

Article

Not peer-reviewed version

---

# Partially Automated Vehicle Simulation for Studying Driver Distraction in Real Scenarios

---

Mohammed Bendahmane and [Lluís Ribas-Xirgo](#)\*

Posted Date: 19 January 2024

doi: 10.20944/preprints202401.1435.v1

Keywords: autonomous vehicles; drivers' distraction; simulated driving




Preprints.org is a free multidiscipline platform providing preprint service that is dedicated to making early versions of research outputs permanently available and citable. Preprints posted at Preprints.org appear in Web of Science, Crossref, Google Scholar, Scilit, Europe PMC.

Copyright: This is an open access article distributed under the Creative Commons Attribution License which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

## Article

# Partially Automated Vehicle Simulation for Studying Driver Distraction in Real Scenarios

Mohammed Bendahmane <sup>†,‡</sup> and Lluís Ribas-Xirgo <sup>\*,†,‡</sup> 

Universitat Autònoma de Barcelona; Mohammed.Bendahmane@autonoma.cat

\* Correspondence: Lluís.Ribas@uab.cat

† Current address: Carrer de les Sitges, School of Engineering, Campus UAB, Barcelona (Spain).

‡ These authors contributed equally to this work.

**Abstract:** Traffic reports, naturalistic observations and in-lab tests are carried out to determine the influence of driver's distractions in car crashes and to develop technology to detect and prevent them from happening. Paradoxically, driver's distractions are more prone to occur with automated driving vehicles, which might turn into cancelling some of their advantages. One of the best strategies to cope with this problem is education. In this work we propose a simulation environment to design laboratory experiments to study driver's distractions as well as simple games to train drivers to take over the control of the vehicle when required. The resulting simulator enables drivers to experience driving partially automated cars in their neighborhoods or in places of their choice with different levels of events to check their ability in taking over the control when required.

**Keywords:** autonomous vehicles; car-driving game; drivers' distraction; simulated driving

## 1. Introduction

Driving on highways or in the neighborhood in a sunny day with few other vehicles on the road are situations where our sense of continuity makes us focus our attention on non-driving related tasks or get into drowsiness. Even if we drive in complex situations with denser traffic or pedestrians around, which may make unpredictable movements, we trust on paying attention to these because of our natural trend to focus on changes in the environment. Additionally, we have an extra confidence on correctly dealing with these situations if our vehicle has some kind of advanced driving system (ADS). Unfortunately, as with our feeling that vehicles and pedestrians have some inertia in their movements, we extrapolate that the ADSs rightly react to any potentially hazardous situation, but we might be in trouble when they ask us to take over the control of the vehicle in the shortest possible time.

In this work, we have designed a simulation platform so that you can train in these situations on the streets of your neighborhood, or the place you prefer, and so that the data collected with the experiments can be used to improve the driving environment, both that of vehicle and the roads on which it travels.

Although there are some games that let players drive on Google maps and the like (e.g. [<https://framesynthesis.com/drivingsimulator/maps>]), the interaction with the scenario is too elementary and cannot be used to measure data about the driver-players' behaviors. In the opposite side, there are realistic virtual reality games based on actual video footages [<https://driving-tests.org/driving-simulator>] or simulated environments [<https://www.cgasimulation.com>, <https://drivesimsimulator.com/en>, <https://citycardriving.com>] which deliver a good immersive experience to the users at the cost of computing power and with limited number of scenarios. The same is true for simulators oriented to testing self-driving cars [1], as they require accurate representation of the 3D world so that the simulated sensors can get data of the same quality than their real counterparts.

This is the first work where simulated traffic environments come from real maps and the simulation is computationally inexpensive, like many of the psychometric driving test tools, with the addition of including the modeling of partially autonomous driving of the vehicle. Differently

from many current driving simulators that try to convey realistic scenes to the users, the proposed solution relies on Gestalt principles to fill in the blanks in the simplified visual environment and on the user-engagement effect of the situated game. In this sense, it provides a generic, non-compute-intensive tool for driving schools [2] and researchers. Like the psychotechnical test drive games and tools to assess and train coordination, reflexes and prediction abilities[3,4], our solution can help doing the same for partially autonomous cars.

### 1.1. Literature Review

Most of us suffer significant impairment when using cell phones while driving [5] and are less aware of road hazards when driving partially automated vehicles (PAVs) [6]. In effect, drivers, particularly the novice [7], trust on advanced driver-assistance systems (ADAS) and ADS [8] and get distracted easily. These distractions and inattention might cause failure to appropriately and timely respond to potentially dangerous events while driving, thus being necessary to study different interventions to mitigate these adverse effects of non-driving related tasks (NDRT) and drowsiness. Unfortunately, their impact on driving performance and their contribution to car crashes are difficult to determine because of different definitions used to classify accidents and because not all safety-critical events end up in a car crash. Anyway, distraction and inattention might be the cause of 10% to 25% of accidents, which could have been avoided with the use of appropriate warning systems and by specific drivers' training [9,10].

Experimentation allows not only to understand the mechanisms that lead drivers to these distraction episodes and inattention but also to test systems for their detection and early warning, as well as to develop instructional materials and exercises for drivers. For example, it is possible to design a better HMI to mitigate the adverse effects of distraction by investigating visual attention while driving [11].

Some studies are carried out with actual vehicles but, for safety reasons, most of them are done on simulators. Anyway, the main problem is they are limited to 30-40 participants driving actual cars in controlled areas or simulated vehicles in simple environments like rural areas [12], which might not be representative of the general population in all sort of situations.

Although simulators have been improved in detail and capability to represent complex scenarios of suburban and urban areas, their hardware requirements hinder their use in studies with larger groups of participants.

Regardless of the experimentation environment, measuring driver's distraction is complex because it depends on observations with cameras [13] and other sensors, including those that measure driving performance (steering, throttle, speed, ...) [14], and on software methods to determine the onset of the distraction events like support vector machines [14,15] and neural networks [16].

In this regard, the Insurance Institute for Highway Safety developed a rating system [17] that evaluates the safety of ADSs and their driver monitoring systems (DMS) based on how good they are at identifying driver distraction, how they alert drivers once the distraction has been identified and also whether they employ any safety measures when the driver does not react to the alerts.

There have been attempts to use driver state assessment technology to keep the driver 'in-the-loop', i.e. to keep drivers' attention to driving [18], and even to safely stop the vehicle if the drivers fail to react to systems' alerts [19], but holistic solutions must consider education, too [20].

Drivers' ability to understand partially automated driving and to know its boundaries might make them timely taking over control of the vehicles, which might avoid crashes. Note that drivers are responsible for the vehicle even in situations where automation required unreasonably rapid reactions from them [21]. Thus, apart from education, it is required that automation is validated against a broad spectrum of possible users.

A recent literature survey [22] concludes that it is easier that drivers get engaged in NDRT while driving vehicles with ADAS and ADS, that the older drivers tend to trust less on these systems than

the younger ones, and that the main solution is that drivers know better how these systems work and how to interact with them.

Hence the main problem being training/education of drivers, which requires appropriate materials and frameworks, including simulators. Following this idea, a driving simulator of an autonomous vehicle is a key element in any drivers' training/experimentation framework. Game engines are software that integrate simulation of physical environments with graphics and other features required to build a suitable driving simulator (Section 2). Achieving a great level of realism would also make the result driving games highly demanding in terms of computational power and hardware. In this work we propose a less resource-demanding simulation software with the ability to generate scenarios from actual roadmaps to compensate for the lack of realism as for players' engagement in the game.

Additionally, this "situated experimentation" allows to study the effects of the traffic infrastructure in drivers' distraction. Note that, for instance, curves make drivers be more attentive [7], that there are other factors that contribute to drowsiness than supervising ADAS like the road conditions and environment, and that drowsiness might be worse than distractions [23].

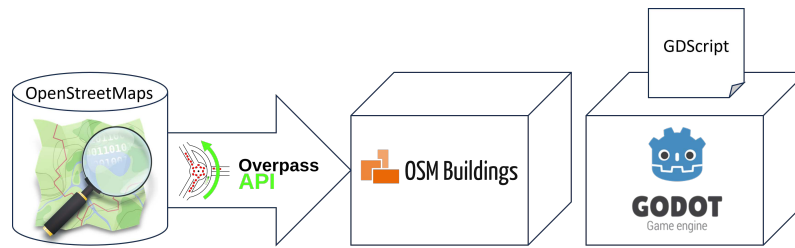
The 3D scenarios are generated from areas of interest taken from OpenStreetMap (Section 3) and the Godot game engine is used to play the simulation of an autonomous vehicle that hands the driving control to the user in different conditions while monitoring its reactions (Section 4). The result framework is a proof of concept for this software, which can be targeted to a larger number of people than with more demanding, though realistic, simulators.

## 2. Game Design and Architecture

In this work, we have created a simple driving-simulation game for users to experience driving vehicles with up to the level 3 of automation [24] and for providing researchers with a tool to study the effects distractions and investigate about systems to minimize their adverse consequences on driving.

The player drives an autonomous vehicle in a 3D world with vehicles and pedestrians roaming the streets. The vehicle goes to a random destination point in the simulated world and, at random points in time, hands the control to the user, which then must react in the shortest time possible to prevent an accident. Parameterized test scenarios allow studies of drivers' behaviors in a broad range of 3D worlds and traffic conditions. Data related to the users' interactions, specifically their awareness and reaction speed, are collected during simulation and can be further analyzed by experts to gain a better understanding of the problem, improving onboard HMI, modifying the road infrastructure, *et cetera*.

The two main components of the simulation platform are the 3D world and the game of driving a car in it, with other cars and pedestrians (Figure 1). The data to build the simulated world is taken from OpenStreetMap (OSM) [<https://www.openstreetmap.org>] and the game is run on the Godot engine [<https://godotengine.org/>] so the result platform is free to use and does not require much computational power. However, to make the simulation more appealing and immersive to the users, it is interesting to add as many details, such as non-essential objects, textures, or visual effects to the world, as possible.



**Figure 1.** The game uses the Godot engine to run a GDScript program that simulates a 3D world, extracted from an area of interest taken from OpenStreetMaps by OverPassAPI and built with OSM Buildings, wandering pedestrians and autonomous vehicles, including the primary or *ego* vehicle.

The Godot game engine is an open-source game engine that provides with all the necessary tools to create a 3D environment and scripting capabilities for all the functionality that our simulation needs. Godot's main scripting language GDScript is an easy-to-use but powerful scripting language in which to describe all kinds of 3D manipulation and implement all the functionality that the simulation might need [25].

However, in other applications, particularly for developing, training, and validating ADS, detailed 3D worlds are required. In this case, you could use, for instance, the open-source CARLA simulator [26], which is built upon Unity [<https://unity.com/>]. This simulator can also be adapted to create a framework like the one we propose in this work by making the 3D world coarser, but we have opted for Godot because of its simplicity to install and use.

OSM is a free wiki world map from which our simulation game takes the data to create a 3D world of the area of interest where players will drive the *ego vehicle*. The XML-type file extracted from OSM (".osm") [27] is further processed to build a suitable representation of the 3D world.

Differently from the main OSM API, the Overpass API [28] allows optimized reads of the OSM database that enable querying great amounts of data in a shorter time, which makes the extraction of the area of interest within our simulation framework easier. The corresponding 3D worlds are generated by OSM Buildings [29], which creates a layer of buildings rendered in 3D using data obtained from OpenStreetMap. These buildings contain all the basic metadata relating to their properties, which might be used to change shapes and aspects.

Although this aspect is out of the scope of this work, note that the simulator must record data about games and users' performance for further analysis. Anyway, the main contribution of this work is the proof of concept that a computationally inexpensive simulation game can be used both in research about drivers' distractions and inattention and in elaborating drivers' training material.

### 3. Generation of 3D Worlds

As mentioned before, the OSM 2D map data is parsed to extract all the relevant information, which then would be manipulated and used to generate all the different 3D shapes of the environment. This geometry consists of 3D meshes of roads and buildings which can be placed in the 3D world according to the coordinates from the OSM map. Additionally, a graph of the area of interest is created from the waypoints in its roads so that a specific GPS module can compute paths from current vehicle position to any given destination. These paths can be used both to direct drivers to chosen spots and to tell ADS how to move the vehicle node to node until the target position is reached, thus acting as a fully autonomous vehicle. In this last mode, the vehicle can ask the user to solve complex situations or to take the control in the vicinity of pedestrians or cars.

#### 3.1. Parsing OSM Data

OSM data can be downloaded directly from its website by choosing the region of interest, provided that the resulting file does not exceed 50000 nodes, which is enough for the purposes of this work. Anyway, this limitation is overcome by using, e.g., the Overpass API, which allows for up to 300 MB



of uncompressed downloads. These *nodes*, which should not be confused with road or graph nodes, identify geolocation data of objects, and are related to other nodes by *ways*, which, at their turn, define shapes of compound objects such as buildings or roads.

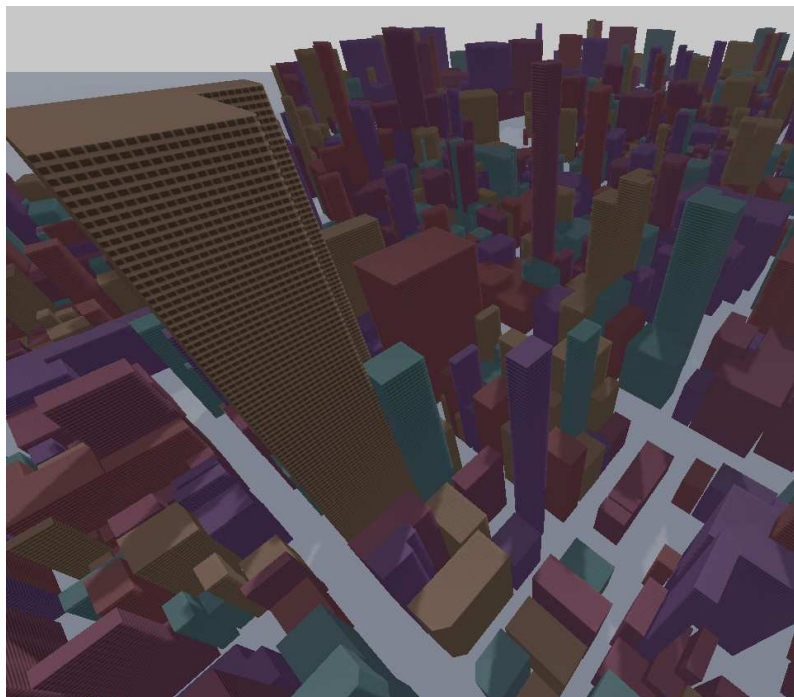
In this version of the game, only objects that are tagged as “highway” or “building” are considered. To build the 3D scene, node latitudes and longitudes are converted into scene  $(x, y)$  coordinates and other object attributes are used to create the scene objects accordingly. For instance, to distinguish between roads and sidewalks, to set the number of lanes in a road, or to construct buildings with its actual number of levels.

### 3.2. Generating 3D Meshes

The 3D scene is made of a collection of 3D meshes that are imported from arrays of vertices that define the shapes of the corresponding objects. These vertex arrays are arranged so that every three consecutive vertices make up a triangle. The final meshes that represent paths and buildings are created with the appropriate *ArrayMesh* class methods from the Godot engine’s library. Each final mesh, among other attributes, has an *ARRAY\_VERTEX* with the  $(x, y, z)$  coordinates of the vertices of the object that is added to the 3D scene of the game.

Buildings are derived from OSM file objects tagged as “building”. The area that they take is described by a set of nodes that draw a polygon which is projected in height so that a wall is added every two consecutive nodes, and a roof is created on top with the same shape than the polygon at the bottom floor.

The walls and roofs of buildings are decomposed into triangles so create the mesh for the whole. While the walls are rectangular and easily split into two triangles, the roofs are partitioned into triangles by using a specific method of the Godot’s *Geometry* class. Thus, each building is a set of triangles stored in a vertex array which translates to an instance of a mesh through *ArrayMesh* class’ object creation method, and the corresponding object is added to the 3D scene.

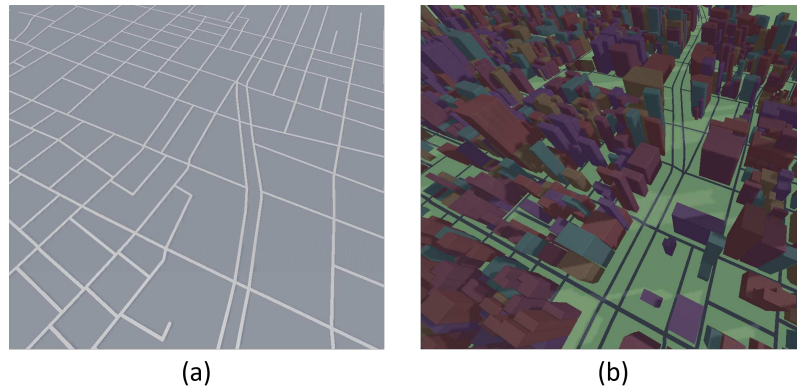


**Figure 2.** The meshes of buildings in the region of interest are 3D projections of polygons that represent their layouts.

Paths are represented as sequences of rectangles between any two consecutive nodes and each rectangle is divided into two triangles so to generate a mesh object in the same way as for the buildings.

Pathway meshes (Figure 3a) are combined with building meshes to get a basic 3D world (Figure 3b) which can be added details afterward.

Note that paths are created independently of buildings. In the current version, buildings that are crossed by paths act as obstacles, but they will be modified to create tunnels so that vehicles can go through them.



**Figure 3.** The objects that represent paths (a) are combined with the buildings to create a basic 3D world (b).

### 3.3. Road graph

The ego vehicle and the rest of vehicles in the game must be able to move autonomously. In this case, actual ADS cannot be used because the 3D world is not detailed enough for sensors to produce realistic data, and because it will take too much computational power. Therefore, it is replaced by a simpler path-following mechanism that will use a sequence of nodes of the *road graph*.

This road graph is made of all nodes of the *drivable* path elements in the OSM file of the region of interest. A path is drivable when is of type “highway” and the value associated with this type allows vehicles drive through. Nodes of the graph are connected when they are consecutive in a path. The resulting edges are tagged with the distance between their end nodes, so that the simulated GPS can use them to generate the shortest paths from the vehicle’s current node to the destination.

## 4. Game Setup

The user can select which area it is interested in and run the game to drive a vehicle manually or in a partial or fully autonomous mode. In the two last cases, the driver must choose a destination. In the first case, the driver may also choose a destination and get directions from the user interface. In fact, the HMI simulates a GPS module so the users can pick up destinations within the area of interest and, as for the manual mode, it lets them to drive the vehicle.

Therefore, the key element of the game is an autonomous *ego vehicle* provided with an HMI that includes a simulated GPS module. The other vehicles in the 3D world are fully autonomous and move around going to random destinations. Additionally, there are pedestrians that might cross streets, eventually. These elements altogether enable mimicking real-life situations to test and evaluate the driver’s attentiveness, namely cases where he or she avoids accidents and/or takes over the control of the ego vehicle.

Although the game stores user reaction times for further analyzes, it also may offer immediate feedback to the user by showing reaction times and scoring its overall behavior.

In the following we shall describe all these components of the game.

### 4.1. Vehicles

All vehicles share the same 3D model [30], which not only includes an external view (see top right block in Figure 4) but also a dashboard (see Figure 5). The body of the vehicle is defined as an

specific class, *VehicleBody*, within Godot. This class is a sub-class of the *RigidBody* class, whose objects are controlled by the physics engine to simulate the behavior of physical objects, thus they can collide to each other and react to physical forces.

In addition to the vehicle's main body, the *VehicleBody* class defines the wheels of the vehicle. The front wheels are used for steering and the rear wheels for accelerating and braking. Each pair of wheels comes with a set of variables that can be accessed and manipulated in order achieve the desired movement: Variable *steering* have values that range from  $-1$  to  $+1$  that determine the orientation of the front wheels, and variables *engine\_force* and *brake* are positive values that determine the acceleration and the braking of the vehicle.

The *engine\_force* determines how much torque is applied to the rear axle in each cycle of Godot's physics engine in terms of the *throttle* position and current rotational speed  $w$  in RPM:

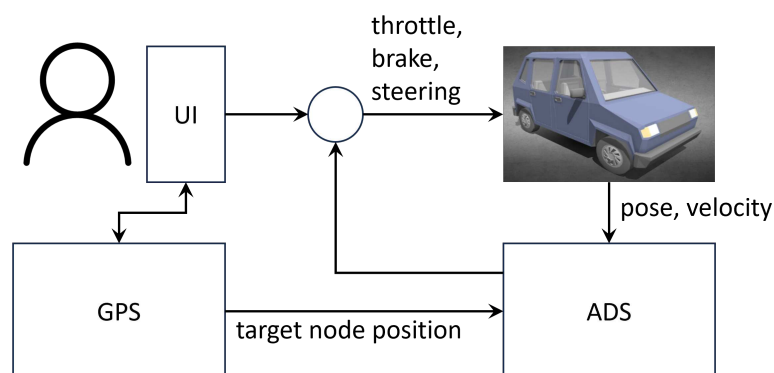
$$engine\_force = throttle \cdot \tau_{max} \cdot \left(1 - \frac{w}{w_{max}}\right) \quad (1)$$

where  $\tau_{max}$  is the maximum torque that can be applied by the engine to the wheels' axle and  $w_{max}$ , the maximum speed in RPM. Thus, increasing the value of the *throttle* position will result in the rear wheels turning faster and the vehicle speeding up towards the direction it is facing to.

The *brake* determines how much force is applied to oppose to the wheels movement so that the vehicle slows down and, eventually, stops.

The steering of the vehicle is controlled using the *steering* variable, whose changes are smoothed to make vehicles turning progressively, thus becoming easier to control for users.

Therefore, the vehicle's pose and linear and angular speeds are controlled by *throttle*, *brake* and *steering* variables, whose values can be given by either by the ADS (all vehicles) or the HMI (only the ego vehicle). Indeed, all vehicles share the same kinematics model and control mechanism (Figure 4, right) and have a simple ADS to be able to follow paths from the road graphs. The difference of ego vehicle with respect to the rest is that it also has an HMI for the user to interact with it (Figure 4, top left), including the capability to define destinations spots. (The rest of vehicles are fully autonomous and select random destinations to move around.)



**Figure 4.** All vehicles include a GPS and an ADS module to be able to follow paths to user-defined or random target positions, and only the ego vehicle needs an user interface (UI) module.





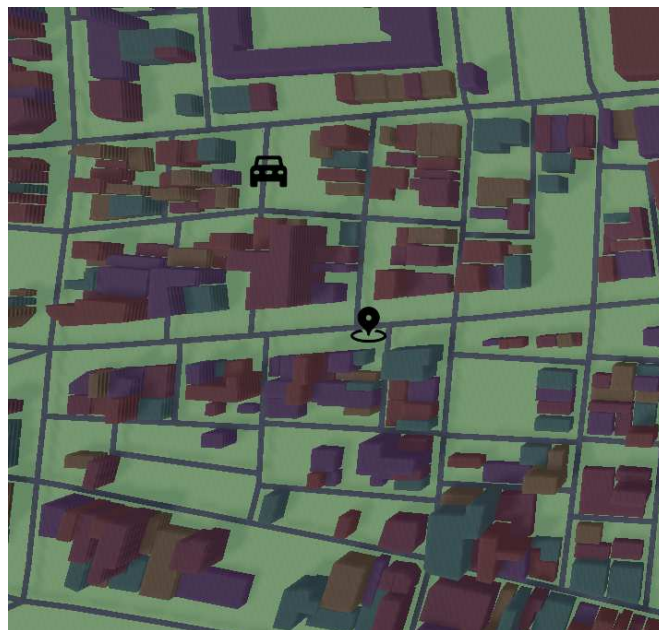
**Figure 5.** Cars are simple to make rendering computationally inexpensive, including the ego vehicle, which can be adapted to the needs of any specific experiment.

#### 4.2. GPS system

The GPS module is a simplified version of an actual GPS of a vehicle that allows the driver to monitor its position and to specify destinations. For this, a 2D view of the world is obtained from a zenith camera on a flattened 3D world, which is a copy of the 3D scene where the heights of buildings are scaled down to prevent the taller ones from shadowing or hiding the roads. On top of this view, there is a position icon at the vehicle's current location and, eventually, another one at the destination point (see Figure 6.)

When the user chooses a destination in the 2D map, this module computes the path in the road graph to reach it by using the A\* algorithm [31] that is available via the *AStar* class that comes with Godot's default library. (Unfortunately, the ID of the nodes for this class is 32-bit long while the OSM node IDs are 64-bit long, thus wrapper methods that make the conversion between 64-bit and 32-bit IDs have been added.)

Once a route has been calculated, a guiding arrow pointing towards the right direction is projected in the display when the car is driven in manual mode. In case of autonomous driving, the vehicle follows the path to the destination with eventual driver requests to hand over the control.



**Figure 6.** GPS 2D view, with icons signalling current vehicle position and destination.

#### 4.2.1. Vehicle User Control

The HMI is designed to make the game usable with any computer, even without (gaming) steering wheels and pedals, so the user input recognizes WASD keys (arrow keys, too) as well as the space bar to steer, accelerate and brake the ego vehicle.

Note that the user control works seamlessly with the ADS component in order to allow the user to switch between manual and autonomous driving modes in a smooth manner at will, apart from those events that cause ADS requiring the driver taking control of the vehicle as soon as possible.

#### 4.2.2. Vehicle ADS

The ADS takes charge of assisting the driver to follow a predefined route or making the vehicle doing so in autonomous mode. To do so, it constantly moves the vehicle forward while steering it towards the next node position in the path. To make sharper turns, it brakes the vehicle when it is turning. It also keeps the vehicle at the right side of the road.

To follow a path, the ADS goes node to node and computes the vehicle orientation with respect to the target node in the current arc. Depending on the driving mode, it only informs the user about the direction to take or applies the right amount of steering directly. The *steering* is the  $y$  component of the resulting vector of the cross product between the forward vector  $\vec{P}$  of the vehicle and the target direction vector  $\vec{T}$ :

$$steering = (\vec{P} \times \vec{T})_y \quad (2)$$

Vectors  $\vec{P}$  and  $\vec{T}$  are computed from the normalization of the current linear velocity vector, and from the difference between the position of the next target node in the path the vehicle is following and the current position of the vehicle in 3D space.

Because node positions are the waypoints at the middle of the roads, they must be offset to the right to achieve that vehicles drive in the right side of roads. Therefore, for a given path to be followed, the position of all the nodes is shifted by the normal vector of each path segment. The length of this normal vector can be changed to determine how far from the right the vehicle will drive and could be configured to make it drive at a specific lane in multiple-lane roads. In this work, however, all roads are considered to have a single lane in both directions and the vehicle will drive at the center of the right lane.

#### 4.3. Non-User-Controlled Vehicles

Apart from the ego vehicle, the game scene is populated with non-user-controlled vehicles at random spots. These vehicles simulate regular traffic on the roads and create situations where user's attention and reaction speed can be assessed. As previously mentioned, they have the same model and components than the ego vehicle but for the user interface and the working modes of GPS and ADS.

In non-user-controlled vehicles, at the beginning of the simulation or at any time a destination is reached the ADS chooses a new random destination, asks the GPS for a path to it and starts following the new route.

The proximity sensors of these vehicles are used by the ADS's to brake or stop before an obstacle in the way. They are simulated by collision shapes in front of the vehicles that trigger events any time physical entities collide or stop colliding with them. For the sake of simplicity, vehicles fully stop when they detect the ego vehicle entering their collision shape to prevent from hitting it. For the rest of the entities, i.e. other vehicles or pedestrians, they slow down and eventually stop. In this work, we have not included this behavior in the ego vehicle, but it could be added to simulate an ADS with a collision avoidance system (CAS) with an autonomous emergency braking (AEB).

#### 4.4. Pedestrians

Pedestrians are included in the simulated scene to make it more realistic and have an additional source of situations to test user reactions and behavior at the wheel. Like the vehicles, they are initially placed at random spots of the scene, at sidewalks. These spots are determined by offsetting node positions, which are at the middle of the roads, either to left or right side, using their widths.

Like the non-user-controlled vehicles, pedestrians move around randomly from their current node positions to a random neighbor node. Unlike them, they do not follow any path or have any specific destination, thus their behavior looks different to the users. They roam around their initial places so that their distribution in the environment is quite even, without forming clusters.

#### 4.5. Chunk System

Populating the 3D world with vehicles and pedestrians may affect performance of the game when their number grows to the thousands. Fortunately, only those in the vicinity of the ego vehicle, at the possible sight of the player, are required at a given moment. Therefore, the game groups entities into chunks so that only those nearby the ego vehicle are simulated.

The *chunk system* is implemented as a dictionary so that chunk coordinates are the key to access a class object that holds the list of all the entities that are inside this chunk in addition to a variable that tells us whether this chunk is active or not. (Note that it could include any other entity besides vehicles and pedestrians.)

Chunks are rectangular cells of a grid that covers all the area of the 3D scene thus, entities belong to the chunk corresponding to their coordinates divided by the width and height of the cells. The cell size can be modified but should not be smaller than the size of a city block or too large so that near all entities are simulated at each simulation cycle.

The activation or deactivation of a chunk depends on the distance from the user's coordinates. In the first case, all entities of the chunk are made visible to the user and their execution is resumed, while deactivation of chunks imply the opposite operations, namely, stopping the execution of the behaviors of the affected entities and their rendering too, thus saving CPU time and increasing performance of the simulation.

### 5. Experimentation

The software is publicly available through GitHub [[https://github.com/bemo10/Autonomous\\_Vehicle\\_Simulation](https://github.com/bemo10/Autonomous_Vehicle_Simulation)] and runs on MS Windows platforms with the Godot engine (version 3.5 stable), and it can be modified freely. In the folder of the project there is a *README.md* file with information about how to select a map for the simulation and the user controls when running the *Autonomous Vehicle Simulation.exe* program.

By default, the simulated worlds are assumed to come from areas not larger than 50,000 OSM nodes and are populated with 10,000 pedestrians and 3,000 vehicles, though only those close to the ego vehicle will be active, thanks to the chunk system.

The performance of the simulation depends directly on the time to render the scene and, particularly, on the number of active entities at each cycle, which is related to the size of the chunks. Table 1 shows how the average frame time increases with the number of active pedestrians and vehicles on a Windows 10 Home 64-bit computer with 16 Gb DDR4 1600 MHz RAM, an AMD Ryzen 5 3600 6-core CPU and a NVIDIA GeForce RTX 3060 Ti 8Gb GDDR6 GPU.

**Table 1.** Average number of frames per second (FPS) with respect to active pedestrians and vehicles.

Active Pedestrians (Avg.)	Active Vehicles (Avg.)	Frame Time [ms]	FPS
6	13	8.73	114
20	17	9.64	103
30	25	22.74	43
40	15	24.04	40

5.1. User Interaction Assessment

The user attentiveness and reaction speed are measured when a potentially hazardous situation arises, and he or she must take over the control of the vehicle. These situations appear when the ego vehicle heads against another vehicle or a pedestrian. In the last case, note that, in addition to the regular behavior of walking at the sidewalks, pedestrians may eventually cross streets.

The game starts when the user chooses a destination, and the vehicle starts driving autonomously to that location. The driver must pay attention to other vehicles and to the pedestrian movements to regain control of the vehicle whenever any risky situation may take place. These cases are randomly fired at specific points of the journey and, for each case, the delay in taking over control and the outcome are recorded.

The game ends when the driver gets to the destination, and a report with the data of how has he or she managed each situation and a final score for that particular journey is displayed and, eventually, stored for further analyses.

As said, the game measures the time it takes to the driver to get the control of the vehicle from the moment a potential accident may occur until the moment that some manual input is done. Additionally, it scores the outcome of the situation, with positive values when nothing happens and negative ones otherwise. In the current version, the scoring system is simple: it adds  $40/t_r$  points for each avoided accident ( $t_r$  is the reaction time) and subtracts 50 on car crashes or hitting pedestrians. (Note that, for the first cases, reaction times are also recorded.)

Table 2 shows a small example of a user test with 4 interactions along a short journey, which scored 145.9 points. Note that car crashes are assumed not to stop the travel, so to be able to complete the route and the test.

**Table 2.** Simulation record of a short journey with 4 events.

Interaction with ...	Outcome	Reaction Time [s]	Points
Vehicle	Accident avoided	0.506	+79.1
Pedestrian	Accident avoided	0.701	+57.1
Vehicle	Accident occurred	N/A	-50.0
Pedestrian	Accident avoided	0.670	+59.7

6. Discussion and Conclusion

Paradoxically, the more the technology provides us with safety while traveling in a vehicle, the easier drivers get distracted or pay less attention to driving. These phenomena are studied to develop technology that prevents their adverse effects from taking place. Notably, some studies conclude that one of the best strategies is drivers’ education, which includes acquiring knowledge about ADAS and ADS and training.

Studies and onboard technology (ADAS, ADS, DMS, HMI, ...) development and validation require experimentation with drivers. While experimentation with actual vehicles and realistic simulators are still required, the only way to make large experiments and reach a broader audience is by using software than can be run in affordable platforms.

In this work, we prove that technology is mature enough to develop such type of software by using open-source tools such as OpenStreetMap and Godot game engine. The result simulation framework [[https://github.com/bemo10/Autonomous\\_Vehicle\\_Simulation](https://github.com/bemo10/Autonomous_Vehicle_Simulation)], though simple, can create a 3D world from a real region of interest taken, at its turn, from OSM, and of let users drive a PAV in it, with other autonomous vehicles and pedestrians moving around and causing situations that make the ego vehicle ADS requiring them to take over the control.

A basic scoring system is used to foster players to minimize their reaction times and to provide a simple example of how the framework can be used to train drivers to face these situations. Anyway, it can be hidden or modified to adjust it to the needs of any study or learning objectives, as well as the types and frequency of situations drivers' run into.

The software simulator reduces its computational power requirements by simulating only the objects that are in the chunks close to the ego vehicle and, therefore, it is possible to include other classes of objects and, particularly, traffic lights or signs. Additionally, the 3D scene can be enriched with non-essential details to get a better level of realism. This option, however, should be included without compromising simulation execution performance.

With this development, we believe that it is possible to use this kind of "light" simulation games to make studies with a larger number of participants and, though results might not be as accurate as with a more realistic platform, they would be more representative of the population. And, last but not least, they might be more useful as educational tools (e.g. in driving schools) and as complementary tools for psychometric tests.

**Author Contributions:** Conceptualization, L.I.R.; methodology, M.B.; software, M.B.; validation, M.B. and L.I.R.; formal analysis, L.I.R.; investigation, L.I.R.; writing—original draft preparation, M.B. and L.I.R.; writing—review and editing, L.I.R.; supervision, L.I.R. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Data sharing not applicable. (Software is publicly available at [https://github.com/bemo10/Autonomous\\_Vehicle\\_Simulation](https://github.com/bemo10/Autonomous_Vehicle_Simulation).)

**Conflicts of Interest:** The authors declare no conflict of interest.

## Abbreviations

The following abbreviations are used in this manuscript:

ADS	Advanced driving system
ADAS	Advanced driver-assistance system
AEB	Autonomous emergency braking
API	Application programming interface
CAS	Collision avoidance system
CPU	Central processing unit
DMS	Driver-monitoring system
GPS	Global-positioning system
HMI	Human-machine interface
NDRT	Non-driving related tasks
OSM	OpenStreetMap
PAV	Partially automated vehicle
RPM	Revolutions per minute



## References

1. Kaur, P.; Taghavi, S.; Tian, Z.; Shi, W. "Survey on Simulators for Testing Self-Driving Cars." *arXiv*, 2021, doi: 10.48550/arXiv.2101.05337.
2. Thorslund, B.; Thellman, S.; Nyberg, V.; Selander, H. "Simulator-based driving test prescreening as a complement to driver testing – Toward safer and more risk-aware drivers." *Accident, Analysis & Prevention*, vol. 194, 2024, ISSN 0001-4575, doi: 10.1016/j.aap.2023.107335.
3. Sánchez, J.S.; Ortiz, J.S.; Mayorga, O.A.; Sánchez, C.R.; Andaluz, G.M.; Bonilla, E.L.; Andaluz, V.H. "Virtual Simulator for the Taking and Evaluation of Psychometric Tests to Obtain a Driver's License." In: De Paolis, L.; Bourdot, P. (eds) *Augmented Reality, Virtual Reality, and Computer Graphics*. AVR 2019. *Lecture Notes in Computer Science*, vol 11613. Springer, Cham., 2019, doi: 10.1007/978-3-030-25965-5\_11.
4. Kaça, G.; İzmitligil, T.; Koyuncu, M.; Amado, S. "How well do the traffic psychological assessment systems predict on-road driving behaviour?" *Applied Cognitive Psychology*, 35(5), 1321–1337, 2021, doi: 10.1002/acp.3867.
5. Strayer, D.L.; Watson, J.M.; Drews, F.A. "Cognitive distraction while multitasking in the automobile." Chapter 2 of *Advances in Research and Theory, Psychology of Learning and Motivation*, vol 54, Elsevier 2011, doi: 10.1016/B978-0-12-385527-5.00002-4.
6. Zangi, N.; Srouf-Zreik, R. Ridet, D.; Chassidim, H.; Borowsky, A. "Driver distraction and its effects on partially automated driving performance: A driving simulator study among young-experienced drivers." *Accident Analysis & Prevention*, vol. 166, 2022, doi: 10.1016/j.aap.2022.106565.
7. Dengbo, H.; Dina, K.; Birsén, D. "Distracted when Using Driving Automation: A Quantile Regression Analysis of Driver Glances Considering the Effects of Road Alignment and Driving Experience." *Frontiers in Future Transportation*, vol. 3, 2022, doi: 10.3389/ffutr.2022.772910.
8. Lambert, F. "Tesla Autopilot results in decreased driver attention, new study finds." Electrek. Sep. 13, 2021. Available online: <https://electrek.co/2021/09/13/tesla-autopilot-decreases-driver-attention-new-study/>.
9. Regan, M.A.; Hallett, C. "Driver Distraction." *Handbook of Traffic Psychology*, 2011.
10. "Driver distraction 2015." Report by the European Road Safety Observatory, 2015.
11. Chen, W.; Liu, W. "HMI Design for Autonomous Cars: Investigating on Driver's Attention Distribution." In: Krömker, H. (eds) *HCI in Mobility, Transport, and Automotive Systems. HCII 2019. Lecture Notes in Computer Science*, vol 11596. Springer, Cham. 2019, doi: 10.1007/978-3-030-22666-4\_7.
12. Papantoniou, P.; Papadimitriou, E.; Yannis, G. "Assessment of Driving Simulator Studies on Driver Distraction." *Advances in Transportation Studies*, vol. 35, 2015, pp 129-144.
13. Fernández, A.; Usamentiaga, R.; Carús, J.L.; Casado, R. "Driver Distraction Using Visual-Based Sensors and Algorithms," *Sensors*, 16, 1805, 2016, doi:10.3390/s16111805.
14. Jin, L.; Niu, Q.; Hou, H.; Xian, H.; Wang, Y.; Shi, D. "Driver Cognitive Distraction Detection Using Driving Performance Measures." *Discrete Dynamics in Nature and Society*, Hindawi Publishing Corp., 2012, doi: 10.1155/2012/432634.
15. Ahangari, S.; Jeyhani, M.; Rahman, M.M.; Dehzangi, A. "Predicting driving distraction patterns in different road classes using a support vector machine." *International Journal for Traffic and Transport Engineering*, 2021, doi: 10.7708/ijtte.2021.11(1).06.
16. Srinivasan, K.; Garg, L.; Datta, D.; Alaboudi, A.A.; Jhanjhi, N.Z.; Agarwal, R.; and Thomas, A.G. "Performance comparison of deep CNN models for detecting driver's distraction." *Computers, Materials & Continua*, vol. 68, no.3, pp. 4109–4124, 2021, doi: 10.32604/cmc.2021.016736.
17. *IIHS creates safeguard ratings for partial automation*. Insurance Institute for Highway Safety (IIHS), Jan. 20, 2022. Available online: <https://www.iihs.org/news/detail/iihs-creates-safeguard-ratings-for-partial-automation>
18. Karatas, N.; Yoshikawa, S.; Tamura, S.; Otaki, S.; Funayama, R.; and Okada, M. "Sociable driving agents to maintain driver's attention in autonomous driving." *2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 2017, pp. 143-149, doi: 10.1109/ROMAN.2017.8172293.
19. Perrone, A. "How Automakers Plan to Keep Drivers Attentive in Self-Driving Cars." *Endurance*, March 08, 2019. Available online: <https://www.endurancewarranty.com/learning-center/tech/keep-driver-attention-self-driving-cars/>.
20. Cunningham, M.L.; and Regan, M.A. "Driver distraction and inattention in the realm of automated driving." *IET Intell. Transp. Syst.*, 12: 407–413. 2018, doi: 10.1049/iet-its.2017.0232.

21. Beckers, N.; Siebert, L.C.; Bruijnes, M.; Jonker, C.; and Abbink, D. "Drivers of partially automated vehicles are blamed for crashes that they cannot reasonably avoid." *Scientific Reports*, vol. 12(1), 2022, doi: 10.1038/s41598-022-19876-0.
22. Hungund, A.P. "Systematic Review of Driver Distraction in the Context of Advanced Driver Assistance Systems (ADAS) & Automated Driving Systems (ADS)." *Masters' Theses*, 1243. 2022, doi: 10.7275/30956149.
23. Miller, D.; Sun, A.; Johns, M.; Ive, H.; Sirkin, D.; Aich, D.; and Ju, W. "Distraction Becomes Engagement in Automated Driving." *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 59(1):1676-1680, September 2015, doi: 10.1177/1541931215591362.
24. *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*. Report J3016\_202104. Society of Automotive Engineers (SAE), 2021. Available online: [https://www.sae.org/standards/content/j3016\\_202104/](https://www.sae.org/standards/content/j3016_202104/)
25. Linietsky, J.; Manzur A.; and the Godot community. "Godot Version 3.5 documentation." Available online: <https://docs.godotengine.org/en/stable/index.html>
26. Dosovitskiy, A.; Ros, G.; Codevilla, F.; López, A.; and Koltun, V. "An Open Urban Driving Simulator," *Proceedings of the 1st Annual Conference on Robot Learning*, 1-16, 2017. Available online: <https://carla.org/>
27. The OpenStreetMap wiki. Available online: [https://wiki.openstreetmap.org/wiki/Main\\_Page](https://wiki.openstreetmap.org/wiki/Main_Page)
28. An API for read-only queries of OSM data Available online: <http://overpass-api.de/>
29. 3D layer of buildings. Available online: <https://osmbuildings.org/>
30. *scailman*. *Low-Poly Small Car*. Sketchfab. Available online: <https://skfb.ly/6QZtA>
31. *A\* search algorithm*. *Wikipedia*. Available online: [https://en.wikipedia.org/wiki/A\\*\\_search\\_algorithm](https://en.wikipedia.org/wiki/A*_search_algorithm).

**Disclaimer/Publisher's Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.