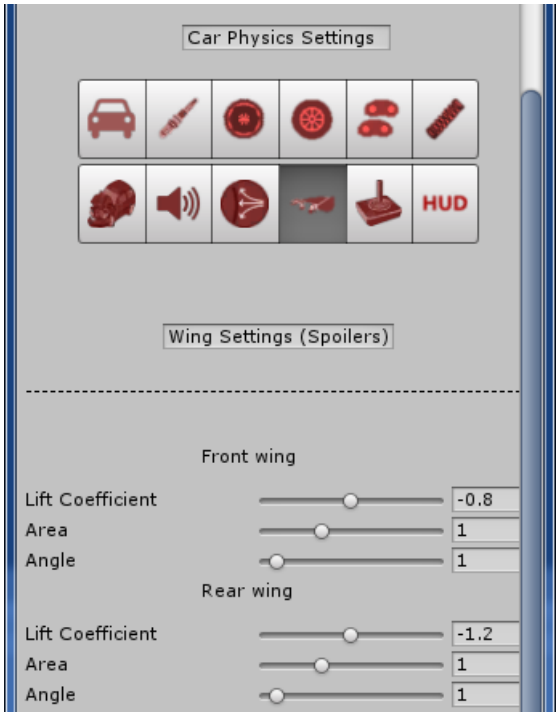


X axis Coefficient: Air resistance on the X axis, that is right and left.

Y axis Coefficient: Air resistance on the Y axis, that is up and Down.

Z axis Coefficient: Air resistance on the Z axis, that is front and back.

Wing Settings

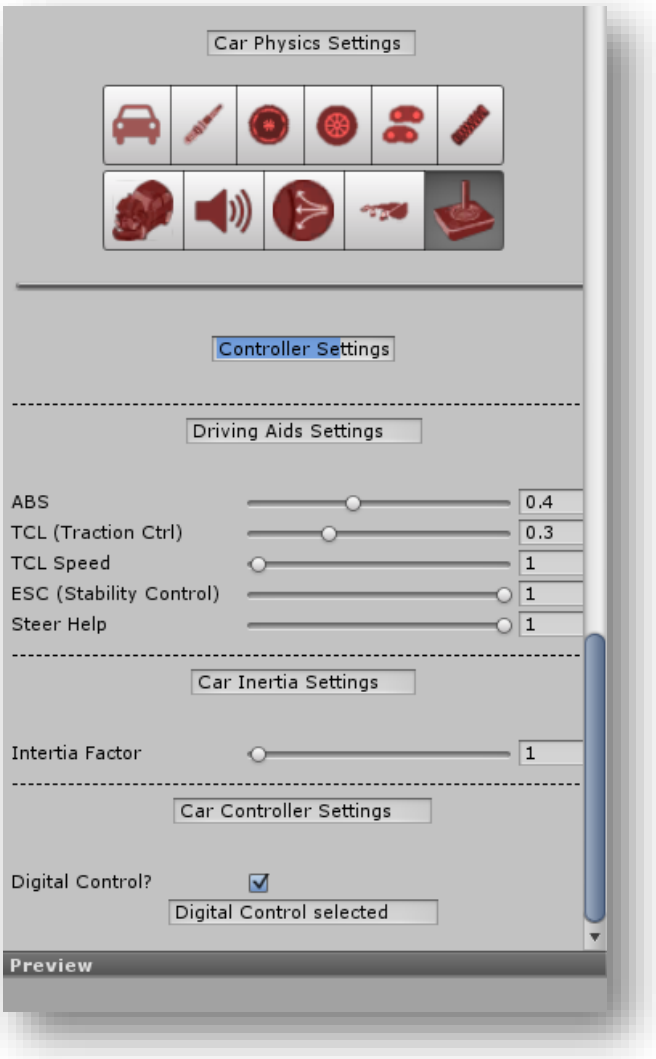


Lift Coefficient: The lift coefficient for the Rear / Front wing.

Area: The area of the wing.

Angle: Angle of the wing.

Controller Settings



Driving Aids Settings

ABS: The amount of ABS used for assisting on braking, the higher the value, the less ABS would avoid the tires from locking while braking.

TCL (Traction Ctrl): The higher this value, the less would interact the Traction control to avoid tire from over spinning.

TCL Speed: The minimum speed for the Traction control to be active.

ESC (Electronic Stability Control): The Stability control would try to keep the car heading to the current steering heading, by breaking, if necessary, a tire in order to keep control of the car.

Steer Help: This value would limit the max steering angle if the car starts loosing grip to Aid on recovering the car control. Lower values greater than 0 makes the steer help more responsive and higher values would make the steer help to help less.

Car Inertia Settings

Inertia Factor: This factor would be multiplied by the Inertia Tensor calculated by the Physx engine.

Car Controller Settings

Digital Control?: Select this option if you want to control the car using the Keyboard, deselect it to use analog input.

Control Input Settings

Settings

Throttle Time: The time step for the throttle

Throttle Time Traction: The time step for the throttle to get traction

Throttle Release Time: The time step for the throttle release

Throttle Release Time Traction: The time step for the throttle release traction

Steer Time: The time step for the steering

Velo Steer Time: The time step for the steering at higher speeds

Steer Release Time: The time step for the steer to release

Velo Steer Release Time: The time step for the steer to release at higher speeds

Steer Correction Factor: The correction factor when changing the steer

Clutch Time: The time step for the clutch

Clutch Release Time: The time step for the clutch to release

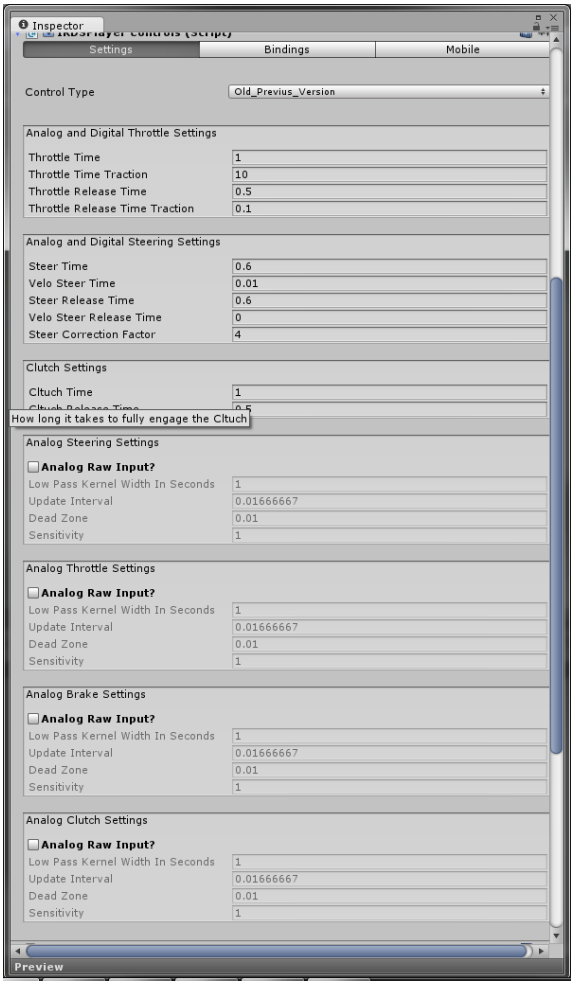
Analog Raw Input?: Use Analog Raw input when in analog mode?

Low Pass Kernel: This is the low pass time that would be used for the low pass filter calculations

Update Interval: The time step for the low pass filter

Dead Zone: The dead zone of the input

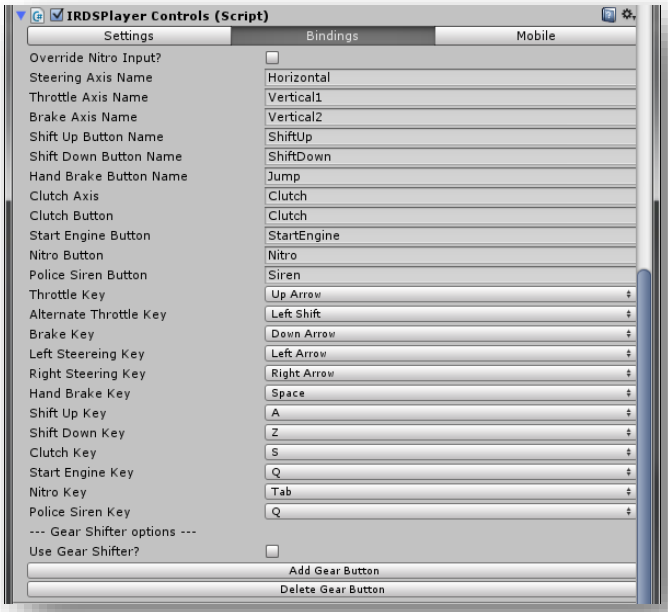
Sensitivity: The sensitivity of the input



Input Bindings

Override Nitro Input?: (for standalone physics use only, won't take effect if used with the IRDSManager and levelload) This would override the nitro input, which is useful to create games were the nitro is activated when the car gets a power up.

Inputs:



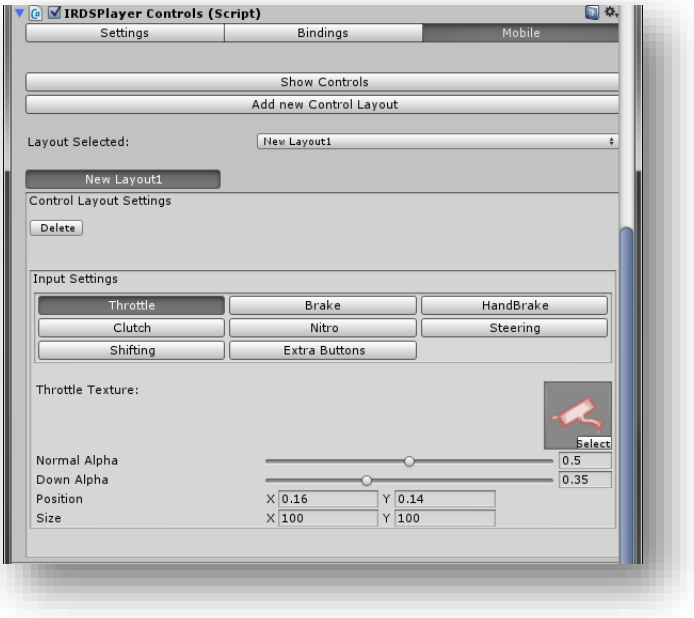
- Steering Axis Name
- Throttle Axis Name
- Brake Axis Name
- Shift up Button Name
- Shift down Button Name
- Handbrake Button Name
- Clutch Axis
- Clutch Button
- Police Siren Button
- Throttle Key
- Alternate Throttle Key
- Brake Key
- Left Steering Key
- Right Steering Key
- Handbrake Key
- Shift up Key
- Shift down Key
- Clutch Key
- Start Engine Key
- Nitro Key
- Police Siren Key

Use Gear Shifter?: Activate the gear shifter on this car?

Add Gear Button: Add a new Gear shifter button

Delete Gear Button: Delete the last gear shifter button

Mobile Inputs



Show controls: Show the controls on the Scene view, for previewing the position of the Mobile controls

Add new control layout: Create a new mobile control layout

Layout Selected: The current layout selected

Delete: Delete the current layout

Normal Alpha: The alpha value when the input is not pressed down

Down Alpha: The alpha value when the input is pressed down

Position: The position of this input

Size: The size of this input

Steering

Accelerometer settings:

Low Pass Kernel: This is the low pass time that would be used for the low pass filter calculations

Update Interval: The time step for the low pass filter

Dead Zone: The dead zone of the input

Sensitivity: The sensitivity of the input

Touch Steering:

Sector Position: The position of the steering sector or area where the input is grab from the screen

Sector Size: The size of the touch sector

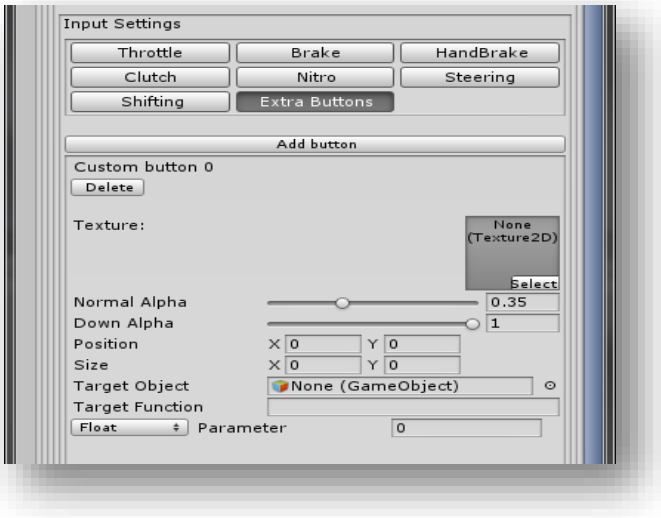
Test Sector?: If this is enabled, the sector is shown in editor mode, on the scene view

Angle: The max angle the texture rotates

Length: The length from the sides is the input grab from screen

Use accelerometer settings?: Use the input settings of the accelerometer (update interval, Dead zone, etc.)

Extra Buttons:



Target Object: The object this button would send the message to

Target Function: The function name that would be called on the target object

Parameter: The parameter is going to be passed to the target function

Force FeedBack

You can enable force feedback for the full version car physics by using any of the following third parties force feedback packages:

- Force Feedback Toolkit
- Logitech Gaming SDK

To use any of those two force feedback packages you need to first import one of them and the go to the scripts folder inside IRDS folder and open up the Force Feedback folder, there is a file named FFB.rar, open it and extract the script that corresponds to the Force Feedback you choose to use.

For using Logitech SDK, the script is called “IRDSLogitechFFB.cs” and for the FFB Toolkit the script’s name is “IRDSFFBInput.cs”. You can modify those scripts as you wish.

When you have extracted the correct script, now you need to add this script to every car you rigged before and disable on all cars the script called IRDSPlayerControl. You can also on the Force Feedback script create the method for controlling the car if the user selects digital input.

Now you have force feedback enabled, and don’t forget to uncheck the digital controls on the LevelLoad Object to enable analog inputs.

Step by Step user Guide – How to use the AI System from scratch

Introduction

This guide is intended to be a step by step guide for installing and implementing this System to your project and includes all the important things you should know and things you must take into account to make the System work properly.

Step by step Guide

Step 1 - Importing the package to your project

First thing you should do is to import the iRDS package to your project, to do this you should open the asset store window, and go to your local bought/downloaded assets list, then import iRDS.

Also, you should be aware that for being able to use iRDS with Edy's Vehicle Physics or UnityCar you need to import first the car physics you want to use and then go to Other Physics Package folder and decompress the scripts for the car physics you want to use, you can decompress the scripts on the scripts folder under the iRDS folder (assets/iRDS/Scripts).

Step 2 – Getting started to setup the AI System

Now you have imported the right package for you, let's get started on setting up the system.

Step 3 – Rig some cars

If you selected the package with its own car physics (continue to next step if you use Edys version), then First you need to add your cars, so the first step would be to rig some cars (see rig car section on this manual).

When you are done rigging the cars, you need to setup some parts of it out of the rigging wizard, you have to setup sounds, suspension, engine, tire, and other stuffs of the car, see Setup Car sections for this.

For Edys package and UnityCar, you need to select the Edys/UnityCar already rigged cars (they could be on the scene or as prefabs) and click on Vehicle → Edy's/UnityCar → Add AI Scripts, and the System would add the scripts to the cars and a game object called Camera Views (you need to

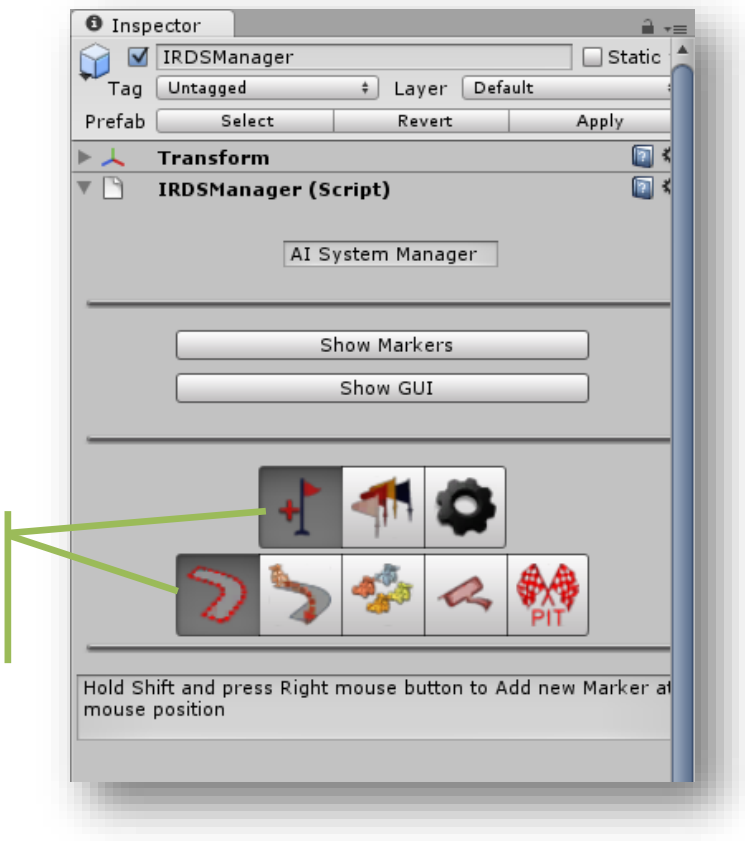
move the objects inside it to be where you want them as desired view, this applies also for the car with the System own car physics)

Step 4 – Add AI System to your track scenes

Now you should add the AI Manager to your scenes that have race tracks, for each of those scene you need to add one AI Manager. Use the Add AI System to current scene option in the iRDS menu under AI System sub menu. See Add AI System section for more details.

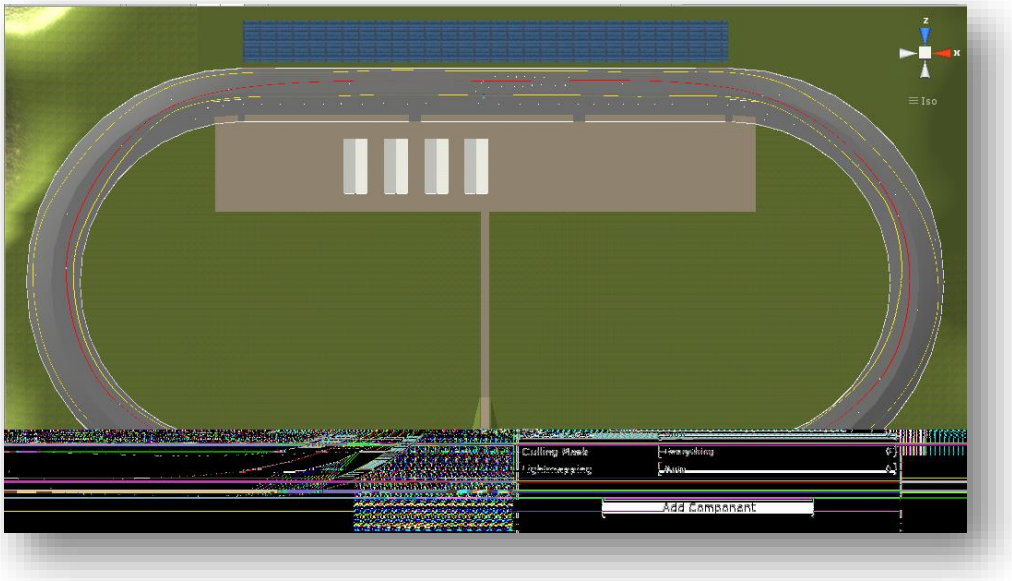
Step 5 – Add track limit points to the track

Now that you have added the AI Manager to your track scene, you need to setup the track limits or the "Virtual track" for the AI Drivers. Right now you should choose the following options from the AI Manager Inspector view:



Selected options are Grey

All that you need to do is start adding the points to the middle of your road and so on until you complete the track, at the end it should look something like this:

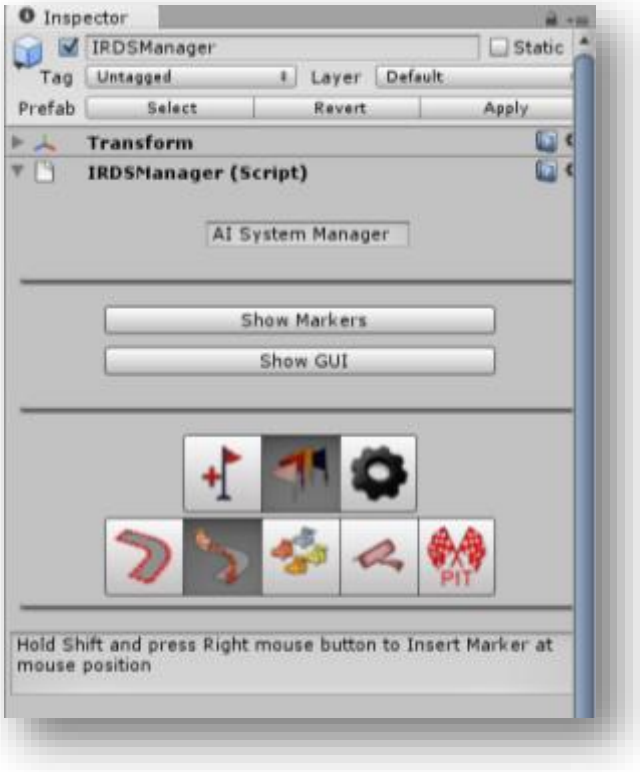


Without the red line, since that is the racing line that we would add later.

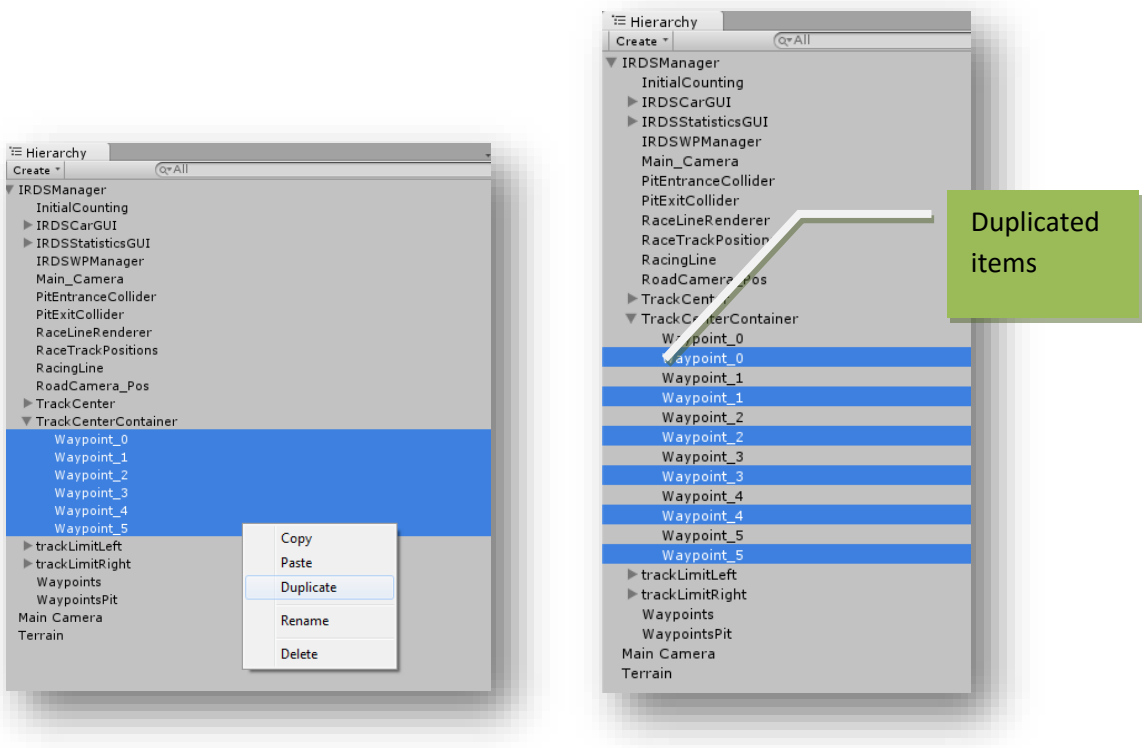
Step 6 – Add track racing line points

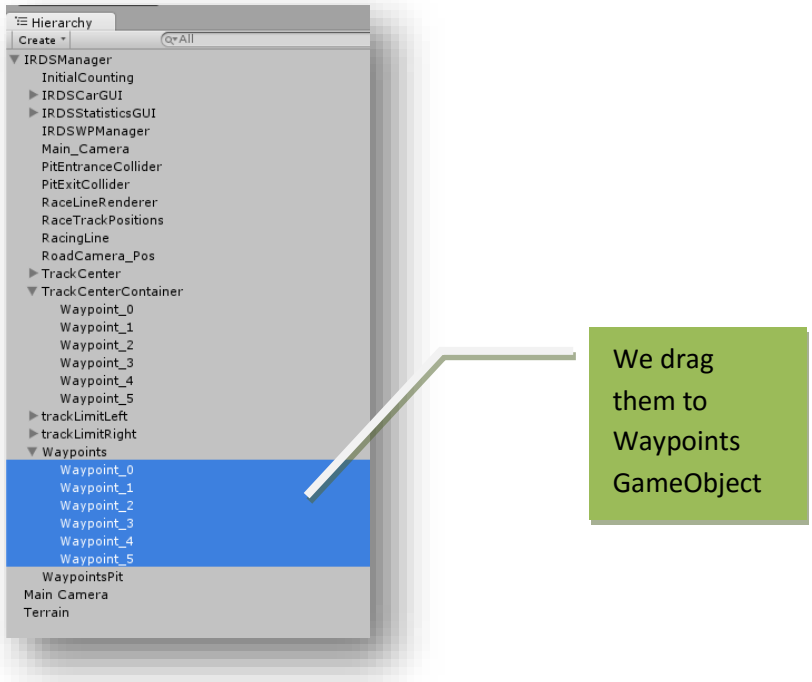
Ok, its time for you to setup the racing line for the AI Drivers to follow on.

There are two ways to do this, one is follow the steps as we did before to create the track limits, adding one by one the points, holding shift and right clicking on the screen with the mouse cursor. With the following options selected:



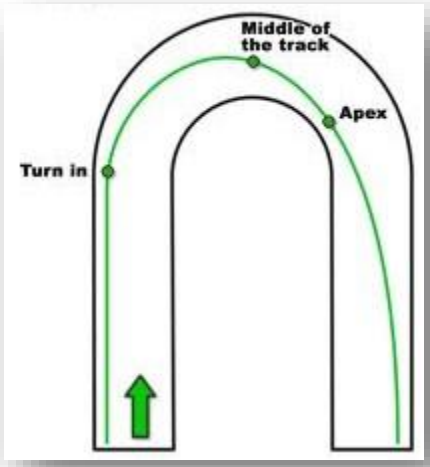
Another way to do this is to select all the waypoints added to the “TrackCenterContainer” game object (they are as children of this game object) and duplicate them, then drag them to the waypoints gam object to make them his children. See pictures below to illustrate:

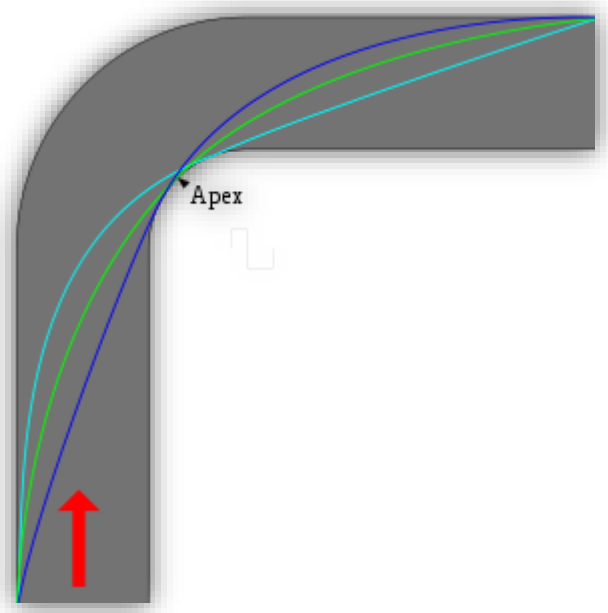




Then you just need to select those waypoints and start moving them to the desired position on the track until you have a race line that when entering the curve it is more to the opposite side of the

turn and on the middle of the curve is on the apex or the inner part of the curve, and on the exit of the curve it should be again more to the outer part of the curve. See pictures below to illustrate:



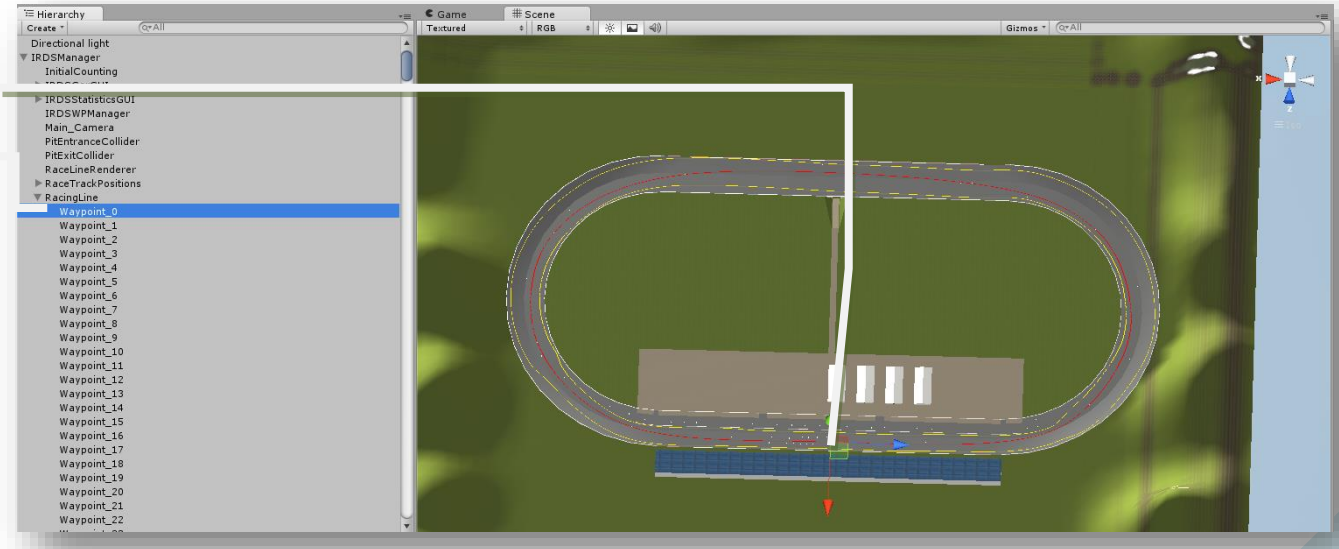


At the end it should look something like this:

Step 7 – Add Grid Positions points

Let's add the grid points, this would be used by the AI system to instantiate the race cars on the track at the starting line, so the first thing to do is to see where the starting line is! And the easiest way to do this is to look at the first waypoint on the racing line, see picture below to illustrate:

Waypoint 0 on RacingLine

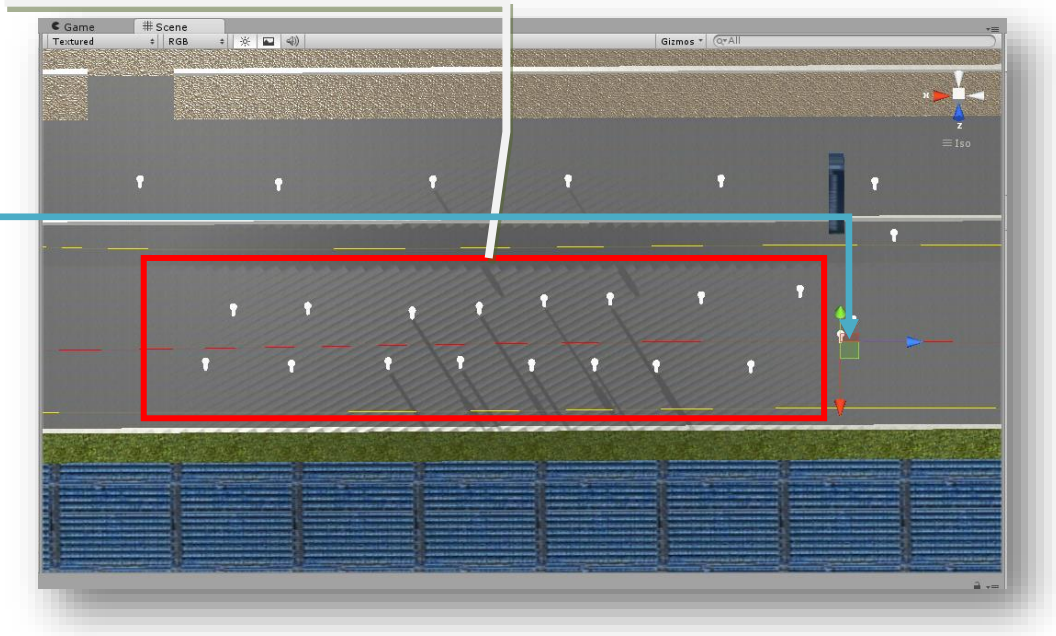


And that's the starting point of the track, and the local Z axis shows the direction of the track, so all the Grid positions should be positioned behind that starting point. We add those points by using the following options on the AI Manager (greyed options):

Add the points by holding down shift key and right clicking the mouse on the screen.

At the end you should have something like this:

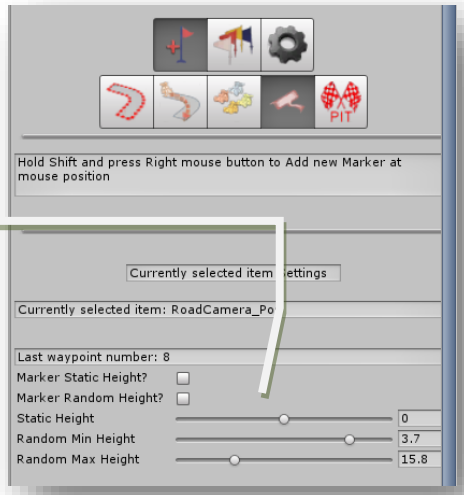
Grid positions are before the first waypoint of the racing line



Step 8 – Add Road Camera points

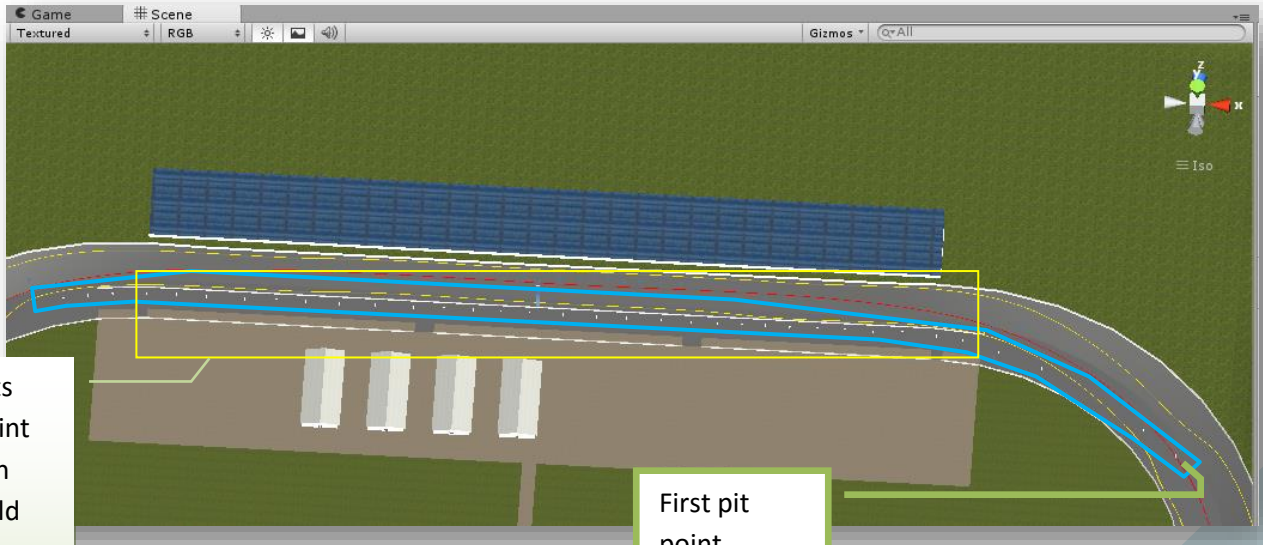
This process is very simple, just need to select the following options and add the points for the cameras on or near the track, remember to check the available options for setting this points height:

Options, make sure they are set up as you prefer and you should select one of the two Height options, Static or Random.



Step 9 – Add pit stop points

This is a very important part of the setup of a track (if the pits are active), this is because the AI System verifies the amount of pits available and compares it with the amount of cars that are going to race, this is to make sure that all the cars have their own pits stop, see the following screenshot to illustrate it better:



These are the pits stop, the first point (point number, in this case 5) should be set up on the "First pit number" and the last one (in this case 30) should be set up on the "Last pit number"

First pit point

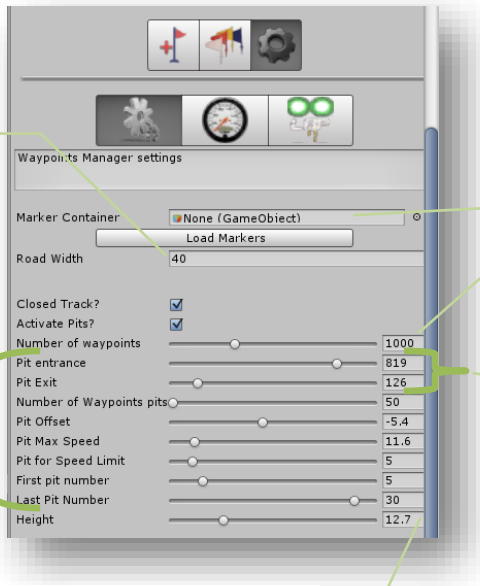
On this track there are plenty of pits stops points, so there is a lot of pit stop for many cars, also notice that the first pit stop point is exactly in the redline (racing line) so that way when the AI enters the pits, the transition between following the racing line and start following the pits line is very smooth.

Also the points that are inside the pits have to be put with some distance between them, because each one of them would become an stopping point for a racer car (normally about 10 meters or more is fine).

Then you should go to the general settings on the AI manager and setup the variables for the pits, see picture bellow:

Remember to set up the road width

Put here the Game Object that contains all the markers of the road if you are using easy roads, this would create a very close track limit to the actual road.



Number of waypoints for the racing line

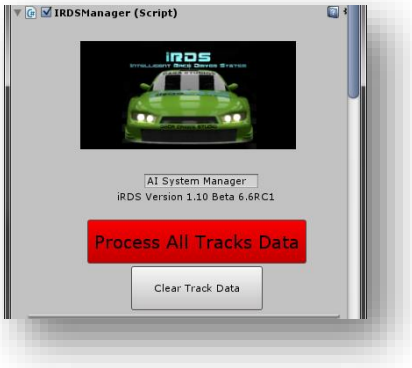
Pits settings

These values are automatically set when you add all your pits points, they should match the nearest point of the racing line for the pit entrance and exit, normally for entrance is better to have it set a few waypoints farther than the nearest point for better transition between racing line and pit line

This values should allways be larger than 0, it is used by the AI system to raycast from that distance above the surface to place then exactly over the surface all the waypoints (track limit left and right, racing line, and the pits line. This should be adjusted in case you have a track with overlapping roads with bridges or something similar to avoid the track that passes under the bridge having the waypoints over the bridge if this height values is set to high.

Step 9.1 –Process the track data

Now you just need to go to IRDSManager and press the button “Process track data”, also there is a new button called clear track data, which is used to clear/delete all the scene track data and reset the IRDSManager.



Step 10 – Setup the HUD’s

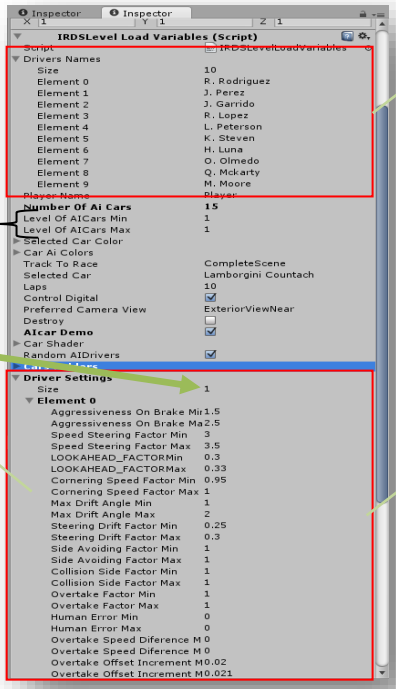
In this step, you should setup all the HUD’s (gears, odometer, tachometer, fuel gauge, turbo gauge, tire gui, laps, position, initial counter, lap time, statistics, etc.) See the Car HUD and General HUD settings for more detail.

Step 11 – Setup the AI Drivers

Now you should set up your AI Drivers, setup the AI Drivers on the Level settings, by felling up the following 2 variables and checking the Random AIDrivers bool variable to be true:

Here you can select the minimum AI level and the Maximum; these values should not exceed the number of the size of the Driver Settings Array

You can add as many AI Drivers settings as you want, using them as different levels, for example easy, moderate and difficult



These are the names that would be used by the AI System, to name the AI Drivers on the race, the number of names is equal to the maximum amount of AI Drivers.

Here you can set up the min and max values for each of the AI Drivers settings, allowing the System to randomly choose those values

See Level Settings Section for more information.

Step 12 – Adding Physic Material to the Track and other colliders

Ok, now you need to add Physics Materials to all the colliders the car would be on, like the road, grass, sand, etc. For this you could use the included physics material (TrackMaterial or Grass) or create your own, then if the material is for the road, you need to setup the dynamic friction and static to 1, that way the System could tell it is a road and also use the skidmark effect and smoke.

If you use a Grass Material then you should lower the dynamic friction to some value less than 1, for example 0.7, this would affect the grip of the tires of all the vehicles passing on that collider.

Another important thing is how you name your material, if you name your material Grass, then the tires would also throw some grass when you drive on that collider, if you name it Sand, then it throws sand, and if you name it GrassSand it would throw both.

Step 13 – Testing your track

For testing your track you need to include the Level Settings on the scene using the iRDS Menu, clicking on AI System → Add Level Settings to current scene, remember that you also could use just one level settings in your game, using it on a scene for letting the player to choose his car, track, car color and anything included on the Level Settings Game Object script and not toggling the Destroy option and when you load any track scene, the game object with all the settings won't be destroyed and its values would be used by the AI Manager included on the loaded scene.

Also if you want to see only the AI in action you can check the AI Car Demo on the level settings, this would make the AI control all the cars on the race.

Capture the flag notes

For Capture the flag to work with obstacle avoidance, it is needed to create a new layer (could be named flag) and assign this new layer to the flag game object, then in the physics manager settings, set this new layer to only be enabled with the default layer and no other layer (or the layer that the cars have assigned).

Obstacle avoidance notes

For using obstacle avoidance, you need to make the main game object of the Cars (only the main object, not its children) to be on a layer named "obstaclesensor" and for the obstacles create a layer called obstacle, then in the physics manager of unity setup only this to layer to collide with each other, if capture the flag is going to be used, make sure that the flag layer does not collides with any of this two layers (obstacle and obstaclesensor). Also you need to change the obstacle Tag to "obstacle" in order for the AI to detect the obstacles.

Helper Methods

Here are some methods you should know about, they would help you on achieving the following:

- Knowing the player's position on the race (also would be necessary to know in which place ended the race).
- Knowing if the player ended the race.
- Knowing the total time of the player for this race.
- Set input controls for implementing your own car control, this could be necessary for mobile devices implementation.

The methods are:

Class IRDSStatistics

This class is responsible for handling the leaders board information and the HUD, here are the Methods available on this class so you could get information about the player, and the other opponents.

IRDSStatistics.getCurrentCarPosition: This is a non-static method, so the best way to retrieve this is by using GameObject.FindObjectOfType(typeOf(IRDSStatistics)) and then accessing the method, this method returns the position as an int.

IRDSStatistics.getCurrentCarTotalTime: This is a non-static method, so the best way to retrieve this is by using GameObject.FindObjectOfType(typeOf(IRDSStatistics)) and then accessing the method, this method returns the TotalTime as a float.

IRDSStatistics.getCurrentCarEndRace: This is a non-static method, so the best way to retrieve this is by using GameObject.FindObjectOfType(typeOf(IRDSStatistics)) and then accessing the method, this method returns true if the player ended the race or false if he hasn't finished yet as a bool.

IRDSStatistics.GetAllDrivers(): This is a non-static method, so the best way to retrieve this is by using GameObject.FindObjectOfType(typeOf(IRDSStatistics)) and then accessing the method, this method returns all the IRDSCarControllerAI on the Scene, so it returns an array of type IRDSCarControllerAI.

IRDSStatistics.GetCarCamera(): This is a non-static method, so the best way to retrieve this is by using GameObject.FindObjectOfType(typeOf(IRDSStatistics)) and then accessing the method, this method returns the Main Camera Script of type IRDSCarCamera.

*****These are New Methods*****

IRDSStatistics.GetCanRace(): This is a static method, so you have to retrieve this is by using IRDSStatistics.GetCanRace(), this method returns false if the race have not started yet and true if the race have started.

IRDSStatistics.SetCanRace(bool flag): This is a static method, so you have to retrieve this is by using IRDSStatistics.SetCanRace(bool flag), this method sets if the AI can race or not, or if the race has started or not (true = race started, false = race not started). You can use this method if you don't want to use the IRDSCarCamera script and want to use your own.

IRDSStatistics.GetCurrentCar() This is a static method, so you have to retrieve this is by using IRDSStatistics.GetCurrentCar, this method returns the current car that is showing currently on the standings, it returns the instance of the class IRDSCarControllerAI.

IRDSStatistics.SetCurrentCar(IRDSCarControllerAI car) This is a static method, so you have to retrieve this is by using IRDSStatistics.SetCurrentCar, this method sets the car that would be showing on the standing as the current car.

IRDSStatistics.GetTotalLaps(): This is a static method, so you have to retrieve this is by using IRDSStatistics.GetTotalLaps(), this method gets the total laps of the race, it returns an int.

IRDSStatistics.GetPlayerStatsActive(): This is a non-static method, so the best way to retrieve this is by using GameObject.FindObjectOfType(typeOf(IRDSStatistics)) and then accessing the method, this method returns true if standings are active or false if not, the value returned is a bool.

IRDSStatistics.GetPositionActive(): This is a non-static method, so the best way to retrieve this is by using GameObject.FindObjectOfType(typeOf(IRDSStatistics)) and then accessing the method, this method returns true if position HUD is active or false if not, the value returned is a bool.

IRDSStatistics.GetLapActive(): This is a non-static method, so the best way to retrieve this is by using GameObject.FindObjectOfType(typeOf(IRDSStatistics)) and then accessing the method, this method returns true if laps HUD is active or false if not, the value returned is a bool.

IRDSStatistics.GetLapTimesActive(): This is a non-static method, so the best way to retrieve this is by using GameObject.FindObjectOfType(typeOf(IRDSStatistics)) and then accessing the method, this method returns true if lap time HUD is active or false if not, the value returned is a bool.

IRDSStatistics.GetMiniMapsActive(): This is a non-static method, so the best way to retrieve this is by using GameObject.FindObjectOfType(typeOf(IRDSStatistics)) and then accessing the method, this method returns true if minimap HUD is active or false if not, the value returned is a bool.

IRDSStatistics.SetPlayerStatsActive(bool active): This is a non-static method, so the best way to retrieve this is by using GameObject.FindObjectOfType(typeOf(IRDSStatistics)) and then accessing the method, this method sets true or false to activate or deactivate the standings on the HUD.

IRDSStatistics.SetPositionActive(bool active): This is a non-static method, so the best way to retrieve this is by using GameObject.FindObjectOfType(typeOf(IRDSStatistics)) and then accessing the method, this method sets true or false to activate or deactivate the Position on the HUD.

IRDSStatistics.SetLapActive(bool active): This is a non-static method, so the best way to retrieve this is by using GameObject.FindObjectOfType(typeOf(IRDSStatistics)) and then accessing the method, this method sets true or false to activate or deactivate the Laps on the HUD.

IRDSStatistics.SetLapTimesActive(bool active): This is a non-static method, so the best way to retrieve this is by using GameObject.FindObjectOfType(typeOf(IRDSStatistics)) and then accessing the method, this method sets true or false to activate or deactivate the Lap time on the HUD.

IRDSStatistics. SetMiniMapsActive(bool active) This is a non-static method, so the best way to retrieve this is by using GameObject.FindObjectOfType(typeOf(IRDSStatistics)) and then accessing the method, this method sets true or false to activate or deactivate the Mini Map on the HUD.

Class IRDSCarControllInput

This Class is responsible of sending the inputs to the engine script and wheels (steering, brake, hand brake, and throttle) you can use this methods to implements your own code for controlling the human car.

IRDSCarControllInput.setBrakeInput: This is a non-static method, so the best way to retrieve this is by using GetComponent< **IRDSCarControllInput** >() and then accessing the method, this method sets the brake input as a float, from 0 to 1 (it is also clamped).

IRDSCarControllInput.setThrottleInput: This is a non-static method, so the best way to retrieve this is by using GetComponent< **IRDSCarControllInput** >() and then accessing the method, this method sets the Throttle input as a float, from 0 to 1 (it is also clamped).

IRDSCarControllInput.setSteerInput: This is a non-static method, so the best way to retrieve this is by using GetComponent< **IRDSCarControllInput** >() and then accessing the method, this method sets the brake input as a float, from -1 to 1 (it is also clamped).

IRDSCarControllInput.setHandBrakeInput: This is a non-static method, so the best way to retrieve this is by using GetComponent< **IRDSCarControllInput** >() and then accessing the method, this method sets the Handbrake input as a float, from 0 to 1 (it is also clamped).

IRDSCarControllInput.GetCarPilot: This is a non-static method, so the best way to retrieve this is by using GetComponent< **IRDSCarControllInput** >() and then accessing the method, this method gets a bool value, if it set to true, then this car is human controlled, otherwise is AI Controlled.

*****These are new methods*****

IRDSCarControllInput.GetOnGround(): This is a non-static method, so the best way to retrieve this is by using GetComponent< **IRDSCarControllInput** >() and then accessing the method, this method gets a bool value, if it returns true, then this car is touching ground.

IRDSCarControllInput.GetOnTrack() : This is a non-static method, so the best way to retrieve this is by using GetComponent< **IRDSCarControllInput** >() and then accessing the method, this method gets a bool value, if it returns true, then this car is on the track, otherwise it is out of the track.

IRDSCarControllInput.GetTopSpeed(): This is a non-static method, so the best way to retrieve this is by using GetComponent< **IRDSCarControllInput** >() and then accessing the method, this method gets a float value, it returns this car top speed.

IRDSCarControllInput.GetCarSpeed(): This is a non-static method, so the best way to retrieve this is by using GetComponent< **IRDSCarControllInput** >() and then accessing the method, this method gets a float value, it returns the car current speed in meters per second, multiply by 3.6 to get km/h and multiply by 5.6 to get mph.

IRDSCarControllInput.SetCarSpeed(float speed): This is a non-static method, so the best way to retrieve this is by using GetComponent< **IRDSCarControllInput** >() and then accessing the method, this method sets a float value, use it to set the car current speed (THIS WILL NOT AFFECT THE RIGIDBODY SPEED!).

IRDSCarControllInput.GetTCLSlip(): This is a non-static method, so the best way to retrieve this is by using GetComponent< **IRDSCarControllInput** >() and then accessing the method, this method gets a float value, it returns the TCL slip value.

IRDSCarControllInput.SetTCLSlip(float tclSlip) : This is a non-static method, so the best way to retrieve this is by using GetComponent< **IRDSCarControllInput** >() and then accessing the method, this method sets a float value, use it to set the TCLSlip value (0 to 1).

IRDSCarControllInput.GetABSSlip() : This is a non-static method, so the best way to retrieve this is by using GetComponent< **IRDSCarControllInput** >() and then accessing the method, this method gets a float value, it returns the ABSSlip value.

IRDSCarControllInput.SetABSSlip(float absSlip) : This is a non-static method, so the best way to retrieve this is by using GetComponent< **IRDSCarControllInput** >() and then accessing the method, this method sets a float value, use it to set the ABSSlip value.

IRDSCarControllInput.GetTCLMinSPD(): This is a non-static method, so the best way to retrieve this is by using GetComponent< **IRDSCarControllInput** >() and then accessing the method, this method gets a float value, it returns the TCL Min speed value.

IRDSCarControllInput.SetTCLMinSPD(float tclMinSPD): This is a non-static method, so the best way to retrieve this is by using GetComponent< **IRDSCarControllInput** >() and then accessing the method, this method sets a float value, use it to set the TCL Min Speed value. (0 to 100)

IRDSCarControllInput.GetescFactor(): This is a non-static method, so the best way to retrieve this is by using GetComponent< **IRDSCarControllInput** >() and then accessing the method, this method gets a float value, it returns the ESCFactor value (0 if disabled, and > 0 <= 1 ESC is enable.

IRDSCarControllInput.SetescFactor(float escfactor): This is a non-static method, so the best way to retrieve this is by using GetComponent< **IRDSCarControllInput** >() and then accessing the method, this method sets a float value, use it to set the ESC Factor value (0 to 1).

IRDSCarControllInput.GetSteerHelpFactor(): This is a non-static method, so the best way to retrieve this is by using GetComponent< **IRDSCarControllInput** >() and then accessing the method, this method gets a float value, it returns the Steer Help Factor value.

IRDSCarControllInput.SetSteerHelpFactor(float SteerHelpfactor): This is a non-static method, so the best way to retrieve this is by using GetComponent< **IRDSCarControllInput** >() and then accessing the method, this method sets a float value, use it to set the SteerHelpFactor (0 to 1).

IRDSCarControllInput.SetDigitalControl (bool digital): This is a non-static method, so the best way to retrieve this is by using GetComponent< **IRDSCarControllInput** >() and then accessing the method, this method sets a bool value, use it to set true to enable digital control or false for analog control.

IRDSCarControllInput.GetDigitalControl (): This is a non-static method, so the best way to retrieve this is by using GetComponent< **IRDSCarControllInput** >() and then accessing the method, this method gets a bool value, if it returns true, then it has digital control enabled otherwise is analog control enabled.

IRDSCarControllInput.GetSlipVelo(): This is a non-static method, so the best way to retrieve this is by using GetComponent< **IRDSCarControllInput** >() and then accessing the method, this method gets a float value with the current average slip velocity of the tires.

IRDSCarControllInput.GetBrakeInput(): This is a non-static method, so the best way to retrieve this is by using GetComponent< **IRDSCarControllInput** >() and then accessing the method, this method gets a float value, with the current value of the brake input (values between 0 and 1).

IRDSCarControllInput.GetMaxRpms(): This is a non-static method, so the best way to retrieve this is by using GetComponent< **IRDSCarControllInput** >() and then accessing the method, this method gets a float value, it returns the car max rpm of the engine.

IRDSCarControllInput.GetRpms(): This is a non-static method, so the best way to retrieve this is by using GetComponent< **IRDSCarControllInput** >() and then accessing the method, this method gets a float value, it returns the current rpm of the car engine.

IRDSCarControllInput.GetLastShiftTime(): This is a non-static method, so the best way to retrieve this is by using GetComponent< **IRDSCarControllInput** >() and then accessing the method, this method gets a float value, it returns the last shift time value.

IRDSCarControllInput.SetRespawnTimer(float time): This is a non-static method, so the best way to retrieve this is by using GetComponent< **IRDSCarControllInput** >() and then accessing the method, this method sets a float value, the value sets the amount of time to respawn the car when it is flipped.

IRDSCarControllInput.GetDrivetrain(): This is a non-static method, so the best way to retrieve this is by using GetComponent< **IRDSCarControllInput** >() and then accessing the method, this method gets a IRDSDrivetrain value, it is the script with the engine and gear ratios value.

IRDSCarControllInput.GetIRDSWheels(): This is a non-static method, so the best way to retrieve this is by using GetComponent< **IRDSCarControllInput** >() and then accessing the method, this method gets a IRDSWheel[] value, these are all the car wheels.

IRDSCarControllInput.GetGoPits(): This is a non-static method, so the best way to retrieve this is by using GetComponent< **IRDSCarControllInput** >() and then accessing the method, this method gets a bool value, if it return true, then the car is going to the pits, otherwise not.

IRDSCarControllInput.SetGoPits(bool _goPits): This is a non-static method, so the best way to retrieve this is by using GetComponent< **IRDSCarControllInput** >() and then accessing the method, this method sets a bool value, if it set to true, then this car is going to enter the pits.

New Variables exposed on this class

absEnable: This is a non-static variable, so the best way to retrieve this is by using GetComponent< **IRDSCarControllInput** >() and then accessing the variable, this variable is a bool value, if it is set to true, then the ABS are on, otherwise not.

public bool escEnable: This is a non-static variable, so the best way to retrieve this is by using GetComponent< **IRDSCarControllInput** >() and then accessing the variable, this variable is a bool value, if it is set to true, then the ABS are on, otherwise not.

public bool tclEnable: This is a non-static variable, so the best way to retrieve this is by using GetComponent< **IRDSCarControllInput** >() and then accessing the variable, this variable is a bool value, if it is set to true, then the ABS are on, otherwise not.

public bool steerHelpEnable: This is a non-static variable, so the best way to retrieve this is by using GetComponent< **IRDSCarControllInput** >() and then accessing the variable, this variable is a bool value, if it is set to true, then the ABS are on, otherwise not.

public bool absTriggered: This is a non-static variable, so the best way to retrieve this is by using GetComponent< **IRDSCarControllInput** >() and then accessing the variable, this variable is a bool value, if it returns true, then the ABS have been triggered, otherwise not.

public bool escTriggered: This is a non-static variable, so the best way to retrieve this is by using GetComponent< **IRDSCarControllInput** >() and then accessing the variable, this variable is a bool value, if it returns true, then the ESC have been triggered, otherwise not.

public bool tclTriggered: This is a non-static variable, so the best way to retrieve this is by using GetComponent< **IRDSCarControllInput** >() and then accessing the variable, this variable is a bool value, if it returns true, then the TCL have been triggered, otherwise not.

public bool steerHelpTriggered: This is a non-static variable, so the best way to retrieve this is by using GetComponent< **IRDSCarControllInput** >() and then accessing the variable, this variable is a bool value, if it returns true, then the SteerHelp have been triggered, otherwise not.

public float inputClutch: This is a non-static variable, so the best way to retrieve this is by using GetComponent< **IRDSCarControllInput** >() and then accessing the variable, this variable is a float value, for the clutch input, it handle values from 0 to 1.

public bool startEngine: This is a non-static variable, so the best way to retrieve this is by using GetComponent< **IRDSCarControllInput** >() and then accessing the variable, this variable is a bool value, if it is set to true, the car would try to start the engine if it is off.

public IRDSSoundController soundController: This is a non-static variable, so the best way to retrieve this is by using GetComponent< **IRDSCarControllInput** >() and then accessing the variable, this variable is a **IRDSSoundController** type, it returns the car **IRDSSoundController class** instance.

public IRDSWing[] wings: This is a non-static variable, so the best way to retrieve this is by using GetComponent< **IRDSCarControllInput** >() and then accessing the variable, this variable is a **IRDSWing** type, it returns the car **IRDSWing[] class** instance.

public IRDSCarVisuals carVisuals: This is a non-static variable, so the best way to retrieve this is by using GetComponent< **IRDSCarControllInput** >() and then accessing the variable, this variable is a **IRDSCarVisuals** type, it returns the car **IRDSCarVisuals class** instance.

public IRDSAerodynamicResistance airResistance: This is a non-static variable, so the best way to retrieve this is by using GetComponent< **IRDSCarControllInput** >() and then accessing the variable, this variable is a **IRDSAerodynamicResistance** type, it returns the car **IRDSAerodynamicResistance class** instance.

public IRDSAntiRollBar[] antirollbars: This is a non-static variable, so the best way to retrieve this is by using GetComponent< **IRDSCarControllInput** >() and then accessing the variable, this variable is a **IRDSAntiRollBar** type, it returns the car **IRDSAntiRollBar[] class** instance.

public IRDSCarDamage carDamage: This is a non-static variable, so the best way to retrieve this is by using GetComponent< **IRDSCarControllInput** >() and then accessing the variable, this variable is a **IRDSCarDamage** type, it returns the car **IRDSCarDamage class** instance.

public IRDSPlayerControls playerControls: This is a non-static variable, so the best way to retrieve this is by using GetComponent< **IRDSCarControllInput** >() and then accessing the variable, this variable is a **IRDSPlayerControls** type, it returns the car **IRDSPlayerControls class** instance.

Class IRDSDrivetrain

This Class is responsible to handle all the engine calculations, here are the methods available:

IRDSDrivetrain.SetAutomatic: This is a non-static method, so you need to get an instance of this Class, this method Sets a bool for enabling / disabling automatic shifting.

IRDSDrivetrain.GetRPM: This is a non-static method, so you need to get an instance of this Class, this method returns a float with the current RPM of the engine.

IRDSDrivetrain.GetMaxRPM: This is a non-static method, so you need to get an instance of this Class, this method returns a float with the Max RPM of the engine.

IRDSDrivetrain.GetCurrentPower: This is a non-static method, so you need to get an instance of this Class, this method returns a float with the current power of the engine in Kilowatts.

IRDSDrivetrain.GetOriginalPower: This is a non-static method, so you need to get an instance of this Class, this method returns a float with the original power of the engine in Kilowatts.

IRDSDrivetrain.GetGear: This is a non-static method, so you need to get an instance of this Class, this method returns an integer with the current Gear (this has values from 0 to the length-1 of the array GearRatios).

IRDSDrivetrain.GetMaxPower: This is a non-static method, so you need to get an instance of this Class, this method returns a float with the Power of the engine.

IRDSDrivetrain.GetPowerRPM: This is a non-static method, so you need to get an instance of this Class, this method returns a float with the Power RPM of the engine. (for example when you see car specs they write stuff like 600HP@6000 rpm the 6000RPM is the value returned here).

IRDSDrivetrain.GetPSI: This is a non-static method, so you need to get an instance of this Class, this method returns a float with the PSI set of the turbo.

IRDSDrivetrain.GetGearRatios: This is a non-static method, so you need to get an instance of this Class, this method returns an array of type float with all the gear ratios for this engine.

IRDSDrivetrain.GetFinalDriveRatio: This is a non-static method, so you need to get an instance of this Class, this method returns a float with Final Drive Ratio of the Gearbox.

*****These are new methods*****

IRDSDrivetrain.GetMaxTorque(): This is a non-static method, so you need to get an instance of this Class, this method returns a float with the engine max torque (when in play mode, this method returns the actual engine max torque, this changes because of the turbo).

IRDSDrivetrain.GetOriginalTorque(): This is a non-static method, so you need to get an instance of this Class, this method returns a float with the engine original torque (when it is in play mode, this returns the engine max torque showed on the inspector in the car engine setup option).

IRDSDrivetrain.GetMinRPM(): This is a non-static method, so you need to get an instance of this Class, this method returns a float with the engine min rpm.

IRDSDrivetrain.GetOriginalMinRPM(): This is a non-static method, so you need to get an instance of this Class, this method returns a float with the original min rpm.

IRDSDrivetrain.GetEngineBaseFriction(): This is a non-static method, so you need to get an instance of this Class, this method returns a float with the engine base friction.

IRDSDrivetrain.GetEngineRPMFriction(): This is a non-static method, so you need to get an instance of this Class, this method returns a float with the engine rpm friction.

IRDSDrivetrain.GetEngineOrientation(): This is a non-static method, so you need to get an instance of this Class, this method returns a Vector3 with the engine orientation.

IRDSDrivetrain.GetDifferentialLockCoefficient(): This is a non-static method, so you need to get an instance of this Class, this method returns a float with the differential lock coefficient.

IRDSDrivetrain.GetEngineInertia(): This is a non-static method, so you need to get an instance of this Class, this method returns a float with engine inertia.

IRDSDrivetrain.GetTurboAirPresure(): This is a non-static method, so you need to get an instance of this Class, this method returns a float with the actual turbo air pressure in PSI.

IRDSDrivetrain.GetTurboRPM(): This is a non-static method, so you need to get an instance of this Class, this method returns a float with the turbo rpm (this is the rpm that would be used for simulating the turbo lag, the turbo is engaged when this rpm is reached).

IRDSDrivetrain.GetFuelConsume(): This is a non-static method, so you need to get an instance of this Class, this method returns a bool with a value of true if the engine would consume fuel, otherwise the engine won't consume fuel.

IRDSDrivetrain.SetEngineBaseFriction(float Value): This is a non-static method, so you need to get an instance of this Class, this method sets a float, use it to set the engine base friction.

IRDSDrivetrain.SetEngineRPMFriction(float Value): This is a non-static method, so you need to get an instance of this Class, this method sets a float, use it to set the engine rpm friction.

IRDSDrivetrain.SetEngineOrientation(Vector3 Value): This is a non-static method, so you need to get an instance of this Class, this method sets a Vector3 with the engine orientation, this is for applying the engine torque to the car.

IRDSDrivetrain.SetDifferentialLockCoefficient(float Value): This is a non-static method, so you need to get an instance of this Class, this method sets a float, use it to set the differential lock coefficient (0 to 100).

IRDSDrivetrain.SetEngineInertia(float Value): This is a non-static method, so you need to get an instance of this Class, this method sets a float, use it to set the engine inertia.

IRDSDrivetrain.SetTurboRPM(float Value): This is a non-static method, so you need to get an instance of this Class, this method sets a float, use it to set the turbo rpm.

IRDSDrivetrain.SetFuelConsume(bool flag): This is a non-static method, so you need to get an instance of this Class, this method sets a bool, use it to enable fuel consumption.

IRDSDrivetrain.SetTorqueRPM(float Value): This is a non-static method, so you need to get an instance of this Class, this method sets a float, use it to set the torque rpm of the engine.

IRDSDrivetrain.SetMinRPM(float Value): This is a non-static method, so you need to get an instance of this Class, this method sets a float, use it to set the engine min rpm.

IRDSDrivetrain.SetMaxRPM(float Value): This is a non-static method, so you need to get an instance of this Class, this method sets a float, use it to set the engine max rpm.

IRDSDrivetrain.SetFinalDriveRatio(float Value): This is a non-static method, so you need to get an instance of this Class, this method sets a float with Final Drive Ratio of the Gearbox.

IRDSDrivetrain.SetGearRatios(float[] Value): This is a non-static method, so you need to get an instance of this Class, this method sets a float[] with the engine gear ratios. (note: the item on this array with index 0 is the reverse gear, the index 1 is neutral and so on)

IRDSDrivetrain.SetTurboPSI(float Value): This is a non-static method, so you need to get an instance of this Class, this method sets a float, use it to set the turbo PSI.

IRDSDrivetrain.SetPowerRPM(float Value): This is a non-static method, so you need to get an instance of this Class, this method sets a float, use it to set the engine Power rpm.

IRDSDrivetrain.SetMaxPower(float Value): This is a non-static method, so you need to get an instance of this Class, this method sets a float, use it to set the engine Max Power.

IRDSDrivetrain.SetGear(int Value): This is a non-static method, so you need to get an instance of this Class, this method sets an int, use it to set the current gear.

IRDSDrivetrain.SetMaxTorque(float Value): This is a non-static method, so you need to get an instance of this Class, this method sets a float, use it to set the engine max torque.

IRDSDrivetrain.GetAutomatic(): This is a non-static method, so you need to get an instance of this Class, this method returns a bool, value of true if automatic shifting is enabled.

*****New variables on this class that are exposed****

IRDSDrivetrain.shiftUpRpmPercent: This is a non-static variable, so you need to get an instance of this Class, this variable holds the shift up percentage for automatic shifting (0 to 1).

IRDSDrivetrain.shiftDownRpmPercent: This is a non-static variable, so you need to get an instance of this Class, this variable holds the shift down percentage for automatic shifting (0 to 1).

public bool automaticClucth: This is a non-static variable, so you need to get an instance of this Class, this variable sets if the clutch is automatic or not.

public float clutchInput: This is a non-static variable, so you need to get an instance of this Class, this variable is used for the clutch input valid values are (0 to 1).

public int gearWanted: This is a non-static variable, so you need to get an instance of this Class, this variable is used to set here the gear to shift to, valid values are 0 to the length of the gear ratios array -1 .

public float lastTimeShifted: This variable is for internal use only.

public float ShiftSpeed: This is a non-static variable, so you need to get an instance of this Class, this variable sets the shoft speed, higher values makes shift between gears slower, and lower values make it shift faster.

public float lastStartEngineTime: This variable is for internal use only.

public bool startEngine: This is a non-static variable, so you need to get an instance of this Class, this variable if is set to true, would try to start the engine, is better to use the StartEngine variable on the class IRDSCarControlInput instead of using this one directly.

public float engineStartingTime: This is a non-static variable, so you need to get an instance of this Class, this variable sets the time the engine takes to start in seconds.

public float engineCrankTime: This is a non-static variable, so you need to get an instance of this Class, this variable sets the time the engine would be cranking before start, in seconds.

public float fuelConsumptionMultiplier: This is a non-static variable, so you need to get an instance of this Class, this variable sets the fuel consumption multiplier, so the engine consumes.

public float nitroFuel: This is a non-static variable, so you need to get an instance of this Class, this variable sets the amount of nitro fuel.

public float nitroBoostDurability: This is a non-static variable, so you need to get an instance of this Class, this variable sets the boost/Durability of the nitro, higher values makes more boost but less durability, and lower values makes less boost but more durability, default is 1.

public bool useNitro: This is a non-static variable, so you need to get an instance of this Class, this variable if is set to true, turns Nitro on, and the cars would immediately would have more power.

Class IRDSCarControllerAI

This class is the responsible of driving the AI Cars and handling all the AI Magic. The methods available are:

GetMyPit: This is a non-static method, so you need to get an instance of this Class; this method returns an integer with the pit number for this car.

GetLap: This is a non-static method, so you need to get an instance of this Class; this method returns an integer with the current lap for this car/driver.

GetPosition: This is a non-static method, so you need to get an instance of this Class, this method returns a float an internal calculation to be used to get the position on the race, and this number is negative.

SetPosition(float position): This is a non-static method, so you need to get an instance of this Class, this method sets the positions of the car on the race, for internal use only (do not use it!).

SetUnstuckTime(float time): This is a non-static method, so you need to get an instance of this Class, this method sets the unstuck time of the car on the race.

GetEllapsedTime: This is a non-static method, so you need to get an instance of this Class; this method returns a float with the last laptime.

GetTotalTime: This is a non-static method, so you need to get an instance of this Class, this method returns a float with the current total time, if the race ended for this car, and this value would remain the same even if the car still goes around the track.

SetTotalTime(float time): This is a non-static method, so you need to get an instance of this Class, this method sets the total time, for internal use only (Do not use it!).

GetStartTime: This is a non-static method, so you need to get an instance of this Class; this method returns a float with the initial race time (the time at the moment that the race began).

GetMaxSpeed: This is a non-static method, so you need to get an instance of this Class; this method returns a float with the current max speed at that time.

GetMaxSpeedEspec(int i): This is a non-static method, so you need to get an instance of this Class, this method returns a float with the max speed for a particular waypoint number.

GetEndRace: This is a non-static method, so you need to get an instance of this Class; this method returns a bool that is true if this car ended the race and false if not.

GetCarDepth: This is a non-static method, so you need to get an instance of this Class; this method returns a float with this car total depth in meters.

GetCarSpeed: This is a non-static method, so you need to get an instance of this Class; this method returns a float with the current car speed.

GetTopSpeed: This is a non-static method, so you need to get an instance of this Class, this method returns a float with this car top speed.

GetCarInputs: This is a non-static method, so you need to get an instance of this Class, this method returns a IRDSCarControllerInput with the instance of this car.

SetLap(int lap): This is a non-static method, so you need to get an instance of this Class, this method sets the current lap, this is for internal use only (do not use it!).

GetCarName: This is a non-static method, so you need to get an instance of this Class; this method returns a string with the name of this car.

GetDriverName: This is a non-static method, so you need to get an instance of this Class; this method returns a string with the name of driver.

***New Methods for this class**

SetMyPit(int pitNo): This is a non-static method, so you need to get an instance of this Class; this method sets an int value, this is for setting the pit number for this driver, this is done automatically by the AI System.

int GetCurrentWaypoint(): This is a non-static method, so you need to get an instance of this Class; this method returns an int with the current waypoint number that the driver is following.

int GetCurrentWaypointS(): This is a non-static method, so you need to get an instance of this Class; this method returns an int with the current waypoint number the car is at.

float GetCarWidth(): This is a non-static method, so you need to get an instance of this Class; this method returns the car width.

SetRespawnTimer(float time): This is a non-static method, so you need to get an instance of this Class; this method sets a float with the amount of time that needs to be the car up-side-down in order to be flipped.

SetStartTime(float time): This is a non-static method, so you need to get an instance of this Class; this method sets a float with the time that would be used as starting race time, this is done automatically by the AI System.

float GetFastestLapTime(): This is a non-static method, so you need to get an instance of this Class; this method returns a float with the current fastest lap time of this driver.

EnableSpeedRestriction(int firstWaypoint, int lastWaypoint, bool flag, float speedLimit): This is a non-static method, so you need to get an instance of this Class; this method sets the speed restriction for simulating a yellow flag situation, the first value "firstWaypoint" and the second value "lastWaypoint" are used to set the portion of the track that would have the speed restriction, the third value is to set the flag, true would enable it, and the last value is the top speed for the restriction.

DisableSpeedRestriction(): This is a non-static method, so you need to get an instance of this Class; this method disables the speed restriction.

FlipCar (): This is a non-static method, so you need to get an instance of this Class; this method flips and resets the car.

public bool GetIsOvertaking(): This returns true if the AI is overtaking an opponent

public bool GetIsAvoidingOpponentSideways(): This returns true if the AI is avoiding an opponent

public bool GetIsBrakingOnOpponent(): This returns true if the AI is braking because there is an opponent in front.

public bool GetWrongWay(): This returns true if the car is going wrong way

public bool GetStucked(): This returns true if the AI is stuck

public void SetStucked(bool val): You can use this method to set if the AI got stuck

public float GetDriftAngle(): Get the drift angle value or the also called Yaw

public IRDSManager GetManager(): This returns the reference to the instance of the IRDSManager class

public void SetMaxSteerLockExternalPhysics(float value) : Use this method for setting the max steer lock when using third party car physics as Edys and UnityCar

public float GetMaxSteerLockExternalPhysics(): This returns the current Max steer lock

public void SetDriverName(string newName): This sets the drivers names of this car

public void SetFrictionExtaernalPhysics(float[] value) : This sets the value of the tire friction for this car, use this for third party car physics

public void SetHFactorExternalPhysics(float value): This sets the value for the H factor used for the calculation of the braking capability of the car, this factor have an example of how to calculate it on the UnityCar interface scripts.

public void SetClFactorExternalPhysics(float value): This sets the value for the Cl factor used for the calculation of the braking capability of the car, this factor have an example of how to calculate it on the UnityCar interface scripts (this is normally the air drag coefficient).

public void SetCWFactorExternalPhysics(float value) : This sets the value for the CW factor used for the calculation of the braking capability of the car, this factor have an example of how to calculate it on the UnityCar interface scripts

public void SetWingaFactorExternalPhysics(float value) : This sets the value for the Wingca factor used for the calculation of the braking capability of the car, this factor have an example of how to calculate it on the UnityCar interface scripts (this is normally the drag coefficient of the wings).

public void SetTireSlipAngleExternalPhysics(bool value) : This is used to set if the car tires are slipping, so the AI could control the car.

public void SetTyreWearExternalPhysics(float value): Use this to set the value of the current tyre wear.

*****The following methods are for getting and setting the AI variables, which are not going to be explained here, see the AI Drivers settings for the explanation of each variable*****

public float GetAggressivenessOnBrake()

public void SetAggressivenessOnBrake(float aggressivenessOnBrake)

public float GetSpeedSteeringFactor()

public void SetSpeedSteeringFactor (float speedSteeringFac)

public float GetLookahead_factor()

public void SetLookahead_factor(float Lookahead_factor)

public float GetCorneringSpeedFactor()

public void SetCorneringSpeedFactor(float corneringSpeedFactor)

```
 public float GetMaxDriftAngle()

public void SetMaxDriftAngle(float maxDriftAng)

 public float GetSteeringDriftFactor()

public void SetSteeringDriftFactor(float steeringDriftFac)

 public float GetCollisionSideFactor()

public void SetCollisionSideFactor(float collisionSideFac)

 public float GetOvertakeFactor()

public void SetOvertakeFactor(float overtakeFac)

 public float GetOvertakeSpeedDiference()

public void SetOvertakeSpeedDiference(float overtakeSpeedDif)

public float GetHumanError()

public void SetHumanError (float humanErr)

 public float GetOvertakeOffsetIncrementMin()

public void SetOvertakeOffsetIncrementMin(float overtakeOffsetIncMin)

public float GetOvertakeOffsetIncrementMax()

public void SetOvertakeOffsetIncrementMax(float overtakeOffsetIncMax)

 public float GetShifFactor()

public void SetShifFactor(float shifFac)

 public float GetFuelTankCapacity()

public void SetFuelTankCapacity(float fuelTankCap)

 public float GetTyrechangePorcentage()

public void SetTyrechangePorcentage(float tyrechangePorcent)

public float GetFuelloadPorcentage()

public void SetFuelloadPorcentage(float fuelloadPorcent)

public float GetFullAccelMaring()

public void SetFullAccelMaring(float fullAccelMaring
```

public float GetFrontCollDist()

public void SetFrontCollDist(float frontCollD)

public float GetBackCollDist()

public void SetBackCollDist(float backCollD)

public float GetSideMargin()

public void SetSideMargin(float sideMar)

public float GetHeightMargin()

public void SetHeightMargin(float heightMar)

public float GetLength_Margin()

public void SetLength_Margin(float length_margin)

public float GetSide_Margin()

public void SetSide_Margin(float Side_Margin)

*****New Variables for this class that are exposed*****

public bool respawn; Use this to enable or disable this car respawning.

public bool respawnAtWP; Use this to enable or disable this car from respawning at the last waypoint.

public float resetTime; Use this to set the amount of time to respawn.

public float fastestLapTime; This variable holds the current fastest lap time for this driver.

public bool yellowFlag; Use the method **EnableSpeedRestriction and DisableSpeedRestriction instead of this variable directly.**

public bool blueFlag; Use this to set the blue flag for this driver, if enabled, this car would let pass the car that are lapping him.

public int firstWaypointSpdRest; Use the method **EnableSpeedRestriction and DisableSpeedRestriction instead of this variable directly.**

public int lastWaypointSpdRest; Use the method **EnableSpeedRestriction and DisableSpeedRestriction instead of this variable directly.**

public float spdLimit; Use the method **EnableSpeedRestriction and DisableSpeedRestriction instead of this variable directly.**

public bool driveThroughPits; Use this to set the driver to just pass by the pits and don't stop, need to be used with SetGoPits to make the driver go to the pits and drive through.

public bool chargeFuel; Use this to set if you want to charge fuel or not when pitting.

public bool changeFrontTires; Use this to set if you want to change front tires or not when pitting.

public bool changeRearTires; Use this to set if you want to change rear tires or not when pitting.

public bool autoSteering: Use this variable to enable auto steering for a player car.

public bool autoThrottle: Use this variables to enable auto throttle for a player car.

public bool autoBrake: Use this variable to enable auto brake for a player car.

public int playerNumber: This variable have the player number, if the car is AI controlled this variable would have a value of -1, otherwise it would have 1, or 2 , or 3, etc. depending on the player number.

public int racePosition: This variable contains the current position on the race for this car.

public int myGridPosition: This variable contains the number of the grid position for this car, it is used to know when this car is removed that this position is available to instantiate another car on the race at this grid position. (this is for internal use only).

Class IRDSCarCamera

This class is responsible of controlling the Main Camera.

GetTarget: This is a non-static method, so you need to get an instance of this Class; this method returns a Transform with the current camera target.

ActivateRoadCamera: This is a non-static method, so you need to get an instance of this Class; this method activates the road camera mode.

DeactivateRoadCamera: This is a non-static method, so you need to get an instance of this Class; this method deactivates the road camera mode.

changeTarget: This is a non-static method, so you need to get an instance of this Class; this method cycles the current camera target with all the cars on the scene.

changeView: This is a non-static method, so you need to get an instance of this Class; this method cycles between all the current car camera positions (cockpit, bumper, hood, etc).

ChangeToPlayerCar: Method to get back to players cars at once, this is helpful when the player just cycled on opponents cars to see them in action and want to get back the camera quickly to his car, see CameraControll for an example (it is assigned there to keyboard key "B").

changeTargetbyPosition(): This is a non-static method, so you need to get an instance of this Class; this method cycles the current camera target with all the cars on the scene by their race position.

changeTargetbyPositionBackwards(): This is a non-static method, so you need to get an instance of this Class; this method cycles the current camera target with all the cars on the scene by their race position but backwards.

changeTarget(int instanceID): This is a non-static method, so you need to get an instance of this Class; this method changes the current camera target by the specified instance ID of the class IRDSCarControllInput, so you need to pass to this method the instance ID of the IRDSCarControllInput of the car you want the camera to target on.

**** Variables exposed for this class ****

public float delayStartTime: This variable sets the delay time to be used before the counter would start to count to start the race.

public float lastDelayTime: This variable is for internal use only.

public bool freeCamera: Activate this to enable a free camera on the delay time, so you could manipulate the camera position and rotation during this time.

public bool forceFreeCamera: Enable this to force the camera to be free to move even during the race, so you could manipulate its position and rotation freely.

public int zoomFactorMin: This variable sets the minimum zoom factor for the road camera mode (this is the minimum field of view).

public int zoomFactorMax: This variable sets the maximum zoom factor for the road camera mode (this is the maximum field of view).

public int zoomingDistance: Use this to set the distance on which the camera would start to zoom at the target.

Class CameraControl

You can modify this script to fit your needs, it controls the change target, change view and road camera activation of the IRDSCarCamera.

Class IRDSWheel

This Class is responsible for calculating the Tire Forces.

GetGrip: This method returns the grip value, this is a multiplier to make the tires behave more like an arcade Car Physics.

GetNormalForce: This method returns the NormalForce for this tire.

GetLocalVelo: This method returns this wheel local velocity at the Hub of the wheel.

CalcMz: This method returns the alignment force, use for Force Feedback.

New Methods for this class

public GameObject GetTireModel(): This method returns the Tire GameObject that contains the renderer with the tire mesh.

public GameObject GetCaliperModel(): This method returns the Brake Caliper GameObject that contains the renderer with the Caliper mesh.

public void SetTireModel(GameObject Model): This method sets the Tire GameObject that contains the renderer with the tire mesh.

public void SetCaliperModel(GameObject Model): This method sets the Brake Caliper GameObject that contains the renderer with the Caliper mesh.

public float GetTyreTemp(): This method returns the actual tire temp percentage. A value of 1 indicates it is at full temp.

public bool GetWear(): This method returns true if the tire can wear.

public void SetWear(bool Wear): Use this method to set if this tire can wear or not.

public float GetTyreHardness(): Use this method to get the tire hardness value. Greater values makes the tire harder, so it gets hotter slowly and wear slowly too, a softer tire would get hotter faster and would wear faster too.

public void SetTyreHardness(float hardness): Use this to set the tire hardness value(typical values are from 1 to 100)

public string GetWheelPosition(): This method returns the positions of the tire as string (FL, FR, RL, RR).

//New Gets and Sets Methods

public void SetGrip(float Value): Use this method to set the grip coefficient of this tire.

public float[] GetBCoefficients(): Use this method to get the pacejka b (Longitudinal) coefficients of this tire.

public float[] GetACoefficients(): Use this method to get the pacejka a (Lateral)coefficients of this tire.

public float[] GetCCoefficients(): Use this method to get the pacejka c (aligning)coefficients of this tire.

public void SetBCoefficients(float[] coef): Use this method to set the pacejka b coefficients of this tire.

public void SetACoefficients(float[] coef): Use this method to set the pacejka a coefficients of this tire.

public void SetCCoefficients(float[] coef): Use this method to set the pacejka c coefficients of this tire.

public float GetRadius(): This method returns the radius of the tire.

public void SetRadius(float Value): This method sets the radius of the tire.

public float GetSuspensionTravel(): This methods returns the suspension travel for this tire

public void SetSuspensionTravel(float Value): Use this method to set the suspension travel.

public float GetDampingRatio(): This method return the damping ratio.

public void SetDampingRatio(float Value): This method sets the damping ratio.

public float GetInertia(): This method returns the tire inertia.

public void SetInertia(float Value): This method sets the tire inertia.

public float GetToeAngle(): This method returns the tire toe angle.

public void SetToeAngle(float Value): This method sets the tire toe angle.

public float GetCamberAngle(): This method returns the tire camber angle.

public float GetCurrentCamberAngle():This method returns the current camber angle.

public void SetCamberAngle(float Value): This method sets the tire camber angle.

public float GetBrakeFrictionTorque() : This method returns the tire brake friction torque.

public void SetBrakeFrictionTorque(float Value): This method sets the tire brake friction torque.

public float GetHandBrakeFrictionTorque(): This method returns the tire Hand brake friction torque.

public void SetHandBrakeFrictionTorque(float Value): This method sets the tire Hand brake friction torque.

public float GetFrictionTorque(): This method returns the tire friction torque.

public void SetFrictionTorque(float Value): This method sets the tire friction torque.

public float GetMaxSteeringAngle(): This method returns the tire max steering angle.

public void SetMaxSteeringAngle(float Value): This method sets the tire max steering angle.

public float GetMassFraction(): This method returns the tire mass fraction.

public void SetMassFraction(float Value): This method sets the tire mass fraction.

public float GetAngularVelocity(): This method returns the tire angular velocity.

public float GetSlipRatio(): This method returns the tire Slip ratio.

public float GetSlipAngle(): This method returns the tire slip angle ratio.

public float GetSlipVelo(): This method returns the tire Slip velocity.

public float GetCompression(): This method returns the tire suspension compression.

public Vector3 GetWheelVelo(): This method returns the tire velocity vector.

public float GetSpringForceRatio(): This method returns the suspension spring force ratio.

public void SetSpringForceRatio(float Value): This method sets the suspension spring force ratio.

*** New Variables exposed on this class***

Float staticFrictionCoefficient: This variable sets the static friction of the tire to avoid slipping when at low speed or full stop; Best values are between 1 and 1.2.

Class IRDSAerodynamicResistance

public Vector3 GetCoefficients() : This method returns the aerodynamic coefficients for the 3 axis.

public void SetCoefficients (Vector3 Value): This method sets the aerodynamic coefficients for the 3 axis.

Class IRDSAntiRollBar

public float GetCoefficient(): This method returns the anti-roll bar coefficient.

public void SetCoefficient(float Value): This method sets the anti-roll bar coefficient.

Class IRDSPlayerControls

public float GetThrottleTime()

public float GetThrottleTimeTraction ()

public float GetThrottleReleaseTime()

public float GetThrottleReleaseTimeTraction()

public float GetSteerTime()

public float GetVeloSteerTime()

public float GetSteerReleaseTime()

public float GetVeloSteerReleaseTime()

public float GetSteerCorrectionFactor()

public void SetThrottleTime(float Value)

public void SetThrottleTimeTraction (float Value)

public void SetThrottleReleaseTime(float Value)

public void SetThrottleReleaseTimeTraction(float Value)

public void SetSteerTime(float Value)

public void SetVeloSteerTime(float Value)

public void SetSteerReleaseTime(float Value)

public void SetVeloSteerReleaseTime(float Value)

public void SetSteerCorrectionFactor(float Value)

Variables exposed

public float clutchInput;

public float clutchTime = 1.0f;

public float clutchReleaseTime = 0.5f;

**Analog control assignment **

public string steeringAxis = "Horizontal";

public string throttleAxis = "Vertical1";

public string brakeAxis = "Vertical2";

public string shiftUpButton = "ShiftUp";

public string shiftDownButton = "ShiftDown";

public string handbrakeButton = "Jump";

public string clutchAxis = "Clutch";

public string nitroButton = "Nitro";

public string startEngineButton = "StartEngine";

**Digital control assignment **

public KeyCode throttleKey = KeyCode.UpArrow;

public KeyCode altThrottleKey = KeyCode.LeftShift;

public KeyCode brakesKey = KeyCode.DownArrow;

public KeyCode leftKey = KeyCode.LeftArrow;

public KeyCode rightKey = KeyCode.RightArrow;

public KeyCode handbrakeKey = KeyCode.Space;

public KeyCode shiftUpKey = KeyCode.A;

public KeyCode shiftDownKey = KeyCode.Z;

public KeyCode clutchKey = KeyCode.S;

public KeyCode startEngineKey = KeyCode.Q;

public KeyCode nitroKey = KeyCode.LeftControl;

Class IRDSWing

public float GetLiftCoefficient()

public void SetLiftCoefficient(float Value)

*** variables exposed ***

public float area = 1;

public float angle = 1;

public float airDensity = 1.23f;

public float dragCoefficient;

Class IRDSPlaceCars

On this class you can access some new methods that allows to add and remove players dynamically during the race, this is useful for creating last man standing races.

The new methods are:

public bool RemoveCar(IRDSCarControllerAI car): This is a non-static method, you need to get the instance of the IRDSPlaceCars class, just call this method and pass the instance of IRDSCarControllerAI from the car you want to remove from the race.

public IRDSCarControllerAI AddNewCar(Object car): This is a non-static method, you need to get the instance of the IRDSPlaceCars class, just call this method and pass the prefab of the car you want to instantiate, you would need to set all the variables manually, so it is better to use one of the two next methods to add cars to the race.

public IRDSCarControllerAI AddNewAICar(Object car, Color carColor, string driverName): This is a non-static method, you need to get the instance of the IRDSPlaceCars class, just call this method and pass the variables indicated on the method, the first variable is the car prefab that is on the resources folder or any other but that have all the IRDS scripts on the car, the second is the color of the car, and the last one is the name of the driver, the AI settings would be gotten from the levelload object driver settings array.

public IRDSCarControllerAI AddNewPlayerCar(Object car, IRDSOtherPlayers controls, string driverName): This is a non-static method, you need to get the instance of the IRDSPlaceCars class, just call this method and pass the variables indicated on the method, the first variable is the car prefab that is on the resources folder or any other but that have all the IRDS scripts on the car, the second is player settings, you need to create your own IRDSOtherPlayers instance and set the parameters there, or you can use one of the instances already on the levelload object (IRDSLevelLoadVariables), and the last one is the name of the driver.