Checkpoint Based Car AI

by Ovdtech
Unity publisher profile: https://assetstore.unity.com/publishers/93673
Asset link: https://assetstore.unity.com/packages/add-ons/checkpoint-based-car-ai-277490
Youtube channel: https://www.youtube.com/channel/UCOy_CqiqEmKUOhmCAL2DFYQ

Summary: Checkpoint Based Car AI is a Unity asset designed to simulate real life traffic through an efficient system designed in c#. This is a complete guide on how to use the asset, starting off with tutorials, then moving on to explaining each feature on all included scripts.

Tutorials:

How to connect your car game object to the ai script

- Make sure your car wheels are separate from the car game model.
- Make sure your car collider is on the parent game object. In other words, when the CarAiController.cs script is imported, it is necessary to be on the same game object that the collider is on.
- Create a rigid body for the car. Make sure the mass is set high (ex: 1000).
- Create the wheel colliders for your car wheels.
- Go to Window -> CarAI -> Setup car
- Drag the car from your scene in the car model field.
- Drag all the wheel transforms and the wheel colliders.
- You can leave acceleration and breaking thresholds to default.
- Add a speed limit to the car (ex: 50). This speed limit will be changed by the checkpoints. The speed is measured in km/h.
- Create an empty object and make sure it is a child of the car game object. Place it at the front of the car in the middle and make sure it is placed outside of the car hitbox but close to it. This empty game object will be used by the AI to detect objects in front of the car and will stop. It ignores triggers.
- Drag the empty game object into the check position field.
- Press apply.
- **NOTE**: If you want the car to go uphill make sure to deselect the car from detecting the ground, otherwise the car will stop if the raycast goes through the ground.

.How to create checkpoints

- Go to Window -> CarAI -> Spawn checkpoints

- Press spawn checkpoint.
- A checkpoint will be spawned at position (0, 0, 0)
- If you press on the checkpoint, you will see instructions there on how to place the next checkpoints at the mouse position.

How to create an intersection

- Go to Window -> CarAI -> Create intersection
- Add the number of stops. This is basically the number of road connections your intersection has.
- Press spawn intersection.
- Place your stops according to your intersection.
- |
- For example, if you add three stops that means your intersection looks like in the image below. The stops indicate the AI if it can continue driving or stop.
- You can change the amount of time (in seconds) that the light stays green.

How to create a car spawner

- Create a cube
- It is ideal for it to be bigger than the car.
- Make the cube a trigger. (Press on the cube, go to BoxCollider and check isTrigger.)
- Set Up a car (or more) that will serve as model(s) for the script to copy. (Check "How to setup the car" tutorial for setting up the car)
- Got to the car model(s) you created and uncheck "isCarControlledByAI"
- Add "carspawnerscript" to the cube you created
- Add the car model(s) to the "cars" list
- Add the total number of cars that will be spawned
- Add the starting checkpoint.
- Add the time interval. (Ex: If you add the number 3, a car will spawn every 3 seconds if other cars are not colliding with the trigger you created.)
- The next variables are for simulating realistic traffic. You can leave them as default but it won't be realistic because all the cars will drive the same, with the same distance between them.

What are If Blocks and how to create them

An If Block is an empty game object that has two children components: a checker and a stopper. The checker is a trigger that checks if any car (game object that contains the CarAIController.cs script) and switches a variable in the stopper script. This variable then

makes the cars that enter the stopper (the stopper being a trigger too) stop. This is useful for personalizing an intersection, making cars enter lanes or creating a roundabout.

To create them is very easy: go to Window -> CarAI -> Spawn if block

How to setup the taxi system

- Click on the car you want to make a Taxi.
- Add the TaxiScript.cs to it.
- Go to the CarAIController script and check the TaxiMode box.
- Now create a new script and name it however you want.
- In the script, you will need a reference to this TaxiScript.cs we created before.
- Now everytime you want the car to go to a destination, you have to enter the start and end checkpoints of the TaxiScript first, then call the ComputeRoute() function.

You will also find video versions of these tutorials on my youtube channel.

Scripts:

1. **CarAIController.cs**
2. **CheckpointScript.cs**
3. **IntersectionScript.cs**
4. **StopScript.cs**
5. **carspawnerscript.cs**

1. CarAIController.cs

- **frontRight (Transform)** - the transform of the front right wheel of your car.
- **frontLeft (Transform)** - the transform of the front left wheel of your car.
- **rearRight (Transform)** - the transform of the rear right wheel of your car.
- **rearLeft (Transform)** - the transform of the rear left wheel of your car.
- **frontRightCollider (WheelCollider)** - the WheelCollider of the front right wheel of your car.
- **frontLeftCollider (WheelCollider)** - the WheelCollider of the front left wheel of your car.
- **rearRightCollider (WheelCollider)** - the WheelCollider of the rear right wheel of your car.
- **rearLeftCollider (WheelCollider)** - the WheelCollider of the rear left wheel of your car.
- **nextCheckpoint (Transform)** - the transform of the checkpoint that the AI will look for and go to. Checkpoints are triggers. When the car enters the Checkpoint, the CheckpointScript.cs will change this variable automatically to the next checkpoint.

- **checks (List<Transform>)** - a list of "checks" transforms. These are basically sensors (raycasts) for the AI to interact with the surroundings. You can add as many of these as you like. Make sure to place them at the front of the car.
- **CheckPointSearch (bool)** - if it's true, the AI will check for checkpoints.
- **objectDetected (bool)** - if it's true it means there is an object in front of the car and the AI will slow down and eventually stop. This variable is meant for reading purposes only.
- **isCarControllerByAI (bool)** - does the exact same thing as CheckPointSearch(bool).
- **seenLayers (LayerMask)** - A LayerMask containing all the layers the AI will react to. By default it is set to Everything, but if you want it to ignore other objects (for example: the ground) you can uncheck the layer you want the AI to ignore.
- **kmh (int)** - The speed of the car measured in Km/h. This variable is meant for reading only.
- **speedLimit (int)** - The speed limit that the AI will try to respect.
- **distanceFromObjects (float)** - The distance that the AI will keep from other objects (default:2). This was created to be used in the carspawnerscript so that we can achieve a realistic traffic environment where everyone haves their own random style of driving.
- **recklessnessThreshold (int)** - The number (in Km/h) that the AI will go above/below the speed limit (ex: 10 - means that the AI will drive with 10 Km/h over the speed limit). Default is set to 0, meaning that the AI will respect the speed limit.
- **acceleration (float)** - Acceleration threshold. It is ideal to be smaller than the breaking threshold.
- **breaking (float)** - Breaking threshold. It is ideal to be bigger than the acceleration.

2. CheckpointScript.cs

- **speedLimit (int)** - The speed limit that the script will apply to cars that enter the checkpoint. When this happens, the script will change the speedLimit variable of the CarAIController.cs to the when this script has.
- **nextCheckpoints (List<Transform>)** - A list of the next checkpoint(s) that will help the car continue the path. If there are more than one in this list, the script will choose one randomly and will assign it to the nextCheckpoint variable of the CarAIController.cs script.

3. IntersectionScript.cs

- **stops (List<GameObject>)** - A list of all the stops from the intersection.
- **wait (float)** - The time a stop is green in seconds.

4. StopScript.cs

- **stop (bool)** - If it is true the car that touches the stop trigger will stop.

5.carspawnerscript.cs

- **cars (List<GameObject>)** - A list containing all the cars that will be instantiated (copied). Ideally these cars should be hidden somewhere in the map and the isCarControlledByAI variable should be set to false so that the car (or cars) don't move.
- **numberOfCarsToSpawn (int)** - The total number of cars that will be spawned.
- **startingCheckpoint (Transform)** - The first checkpoint that the car(s) will be redirected to.
- **timeIntervalBetweenCarsInSeconds(float)** - The time frequency at which the cars will be spawned (ex: 5 - means that one car will be spawned every 5 seconds). Keep in mind that if a car collides with the spawner (the spawner being a trigger), no cars will be spawned until the car spawner "space" (or area) will be cleared.
- **distanceKeptMin (float)** - The minimum distance of the interval where the script will randomly assign the distanceFromObjects variable from CarAIController.cs script for each car spawned.
- **distanceKeptMax (float)** - The maximum distance of the interval where the script will randomly assign the distanceFromObjects variable from CarAIController.cs script for each car spawned.
- **recklessnessMin(int)** - The minimum threshold of the interval where the script will randomly assign the recklessnessThreshold variable from CarAIController.cs script for each car spawned.
- **recklessnessMax(int)** - The maximum threshold of the interval where the script will randomly assign the recklessnessThreshold variable from CarAIController.cs script for each car spawned.

5.TaxiScript.cs

- **startCheckpoint (Transform)** - The first checkpoint that the script will start calculating the route from.
- **endCheckpoint (Transform)** - The target checkpoint that the car will go to.
- **bestRoute(List<Transform>)** - A list of the checkpoints that the car will follow to reach the target checkpoint.
- **coroutines (int)** - The number of coroutines that are running to calculate the best route. If this number is greater than 0, it means that the script is computing.
- **iterations(int)** - The number of checkpoints that will be processed per coroutine. Increasing this number can make the game run slower depending on your pc specifications, but it will calculate the best route faster.

5.CheckerScript.cs

- **stopScript (StopScript)** - The StopScript.cs of the assigned stopper.