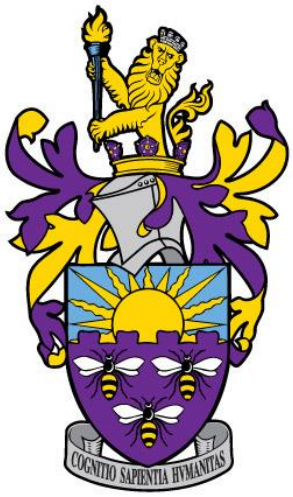
UNIVERSITY OF MANCHESTER

# Developing a legal advisor agent for self-autonomous cars




by
Sebastian Bartuzi

MEng in Artificial Intelligence
University of Manchester
Department of Computer Science

Supervisor: Louise Dennis

Word count: 10,172

# Abstract

*Past years have seen significant changes in self-driving vehicles industry. Assistance features in cars are getting more advanced and therefore cars become more autonomous. UK roads could see first self-driving cars on roads in 2025. In this project I focused on developing a driving legal assistant that would analyse road conditions and driver's intentions in order to return a proper set of law recommendations that should be followed in different situations. I developed a high-quality 3D simulator based in Unity which enables users to fully control the car and see recommendations displayed on the screen. The driving assistant is supported by The Rules of The Road Advisor (RoTRA) tool. Evaluation performed on a group of people has proved simulator's user-friendliness and reliability of given advice. The assistant performs very well and could be extended to give more specific guidance and implemented in autonomous vehicles to make them drive legally and safely.*

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1. Context and motivation

Recent years have seen a sharp rise in discussions about autonomous vehicles. As more AV car prototypes successfully pass test phases and streets see robots delivering groceries to houses (for instance in Leeds or Trafford, UK), more discussions and parliamentary debates concern this topic. People realise this is a revolutionary technology that will completely transform their lifestyles, so they want to be absolutely sure it is safe to use.

One of the key challenges in the area is to make autonomous cars follow proper highway rules. It is estimated that a driver makes between 50 and 100 decisions per mile [1] and sometimes they make bad choices. The challenge in AVs development is to reflect this decision-making process (obviously, with mistakes elimination) to obtain a procedure in which intelligent cars make right decisions on how to behave in different road situations. A system that can support this process is RoTRA (The Rules of The Road Advisor), a tool developed by Joe Colenette [2]. It is the UK Highway Code subset written into a large set of Prolog rules.

The most crucial part when making a decision is to properly understand the situation around and your own state, as well as define the intended next steps. Based on these, you should be able to make a right decision. And this is how drivers operate: they analyse the situation and identify their plans, and then decide on actions. RoTRA follows the same way of thinking as it requires independent sets of beliefs and intentions to call a rule.

The arising question is whether this system could effectively support human drivers. How accurate could a system that tells drivers what to do on every step be? What would be people's reactions to such system and what would their feedback say about possible enhancements to AVs legal advising tools?

In this project I developed a simulator for a car with RoTRA-based legal advisor system that answers the questions above. The simulator is focused on studying human's reactions to it rather than developing a fully autonomous car, therefore users are in a full control of the vehicle.

The evaluation of the project has been conducted by sharing with a number of people my simulator and asking them to fill in a Google Forms questionnaire. It has checked user's feelings and experience on the simulator, which later allowed me to perform analysis of the results to check for bugs and proposed enhancements, but most importantly, to conclude on whether such a legal advisor system could be helpful for human drivers. Additionally, suggestions of enhancements have been discussed and an overview of possible further work to implement my agent in autonomous vehicles has been presented.

## 1.2. Aims and objectives

Following from the motivation, the project has aims and objectives stated below:

(1) To explore further the subject of autonomous vehicles, paying particular interest to the safety of such devices and existing assistance features.
(2) To develop a 3D Unity-based simulator which enables a user to freely control the car.
(3) To enrich the car with methods that enables it to properly analyse the environment.
(4) To convert the environment analysis into a set of law recommendations by establishing communication between Unity and RoTRA.
(5) To perform a user evaluation which will check if the simulator is user-friendly and if the recommendations displayed are relevant.

## 1.3. Report structure

The report is built with six chapters:

(1) **Chapter 1** contains an introduction to the project.
(2) **Chapter 2** gives an insight into the theoretical background behind the project.
(3) **Chapter 3** presents methodologies and the design of the simulator.
(4) **Chapter 4** presents results and conclusions of the project's user evaluation.
(5) **Chapter 5** describes the project's development phase.
(6) **Chapter 6** is a final summary of the project.

# Chapter 2

# Background

This chapter presents the theoretical background of the project. It explains the basics of RoTRA and gives an insight into the topic of autonomous vehicles.

## 2.1. Autonomous vehicles

As AV industry and policies around it are fast-changing, the information in this chapter is relevant as of 31st March 2023.

### 2.1.1. Overview

Autonomous vehicles (AVs), also known as self-driving cars or driverless cars (an example of such a car is shown in Fig. 1), are a revolutionary technological advancement in the automotive industry. The main goal of AVs is to provide safer, more efficient, and more convenient transportation. These vehicles are designed to navigate through roads and highways without the need for human control or intervention. Instead, they rely on advanced sensors, cameras, and other cutting-edge technologies to perceive the environment, analyse data, and make informed decisions. They work together to create a detailed, 360-degree map of the vehicle's environment, allowing it to detect and avoid obstacles, pedestrians, and other vehicles. Additionally, autonomous vehicles are equipped with advanced algorithms that can analyse and interpret this data in real-time. This allows the vehicle to make split-second decisions about how to navigate the road ahead safely and efficiently.

In more detail, AVs function within constant “Sense – Plan – Act” (SPA) cycle [3]:

- **Sense:** analyse the environment using in-built sensors.
- **Plan:** prepare the next action.
- **Act:** perform planned action.

In the simulator it is the user that fully operates the car, so the step Act is replaced with a step that could be called Recommend: instead of taking over the vehicle, information on what action must or should be taken will be returned.

Definitions on autonomy vary, so multiple taxonomies were created to differentiate between different levels of vehicle’s self-control [4]. The most common division is the one proposed by SAE International (2014) [5]:

- **Level 0: No driving autonomation.** Driver has a full control over the car, however, the vehicle may have some assistance features.
- **Level 1: Driver Assistance.** The driver has to constantly control and monitor the car, although assistance features may support them in actions like steering or accelerating.
- **Level 2: Partial Driving Automation.** These cars are equipped with Advance Driving Assistance Systems (ADAS), which may take over control on some of car’s functions, however, the driver still needs to be fully monitoring the vehicle.

- **Level 3: Conditional Driving Automation.** This type of cars is able to take over full control, although the driver is required to constantly monitor the road situation and be ready to take control at any point.
- **Level 4: High Driving Automation.** The technology is fully in charge, however, the driver can take control at any point if needed.
- **Level 5: Full Driving Automation.** The car is fully self-controlled and the driver is not able to control it as it is not equipped with human controls.

**Fig. 1:** An example of a self-driving car.

In terms of the simulator, the user is in full control of the car, thus, we can classify this type of vehicle to Level 0. However, the project's aim is not to increase the autonomy level by building a fully self-controlled car, but rather concerns developing a legal assistant that would support such vehicles.

For the revolution of autonomous cars to take place in the automotive industry, it is essential that the general public has complete trust in the safety and reliability of the technology. Strict safety requirements must be met before any further advancements can be made in legislation. Advancements in technology must be paired with a comprehensive understanding of the risks and benefits of autonomous driving to create an effective regulatory framework that ensures the safety of all road users. The following subchapters take a closer look at these pressing issues.

## 2.1.2. Legal status

The history of first ever, although unsuccessful attempts to develop driverless technologies, reaches 1920s, however, it was in 1980s when pioneer pilots were first rolled-out [6]. The most rapid pace of developments in the area is taking place now.

Currently, although self-driving cars are allowed in the UK for commercial use, there are no such vehicles listed by The Secretary of State for Transport [7]. Under the current UK government plans driverless cars will be legally approved on British roads within the next year [8] with an aim to have self-driving cars fully operational on streets by 2025. The 1st July 2022 update to The UK Highway Code (more on the Code in chapter 2.2.1) has introduced the term of *self-driving vehicles* as vehicles that are *capable of safely driving themselves when the self-driving function is correctly turned on and the driver follows the manufacturer's instructions*. The Code points out that despite car's autonomy, the driver must be fit to drive and, although they can turn their attention away from the road situation, they must not hold a hand-held device (as rule 149 to not do anything illegal applies) [9]. Apart from that, autonomous vehicles just like standard cars must be roadworthy and road legal [10].

Within government plans it is to implement *a robust regulatory framework* to popularize autonomous vehicles and services [11]. These plans are backed with funds of £100 million, including £34 million dedicated to a wider safety research. The government estimates the industry will create 38,000 jobs and might be worth up to £42 billion [12].

Despite not having a full legislation for self-driving cars in place, British streets can see a range of automated machines moving along streets. For instance, Co-op and Starship Technologies test their grocery delivery robots in Leeds [13] and Greater Manchester [14] (shown in Fig. 2).


**Fig. 2:** Grocery delivery robots developed by Co-op and Starship Technologies.

### 2.1.3. Safety

When it comes to public debates about AVs, their safety is the most popular topic to argue about. The public wants to be absolutely sure self-driving cars can be trusted before taking a seat inside or meeting such a car on their way.

According to the research conducted in 2020 by InsuretheGap only 5% of the European public would feel completely safe about using a fully autonomous car, while 69% of them state that drivers should be in control of their car at all times [15]. Another measure the public is keen on seeing is to label the self-driving vehicles so it would be easy to distinguish them from human-controlled cars, as 86% of UK public agree with this statement [16].

Self-driving cars are being tested on city streets by multiple technology companies and they often turn out to perform close to flawlessly – for instance, Nissan Leaf cars drove 1,600 miles without a single accident [17]. Studies suggest that AVs could improve the road safety by more than 90% due to human's error elimination [18].

The main fear of self-driving cars is that they may get out of control and do harm. The most popular concerns about autonomous vehicles are about interactions with other road users (pedestrians, cyclists, human-controllable cars) in congested urban environment, system malfunction or possibility of having the system hacked [19].

Those safety concerns can be beaten by constant improvements of assistance features. The project's focus is on legal advisor, and the technology behind it is RoTRA, described in chapter 2.2.2.

### 2.1.4. Assistance features

As mentioned in 2.1.1., improvements of assistance features play crucial role when giving more autonomy to a car. The legal advisor of the project will be supported by frameworks of other existing assistance features which will be explained in more detail in this subchapter.

**Forward Collision Warning** systems detect an incoming collision with a stopped or slower vehicle in front. Usually, it is implemented with a camera that, when an obstacle is found, calculates the distance to it. Proper warning is returned when time to contact ($TTC$) is less than the threshold (usually 2 seconds). It is defined by the formula:

$$TTC = \frac{Z}{v}$$

where $Z$ is the distance between objects and $v$ is car's relative velocity [20].

An assistance feature similar to Forward Collision Warning is **Collision Avoidance System**, which is equipped with the possibility to take over the control and avoid a crash [21].

**Backup camera** (shown in Fig. 3) is an ADAS feature that improves the environmental monitoring. Instead of rotating the body to see through the back screen, a driver can simply look at the small screen, usually near the controls. The visual image is often equipped with indicators that allow to monitor distance from obstacles [22].


**Fig. 3:** Backup camera.

**Traffic Sign Recognition** system is another method that assists the vehicle. As the name suggests, this system is responsible for differentiating traffic signs, usually using one of two methods:

(1) Colour thresholding, region detection and shape analysis.
(2) Border detection in a black and white image [23].

Frameworks of these driving assistants have shaped the background behind my legal advisor's methods to scan the environment, as well as behaviour behind the NPC car. More information on how they have been used in the project's development will be presented throughout chapter 3.

# 2.2. Obeying rules

## 2.2.1. UK Highway Code

Every vehicle on the road needs to follow their national rules about driving, and every country has a defined set of road rules. For instance, German road rules are detailed in Straßenverkehrsordnung (StVO) [2]. In case of the UK, these are defined in the UK Highway Code [24] with an example rule presented in Fig. 4.

This document defines *rules for motorists, cyclists and pedestrians, road and boat safety* [25]. This law is presented in a set of ordered rules for various situations with annexes about penalties for not following the code, first aid and vehicle maintenance.

Drivers are obliged to follow many of those rules and these are identified with the use of words *MUST/MUST NOT*. Failure to comply with these is a criminal offence, and in the worst cases penalties could be equal to prison sentences. In the meantime, rules that use words *SHOULD/SHOULD NOT* or *DO/DO NOT* are advisory, however, disobeying them could be used as evidence to establish liability [10].

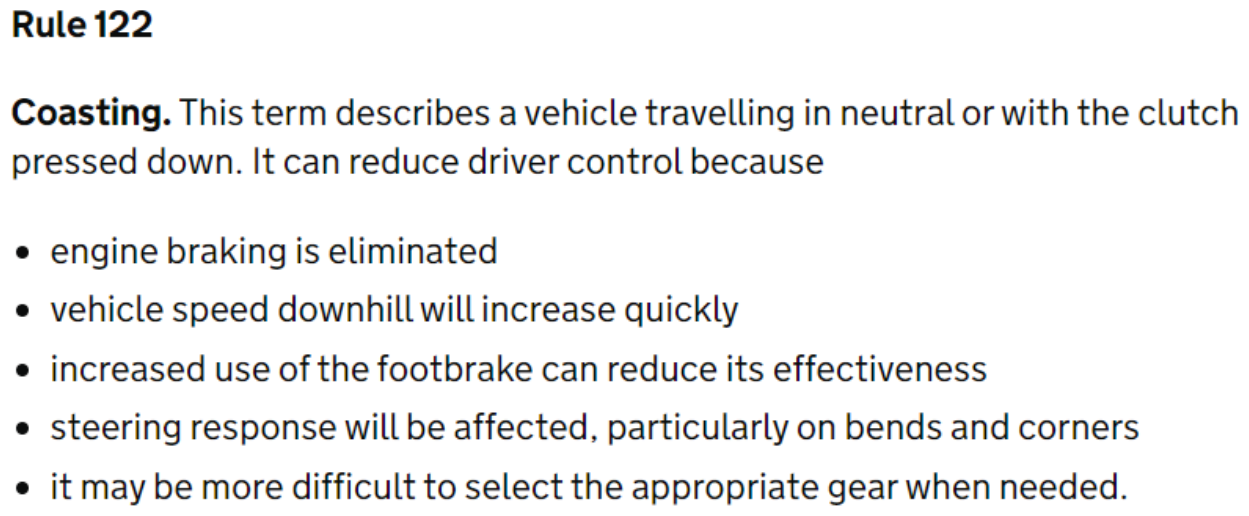



**Fig. 4:** An example rule in the UK Highway Code.

### 2.2.2. RoTRA

When considering on how to apply road rules into the legal agent, a tool that defines them in a programming language must exist. As for Straßenverkehrsordnung mentioned in the previous subchapter, a subset of these rules have been encoded in Linear Temporal Logic (LTL). But for the case of the UK, one such a candidate technology is RoTRA (Rules of The Road Advisor) [2].

It was briefly defined in chapter 1.1 as a tool that implements UK Highway Code rules. RoTRA represents only a subset of the Code, which is relevant to cars behaviour (so rules applying to pedestrians or car's driver are skipped). The rules are written in the Prolog language, and they are divided into those that *must* be followed (originally defined by *MUST/MUST* NOT) and those that *should* be followed (*SHOULD/SHOULD NOT* or *DO/DO NOT*). RoTRA is compatible with the UK Highway Code update from 14th Sep 2021 [26].

Encoding of the Fig. 4 rule is presented in Fig. 5. Since coasting may affect the driver's control, when intentions are *gearNeutral* (gear is neutral or the clutch is pressed down) and *driving* (the vehicle is on), the recommendation is *should-avoid_coasting* (the driver should avoid coasting).

```
%Rule Description: Coasting
rule(r122, standard, [gearNeutral, driving], [], [should-avoid_coasting]).
```

**Fig. 5:** Fig. 4 rule encoded in RoTRA.

RoTRA rules are encoded with both set of beliefs and set of intentions to be met. This is because road rules depend on the current state of the world (beliefs) and what the driver wants to do (intentions). Combining these two elements together, we can infer what rules should be followed in a situation.

For example, the recommendation 128a *should-cancelOvertake* (the driver should cancel overtaking) will be enforced when beliefs *doubleWhiteClosestBroken* (the relevant white line separating the lanes is broken) and *cantOvertakeBeforeBrokenSolidifies* (the driver would not be able to come back to their lane before the line solidifies), as well as intention *overtake* (the driver wants to overtake a vehicle) are met. If the belief *cantOvertakeBeforeBrokenSolidifies* is not true (so the driver would be able to finish overtaking before reaching solid double white line), the recommendation will not be returned. The same applies with *overtake*

intention – if the driver is not overtaking, no overtaking needs to be cancelled. This example shows how both beliefs and intentions are crucial in RoTRA implementation.

Some recommendations are based only on either beliefs or intentions. For instance, rule 119 contains only *skidding* (the vehicle got into skidding) belief. No matter what the driver currently wants to do, they should exit the skidding state, so recommendations *should-turn_into_skid* (turn into the direction of skidding), *should-release_brake* (release fully the brake pedal) and *should-ease_off* (ease off the accelerator) will be returned. The other way round, recommendation 117 is *should-brake_early_lightly* (brake as early and lightly as possible), activated only by *brake* (the driver wants to brake) intention. Regardless of what is going around, if the driver wants to push the brake pedal, they should always do it early and lightly.

There are also a few recommendations that do not need any beliefs and intentions to be activated (so they are relevant all the time). You have to drive cautiously and with full attention since starting to turning off the engine, so recommendations *must-not_drive_dangerously* (drive without creating any situations that could cause harm to others), *must-drive_care_attention* (drive with a full attention) and *must-consideration_others* (drive with consideration of the safety of other road users) will be true all the time.

# Chapter 3

# Design

The following chapter presents how the simulator has been developed, including technologies used, methods to read the beliefs and intentions of the car and its environment, and logics behind items in the environment.

## 3.1. Technologies

### 3.1.1. RoTRA

Theoretical aspects of RoTRA have been described in chapter 2.2.2. Original RoTRA Prolog files have been used in the project, however, as they contain the set of all UK Highway Code rules, only a subset of them has been implemented into the legal advisor. Beliefs and intentions of the undeployed recommendations have not been touched, however, the tool could be easily extended to take these rules into account as well.

### 3.1.2. Unity

Unity is an engine for games and simulators development. The simulator has been developed in Unity with the city map (its overview is shown in Fig. 6), cars, car's physics, road signs, pedestrians and traffic lights downloaded from Unity's Assets Store. It is worth mentioning that the original map was suitable for right-hand traffic system so the map's x dimension has been set from 1 to -1 (as a complication of that, it can be noticed that some signboards are mirrored). Scripts for Unity use the C# language, therefore pedestrians (PedestrianPathFollower.cs), environment's car (CarPathFollower.cs) and traffic lights (TrafficLightsController.cs) behaviours, as well as the bridge to RoTRA (ROTRAConnector.cs) have been encoded in respective C# scripts. An overview of the aim of each script is presented in Table 1.

| **File** | **Task** |
|---|---|
| ROTRAConnector.cs | The file establishes the connection between RoTRA and the simulator which is made every time either a set of beliefs or a set of intentions has changed. The file defines methods that are responsible for detecting beliefs and intentions changes. Furthermore, this file contains methods responsible for the advice display on the screen. |
| PedestrianPathFollower.cs | This file defines pedestrians' behaviours, which include path following and stopping before crossing the road when a car is approaching. |
| CarPathFollower.cs | This file defines the NPC car's behaviour, which include path following, stopping at red traffic lights and giving way to pedestrians. |
| TrafficLightsController.cs | This file defines a behaviour behind tiles covering traffic lights so the crossroads function properly. |

**Table 1:** Script files and their tasks.

**Fig. 6:** Map overview.

### 3.1.3. Communication

Unity and RoTRA communicate through command line calls in C#. The script that operates the communication makes a parallel call via command line to Prolog every time the set of beliefs and intentions changes. The list of current recommendations is updated with the call's output, and any new recommendation appears on the screen. However, calls functionality has been implemented only for Windows OS. When using any other operating system, calls to the command line will not be executed, so no recommendations will be displayed.

## 3.2. Logics of the environment

Some items in the world got equipped with a set of behaviours in order to provide user with a better real-world experience. The following section explains logics of these behaviours.

### 3.2.1. Traffic lights

The following behaviour is implemented in the TrafficLightsController.cs file.

There is one crossroad with traffic lights in the map. The traffic light A is always of the same light as the opposite light B, and the light C – identical to the light D (as shown in Fig. 7). The lights change in two phases (from phase 1 to phase 2, from phase 2 to phase 1) with both of them lasting the same amount of time, and A&B being always in the opposite phase as lights C&D.

Phase 1: for 15 seconds the light is red. After the time passes, change the light to red&amber.

Phase 2:

(1) For 2 seconds the light is red&amber. After the time passes, change the light for green.
(2) For 10 seconds the light is green. After the time passes, change the light for amber.
(3) For 3 seconds the light is amber. After the time passes, change the light for red.

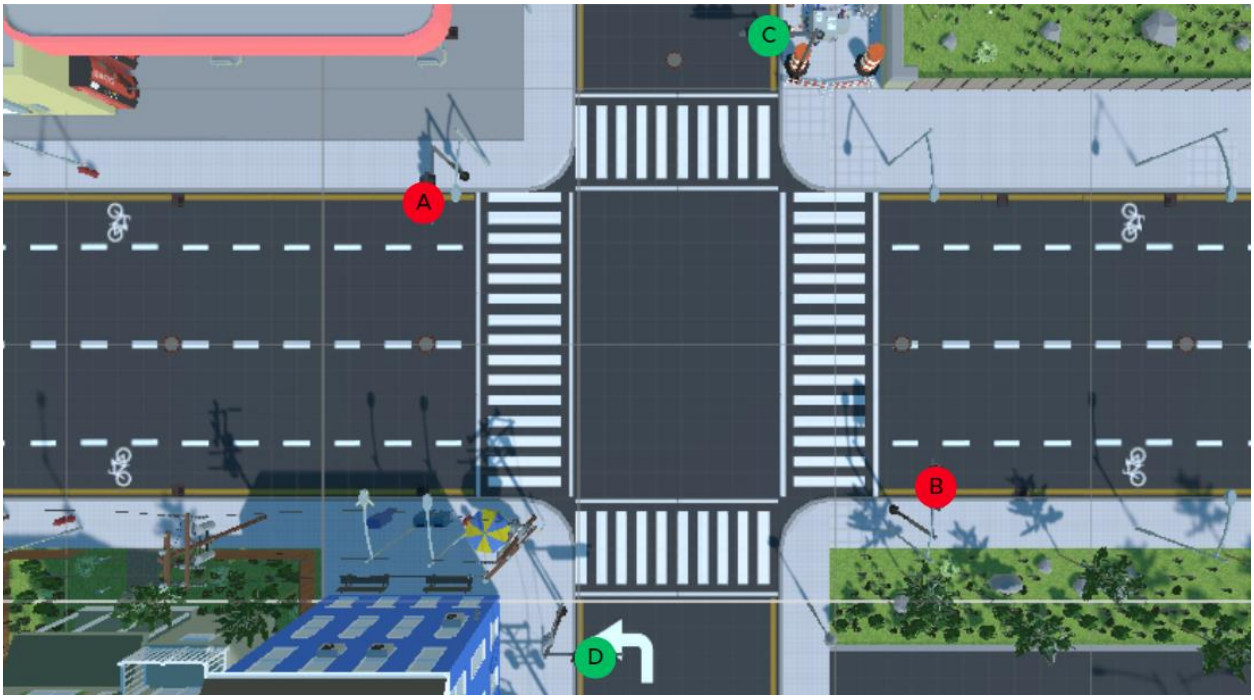



**Fig. 7:** The crossroad with an example of traffic lights state.

From the frontend perspective, there is a square object in a colour of the pole just in front of every light, so they are properly hidden. When a light is to be shown, a square is deactivated. As shown in Fig. 8, squares in front of green and amber lights are activated and hide the respective lights. The square in front of the red light is deactivated but will be set active when the light changes for green.

**Fig. 8:** Red light state.

### 3.2.2. Pedestrians

The following behaviour is implemented in the PedestrianPathFollower.cs file.

There are two pedestrians in the map who move on a designated route (an example has been shown in Fig. 9). The pedestrian follows the route by going from one corner to another. In every corner there is a collision sphere that when entered (*OnTriggerEnter* method) forces the pedestrian to rotate. The angle of that action is determined by the angle between the vector from the previous corner to the collision corner and the vector from the collision corner to the next corner, as shown in Fig. 10.

Another functionality for pedestrians is that it is looking around before entering the street. It is obvious that it is dangerous for a pedestrian to enter the street when a car is very close. Therefore, if there is any car (either user or NPC car) moving with a speed greater than 2 (if smaller, pedestrian assumes that the car is already giving them way) in its collision zone the pedestrian will stop (however, they will still have walking animation) and wait for the car to either move away from the zone or slow down to speed smaller than 2, as shown in Fig. 11.

**Fig. 9:** A route of the pedestrian.

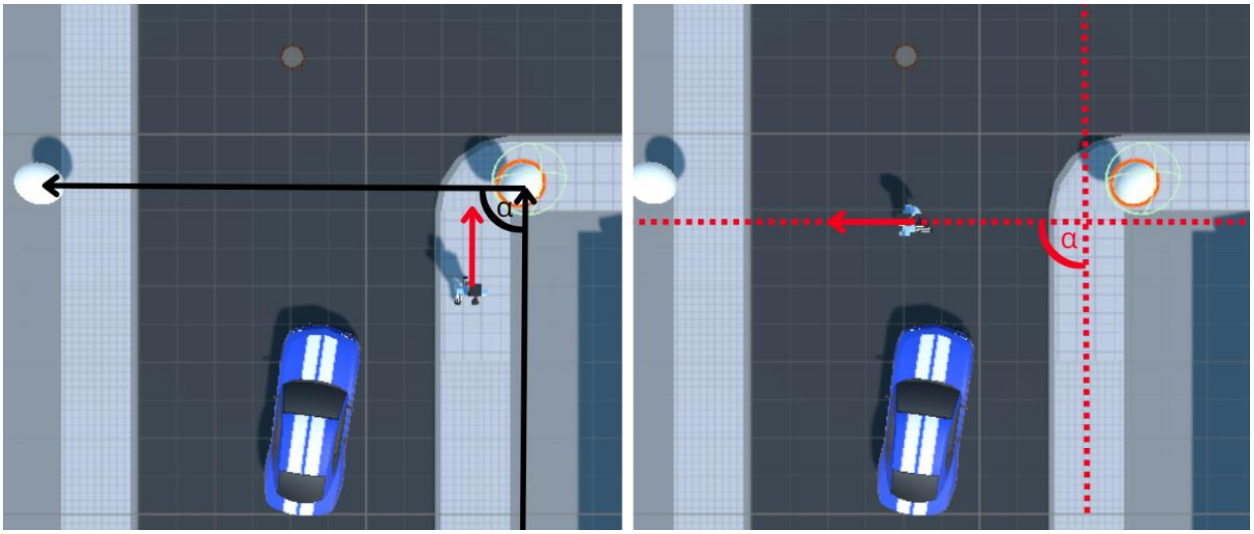



**Fig. 10:** Pedestrian turning left.

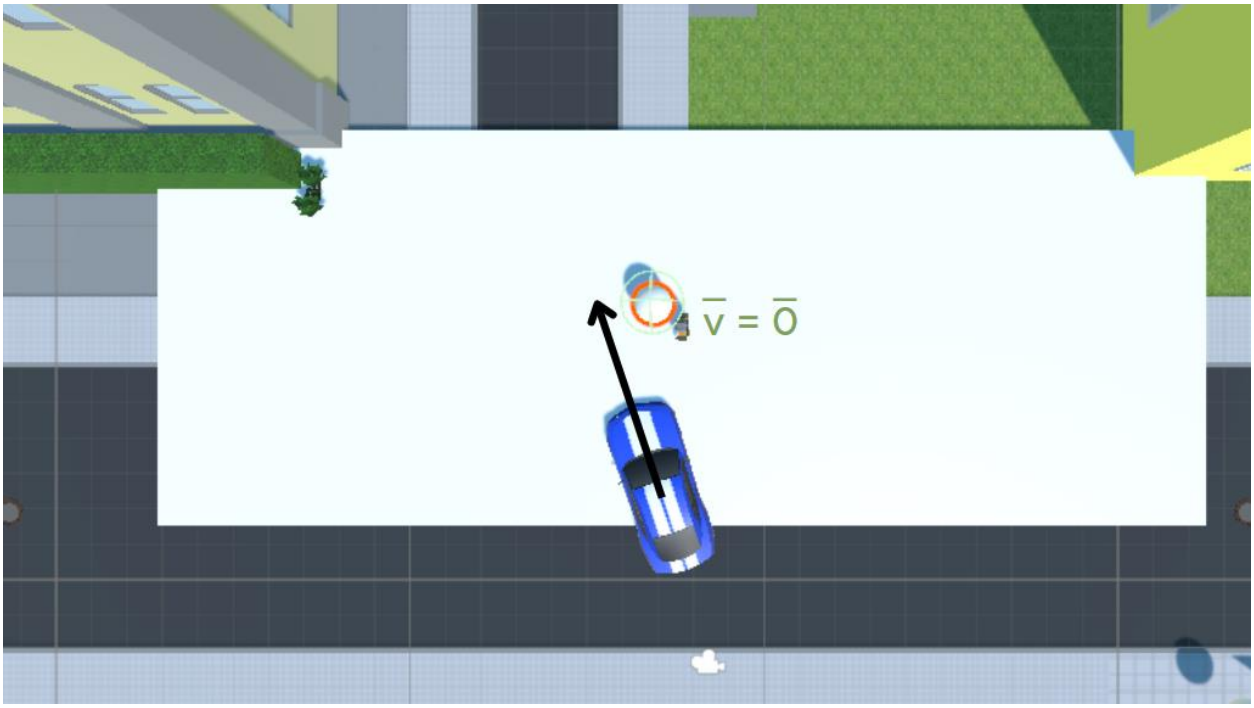



**Fig. 11:** Pedestrian stopping and waiting for the car to pass/give way.

### 3.2.3. NPC car

The following behaviour is implemented in the CarPathFollower.cs file.

Just as pedestrians, the NPC car (only one present in the map) moves on a designated route. However, since car's rigidbody is a long object, it would be making unnatural turns when hit with a collision sphere. Therefore, collision spheres have been moved over the car and the car has been equipped with additional collision box which has a shape of high tower in the car's central point (as shown in Fig. 12).

**Fig. 12:** NPC car before and after turning on a sphere collider point.

When moving around the map, the NPC car meets traffic lights on its way. It must recognise when a light is red and therefore it is obliged to stop and wait for the green light to shine. The car will stop at a collision zone in front of the traffic light, if its tag is *MustStop* (activate when the light is red or red&amber) and go when the tag is *CanGo* (activate when the light is green or amber), as shown in Fig. 13.


**Fig. 13:** The NPC car on traffic lights.

The similar behaviour has been implemented for stopping in front of pedestrians, as shown in Fig. 14. When the NPC car is in the collision zone with the tag *MustStop* (activated when a pedestrian enters the street – they collide with object of tag *Street*) it again stops, and goes when the zone changes its tag to *CanGo* (activated when a pedestrian enters the pavement – they collide with object of tag *Sideway*).

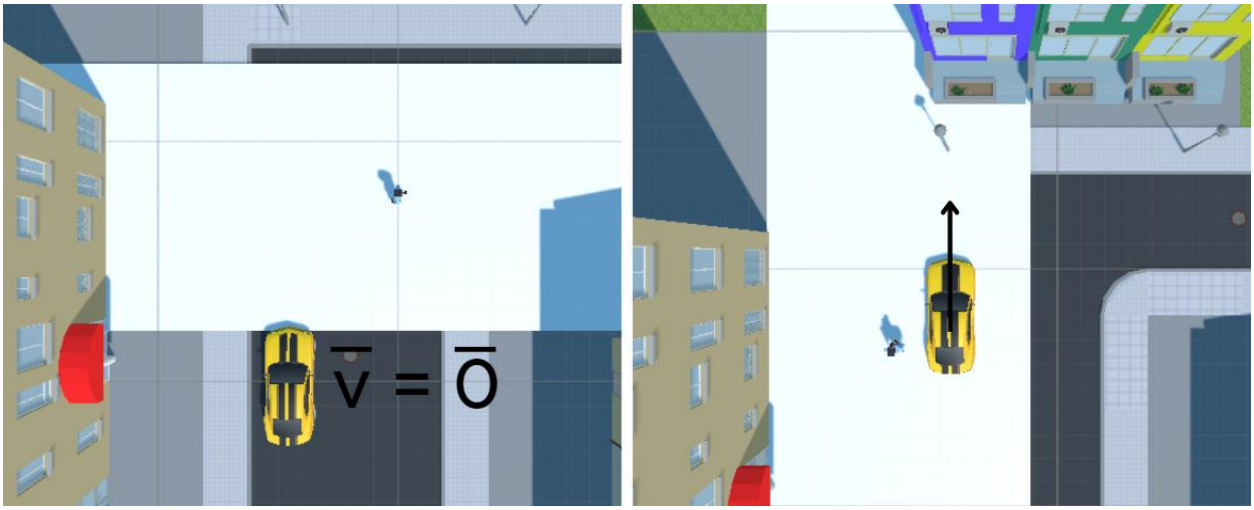



**Fig. 14:** The NPC car giving way to a pedestrian.

The NPC car must recognise when any other car (in the simulator the only other car is the user) is just in front of it so to keep the safe distance. The car has its own Collision Avoidance System. It is implemented using front collision zone, as shown in Fig. 15. When a car is in this zone, the NPC car stops and waits for the other car to move away. The zone does not change its size at no point since the car moves with a constant speed.


**Fig. 15:** The NPC car's behaviour against user car.

For both pedestrians and the car, their moves are controlled with vector translation. Normally the vector is (1, 0, 0), however, if any condition forces the object to stop, the vector is (0, 0, 0).

## 3.3. Current state retrieval

The user moves the car freely around the map. The beliefs and intentions are read based on what other items are around the car, speed and pressed controls.

The vast majority of beliefs, intentions and recommendations are intuitive to understand, however, a table giving explanations in human-readable language is presented in Appendix A.

### 3.3.1. Surrounding items

Car interacts with items around. This is done using one of 3 ways:

(1) **Zone collision:** Item has a corresponding flat zone that a car must be colliding with (encoded with *OnTriggerStay* method) in order to activate the right belief or/and intention. For example, the traffic light has a long zone in front of it, as shown in Fig. 16. If entered, intention *approachingTrafficLight* will be activated. Moreover, with the red light on belief *lightRed* will be passed on, and with amber light on – beliefs *lightAmber* and *ableToStopByWhiteLine*. The similar behaviour is implemented for road signs and pedestrians (pedestrians have two zones, one on each hand side). For all of these items additional requirement is that the car must be approaching from the proper side (as there is no sense to display recommendation of stopping at white line when the stop sign is turned away). As shown in Fig. 17, the angle between zone's rotation and car's velocity vectors must have its absolute value less than 90° in order to activate right beliefs and intentions. This method is also implemented for traffic signs. As the car is not equipped with any image processing methods, as it is with Traffic Sign Recognition systems, the zones in front of traffic signs have been named properly – *SignStop* and *SignGiveWay*. Car recognizes when it approaches a traffic sign, as the zone's name starts with *Sign*, and recognises which sign is it by looking at the full tag name.

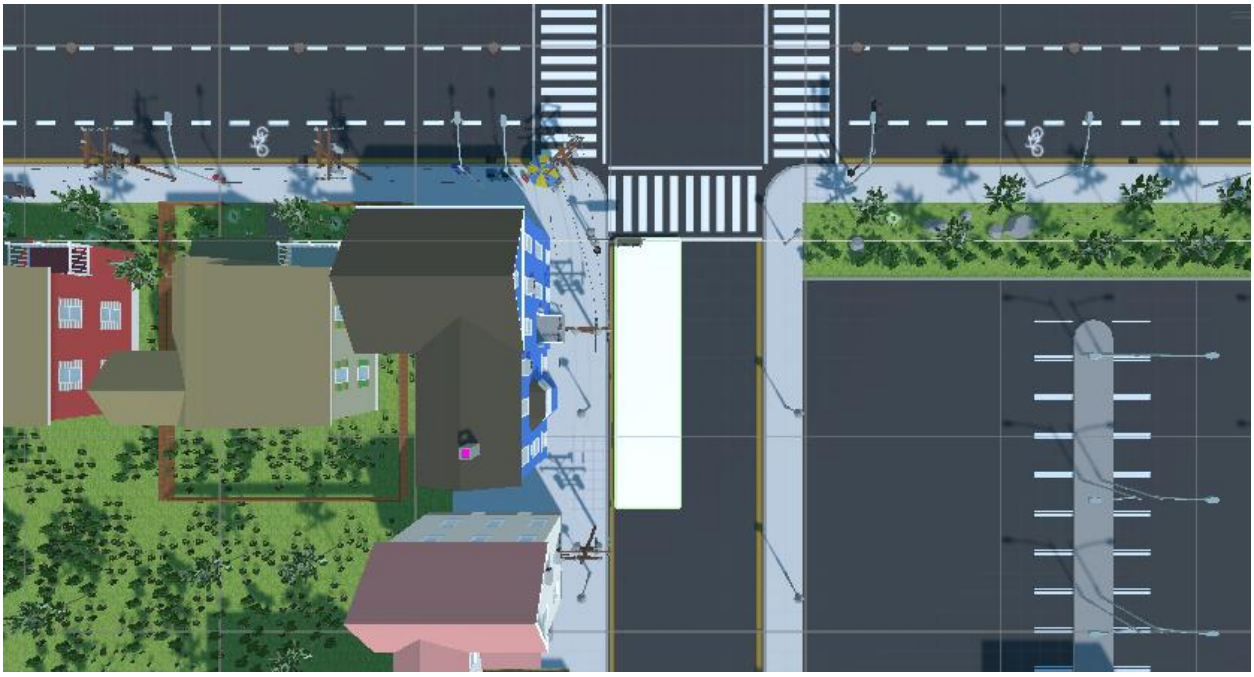


**Fig. 16:** The simulator overview (from bird's eye-view) and the action zone for the traffic light (white rectangle).

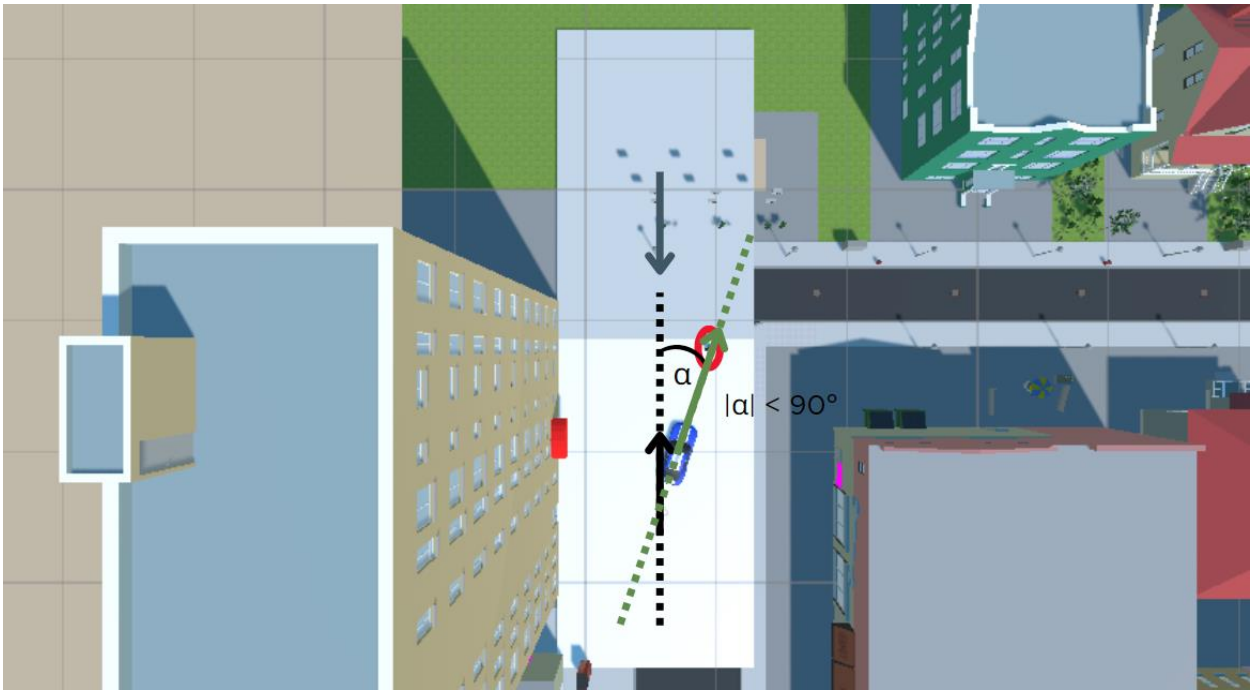



**Fig. 17:** The angle between car's movement and pedestrian's (red circle) left hand side collision zone rotation.

(2) **Item collision:** Item has its collider and a proper tag on, so when the car is colliding (*OnTriggerStay* method) with an item of a specific tag, proper beliefs and intentions are passed on. For instance, colliding with an item of tag *CyclingPath* will activate belief *cycleLaneAvoidable* and intention *beInCycleLane*, therefore recommendations will be displayed, as shown in Fig. 18. This behaviour is implemented for pavements too.


**Fig. 18:** The car entering cycling path.

(3) **Car's zone collision:** The user car cannot get too close to other cars (only one NPC car implemented in the simulator), so it has its own Forward Collision Warning system (in opposition to what the name suggests, it has been implemented for backwards moving too). The user car has its two collision zones (one in front, the other one – in the back) which both change their length depending on the speed. The length of collision zones $l$ is described with the following formula:

$$l = l_0 + l(v),$$

where $l_0$ is the initial length for the zone, it is a distance that should be kept between two non-moving cars on the road. $l(v)$ is the additional length, it is a function of car's velocity $v$. Inferring from the formula presented in chapter 2.1.4.:

$$TTC = \frac{Z}{v},$$

we can compute $Z$ which is a gap between cars, the desired additional length $l(v)$:

$$Z = l(v) = TTC \times v.$$

Only one zone is active at a time, and it is depending on car's direction of movement. It is explained in Fig. 19, where white zone is the active one, and grey one – the inactive one. In the first situation the car is not moving and in the second – the car is moving forward, so in both cases the front collider is active. Whereas, in the last situation the car is moving backwards, so it is the back collider that is set on.

If the active zone is colliding with the environment car (*OnTriggerStay* method, *Car* tag), belief *cantStopBeforeCarInFrontStops* is activated, as shown in Fig. 20.

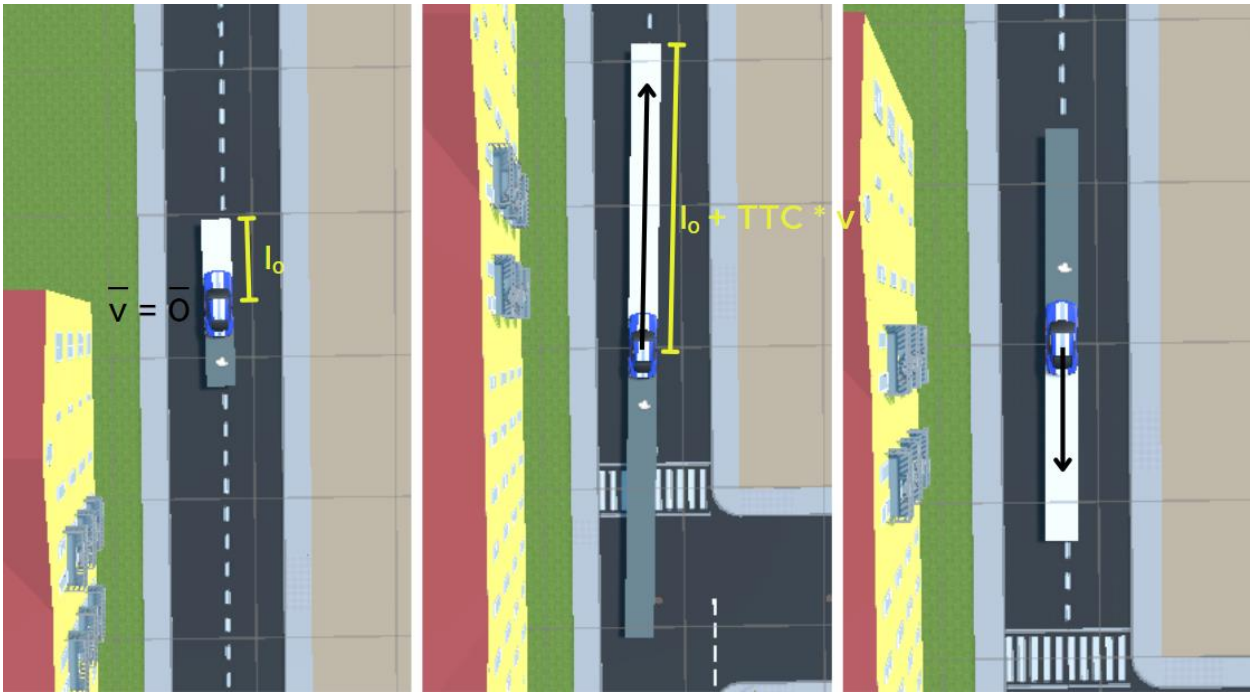



**Fig. 19:** Car and its collision zones, depending on car's speed and direction of movement.

**Fig. 20:** The user car meeting the other car in its active collision zone.

Other requirements for some actions have been implemented so to guarantee a more reasonable output:

- to activate beliefs and intentions for a pedestrian on road, car must have a speed higher than 2 – we can assume that with such a small velocity the user is already expecting the pedestrian to cross the street and giving them way (however, if they suddenly speed up, the recommendation will show up);
- to activate beliefs and intentions for traffic amber light on, car's speed must be less than 6 – it is to ensure that the car will stop on time in front of the light;
- to activate beliefs and intentions for driving on a pavement or a cycling path, or being too close to the car in front, 200 time units must pass since the last time the car has collided with the item/zone activating the belief/intention (this is to prevent the spam of identical recommendations – the assumption is that the user will bear in mind the recommendation for a few seconds).

As mentioned, all these conditions are being checked with *OnTriggerStay* method – as long as the collision is occurring, *addConnectedBeliefsAndIntentions* method will be constantly called. The method will add right beliefs and intentions if they are not yet present in dictionaries storing the current state of beliefs and intentions. Using *OnTriggerStay* method preferably to *OnTriggerEnter* ensures that beliefs and intentions are constantly stored when the car is constantly moving and touching different items of the same tag.

Upon exiting collision, *OnTriggerExit* method takes care of removing intentions and beliefs by calling another method, *removeConnectedBeliefsAndIntentions*. Additionally, this method is also called a few times in *OnTriggerStay* – when traffic lights change to red&amber or green, or pedestrian is entering the pavement (in both cases the zone changes its tag). It is because *OnTriggerExit* requires one of the items to move, while *OnTriggerStay* does not.

### 3.3.2. Car's behaviour

There are two behaviours that do not require any collisions and are being checked using different conditions:

(1) The recommendation for early and light break is checked with following conditions:
    a. the "S" key has been constantly pressed for 0.5 seconds;
    b. ending velocity is less than starting velocity;
    c. starting velocity must be higher than 8, as breaking below that velocity is safe enough.
(2) The recommendation when the maximum speed is exceeded will be displayed when the car's velocity is higher than 16.

### 3.3.3. Summary of current state retrieval

The following tables present a summary of how beliefs (Table 2) and intentions (Table 3) are activated.

| **Belief** | **Activation method** | **Additional requirements** |
|---|---|---|
| *seenSign* | **Zone collision** with a zone of a tag starting with *Sign* | - |
| *signNoConflictsWithAuthorisedPersons* | **Zone collision** with a zone of a tag starting with *Sign* | - |
| *exceedingSpeedLimit* | Overspeeding (as described in 3.3.2.(1)) | - |
| *cantStopBeforeCarInFrontStops* | **Car's zone collision** | 200 times must have passed since the last collision with *Car* item |
| *cycleLaneAvoidable* | **Item collision** with an item of tag *CyclingPath* | 200 times must have passed since the last collision with *CyclingPath* item |
| *pavement* | **Item collision** with an item of tag *Sideway** | 200 times must have passed since the last collision with *Sideway* item |
| *notAccessProperty* | **Item collision** with an item of tag *Sideway** | 200 times must have passed since the last collision with *Sideway* item |
| *pedestriansInRoad* | **Zone collision** with a zone of tag *PedestrianOnStreet* | Car's speed greater than 2 |
| *stopSign* | **Zone collision** with a zone of tag *SignStop* | - |
| *whiteLineAcrossRoad* | **Zone collision** with a zone of tag *SignStop* | - |
| *giveWaySign* | **Zone collision** with a zone of tag *SignGiveWay* | - |
| *dottedWhiteLineAcrossRoad* | **Zone collision** with a zone of tag *SignGiveWay* | - |
| *lightRed* | **Zone collision** with a zone of tag *TrafficRed* | - |
| *lightAmber* | **Zone collision** with a zone of tag *TrafficAmber* | Car's speed less than 6 |
| *ableToStopByWhiteLine* | **Zone collision** with a zone of tag *TrafficAmber* | Car's speed less than 6 |

**Table 2:** All beliefs and methods they are activated with.

| Intention | Activation method | Additional requirements |
|---|---|---|
| *brake* | Braking (as described in 3.3.2.(2)) | - |
| *beInCycleLane* | **Item collision** with an item of tag *CyclingPath* | 200 times must have passed since the last collision with *CyclingPath* item |
| *approachingTrafficLight* | **Zone collision** with a zone of a tag starting with *Traffic* | - |

**Table 3:** All intentions and methods they are activated with.

* In a more complicated world map, my suggestion is to rename Sideway tag into *NotAccessPropertySideway* and change the condition of belief *notAccessProperty* so it is activated when colliding with an item of a tag starting with *NotAccessProperty*. This is due to the fact that in the simulator only one type of property with forbidden access has been implemented and no attention has been paid to allow future workers to extend that feature. Fig. 21 represents suggested methodology – at first it would be checked whether the tag is starting with *NotAccessProperty* so belief *notAccessProperty* could be activated. Then the full name of the tag would be compared so one of tags *footpath*, *pavement* or *bridleway* could be activated too, and therefore recommendation *must-road_surfaces-avoid_non* could be returned. This reasoning is equivalent to methods applied to items which tags start with *Sign* or *Traffic*.

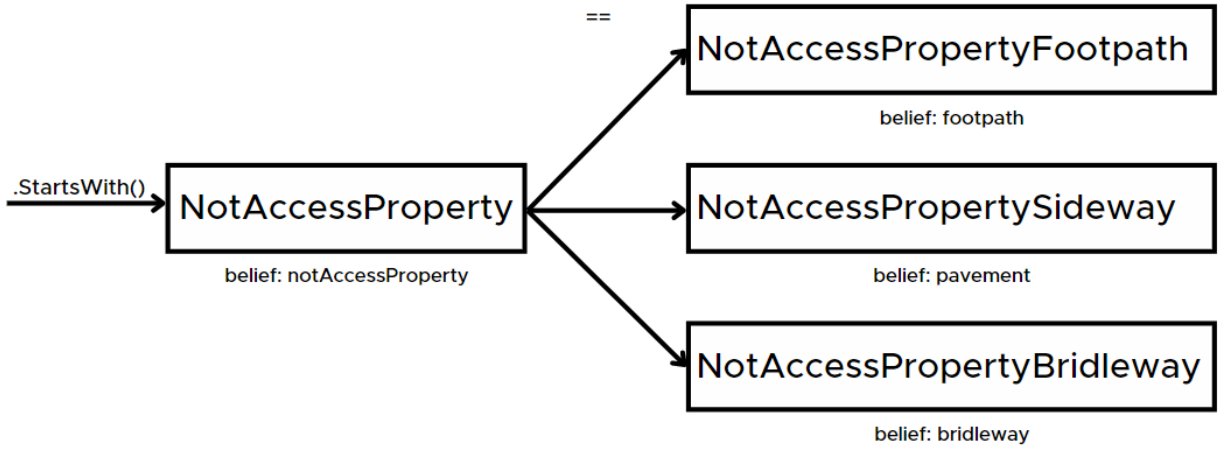



**Fig. 21:** Diagram representing reading beliefs for properties with forbidden access.

# 3.4. Backup camera

To give a better overview of the situation behind the trunk, when heading to that direction, a backup camera has been implemented. As mentioned in 3.3.1., car's collision zones are either active or inactive, depending on its direction of movement. This is checked by *getDirectionOfMovement* method which returns 0 if car is not moving, returns 1 if car is moving forwards or returns -1 if car is moving backwards. This is done by comparing velocities x, z and car's rotation. As in an example shown in Fig. 22, car's velocities x and z have their values both positive or equal to 0 which means that the car is moving towards 1st quarter (as shown in the right side of Fig. 22, the map has been divided into plane coordinate system with car being in its centre). At the same time car's rotation is greater or equal to 0° and less or equal to 90° meaning that the car is moving forwards so 1 is returned. It could have been moving towards 1st quarter with the same vector but with having its rotation between 180° or 270°, meaning backwards moving (and returning -1).

```
// 1st quarter
else if (deltaX >= 0 && deltaZ >= 0)
{
    if (rot >= 0 && rot <= 90)
    {
        direction = 1;
    }
    else
    {
        direction = -1;
    }
}
```

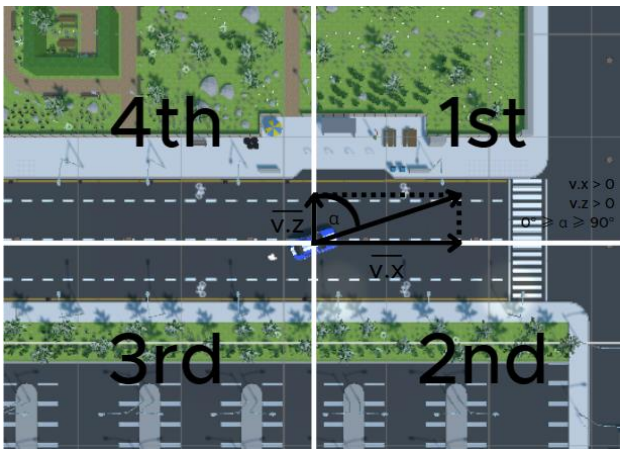



**Fig. 22:** The example of *getDirectionOfMovement* code (on the left) and map's division into quarters with car in the centre (on the right).

The camera starts with being slightly over the car and behind its trunk, showing what is in front of the vehicle. When car's movement direction changes to -1, camera moves to in front of car's hood and changes its y position by 180° so it shows what is behind the car. When movement direction changes to 1, camera comes back to its original position. This is illustrated in Fig. 23, where the car is moving forwards on the left image and backwards in the right image. The camera views are presented in bottom right corners.


**Fig. 23:** Camera's positions.

When testing the algorithm, it was noticed that sometimes inaccurate values were returned resulting in camera rapidly flashing. To prevent that, the *getDirectionOfMovement* method was implemented (instead of returning the result of a single check) and it gives the most common result of last 25 checks (stored in the queue). Moreover, camera's change is possible only at speeds between 0.2 and 6 – that is to prevent flashes when car is not moving or going with high speeds.

# 3.5. Recommendations display

The feature of showing recommendations has been implemented using UnityEngine.UI. An example of recommendations displayed to the user has been presented in Fig. 24. The full list of possible recommendations and their causes is presented in Table 4.

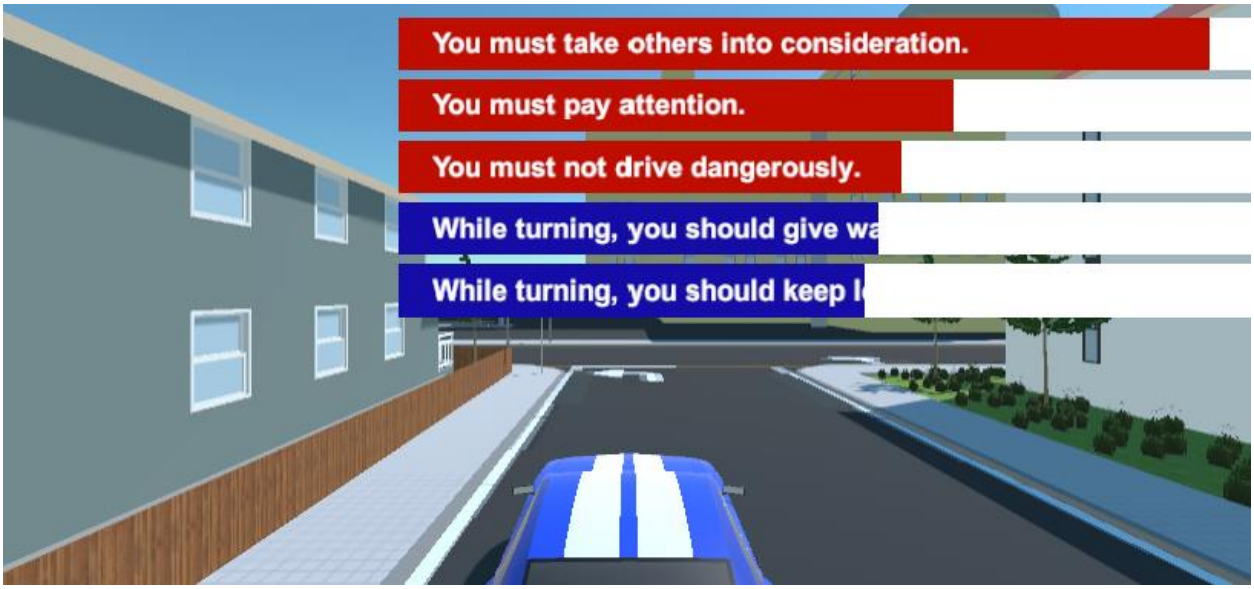



**Fig. 24:** Set of initial recommendations being displayed upon starting the simulator.

| **Beliefs** | **Intentions** | **Recommendations** | **Rule number** |
|---|---|---|---|
| *seenSign, signNoConflictsWithAuthorisedPersons* | - | *must-follow_sign* | 109 |
| - | *brake* | *should-brake_early_lightly* | 117 |
| *exceedingSpeedLimit* | - | *must-reduce_speed* | 124 |
| *cantStopBeforeCarInFrontStops* | - | *should-increase_distance_to_car_in front* | 126 |
| *cycleLaneAvoidable* | *beInCycleLane* | *must-use_road, must-avoid_parking* | 140 |
| - | - | *must-not_drive_dangerously, must-drive_care_attention, must-consideration_others* | 144 |
| *pavement, notAccessProperty* | - | *must-road_surfaces-avoid_non* | 145a |
| *pedestriansInRoad* | - | *should-give_way_to_pedestrians* | 170 |
| *stopSign, whiteLineAcrossRoad* | | *must-stop_at_white_line, should-wait_for_gap_before_moving_off* | 171 |
| *giveWaySign, dottedWhiteLineAcrossRoad* | - | *must-give_way_at_dotted_white_line* | 172 |
| *lightRed* | *approachingTrafficLight* | *must-stop_at_white_line* | 175a |
| *lightAmber, ableToStopByWhiteLine* | *approachingTrafficLight* | *must-stop_at_white_line* | 175b |

| | | | |
|---|---|---|---|
| - | - | *should-keep_left, should-give_way_other_roads* | 183 |

**Table 4:** All implemented recommendations with beliefs and intentions causing them.

To ensure that the simulator is as readable as possible, a few measurements have been implemented:

(1) A maximum of 5 recommendations can be displayed at a time. If any recommendation is waiting to be shown, it will show up as soon as the top recommendation has disappeared.
(2) The top recommendation will disappear after 50 time units have passed since it appeared. The others will be then moved one position up.
(3) The recommendations starting with *must* are being displayed in red, while those starting with *should* – in blue.
(4) The recommendations have been translated into human-readable language (they are presented in Appenidx A). Translations have been provided manually instead of having a translating algorithm as some original recommendations look confusing (therefore, *must-drive_care_attention* has been translated into *You must pay attention.*). However, with a bigger recommendations set a translation algorithm would have been implemented.

# Chapter 4

# User Evaluation

The following chapter gives an overview of the user evaluation conducted in order to examine the user's experience with the simulator. It is a fulfilment of objective (5) *To perform user evaluation which will check if the simulator is user-friendly and if recommendations displayed are relevant* described in chapter 1.3.

## 4.1. The questionnaire

The evaluation has been carried out using Google Forms tool. The questionnaire contained clear instructions on how to download the simulator and how to install SWI-Prolog, if not already operating in the machine. The questions were the following:

1. What is your age?
2. Are you a computer scientist/Do you study CS or anything related to that?
3. Do you have experience with other simulators, especially the driving ones?
4. How much time did you spend on the simulator?
5. What were your first impressions about the simulator? Did you find it easy to adapt?
6. The recommendations you see on the screen: are they readable? Do they show up too fast/too slow? Are they understandable?
7. Have you seen any recommendations that were not relevant for a particular road situation? Can you describe all such situations and rules you have seen?
8. Have you seen/experienced any situation that you expected a particular recommendation to appear, but eventually it did not? Can you describe all such situations and rules you expected?
9. Have you noticed any bugs (not concerning the recommendations) that definitely should not appear there? Did you experience anything that prevented you from using the simulator properly?
10. After running the simulator a couple of times, what are your general feelings?
11. If you have experience with other simulators, how would you compare mine to them?
12. Anything else about the simulator you would like to speak about?

As it is a common knowledge that younger people deal with technologies better than their parents' generation, the reason behind asking for the person's age in question 1 was to compare impressions on the simulator of people in different age groups.

The purpose of questions 2-3 was to distinguish how people with some experience with coding and some simulators have reacted to my simulator in comparison to people with no or little experience.

The reason behind question 4 was to check on how many bugs and mistakes would have been noticed depending on the time spent exploring the simulator. Do bugs show up immediately or do you have to go deeper in the simulator to find them?

Questions 1-4 asked for responders' demographics so to know who are the people answering the questionnaire and if there are any particular differences between some groups in terms of their feelings on

the simulator. Questions 5-12, all of them being open-ended, asked for impressions and abnormalities noticed. Questions 11 and 12 were optional.

This method of evaluation has been checked against The University of Manchester Ethics Decision Tool. As the evidence shows in Fig. 25, no ethical approval was required for this evaluation method.

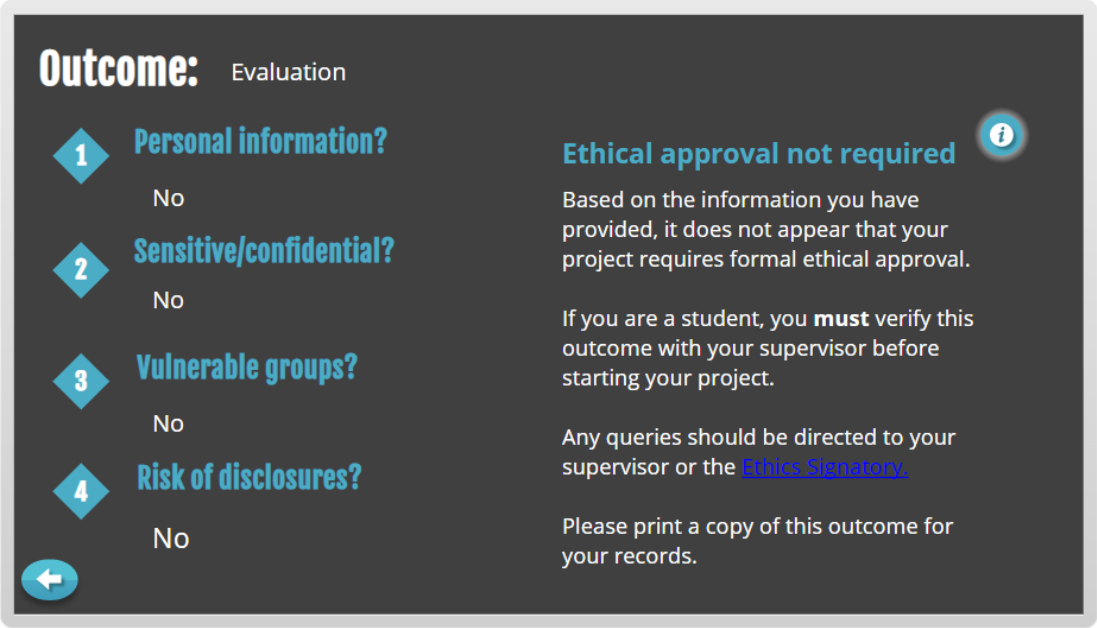



**Fig. 25:** The outcome of The University of Manchester Ethics Decision Tool.

# 4.2. Responses

In total, there have been 11 responses in the evaluation.

**Q1-4:** Since the majority (90.1%) of responders come from the age group 18-25, it is impossible to perform a comparison of impressions on the simulator depending on user's age. Only 27.3% of responders are Computer Science students or professionals. In terms of experience, half of responders have already had experience with driving simulator, while the other ones had only a little or none at all. Quite even split of answers is also on time declared to have spent with the simulator. Half of responders have been exploring the simulator for less than 15 minutes, while others devoted some more time to that (but less than 45 minutes). Answers to these four questions are presented in Fig. 26.

**Q5:** First impressions have been positive in general: people found it easy or moderate to adapt to car's control and physics. One answer was critical about the car's physics, however, it has been imported from Unity Assets Store instead of being implemented by myself. Two answers also noticed problems with flashing camera. This has been noticed as well when developing the simulator – that is why some measurements have been implemented to camera rotation feature (described in 3.4.). However, implementing another method could be considered in the future.

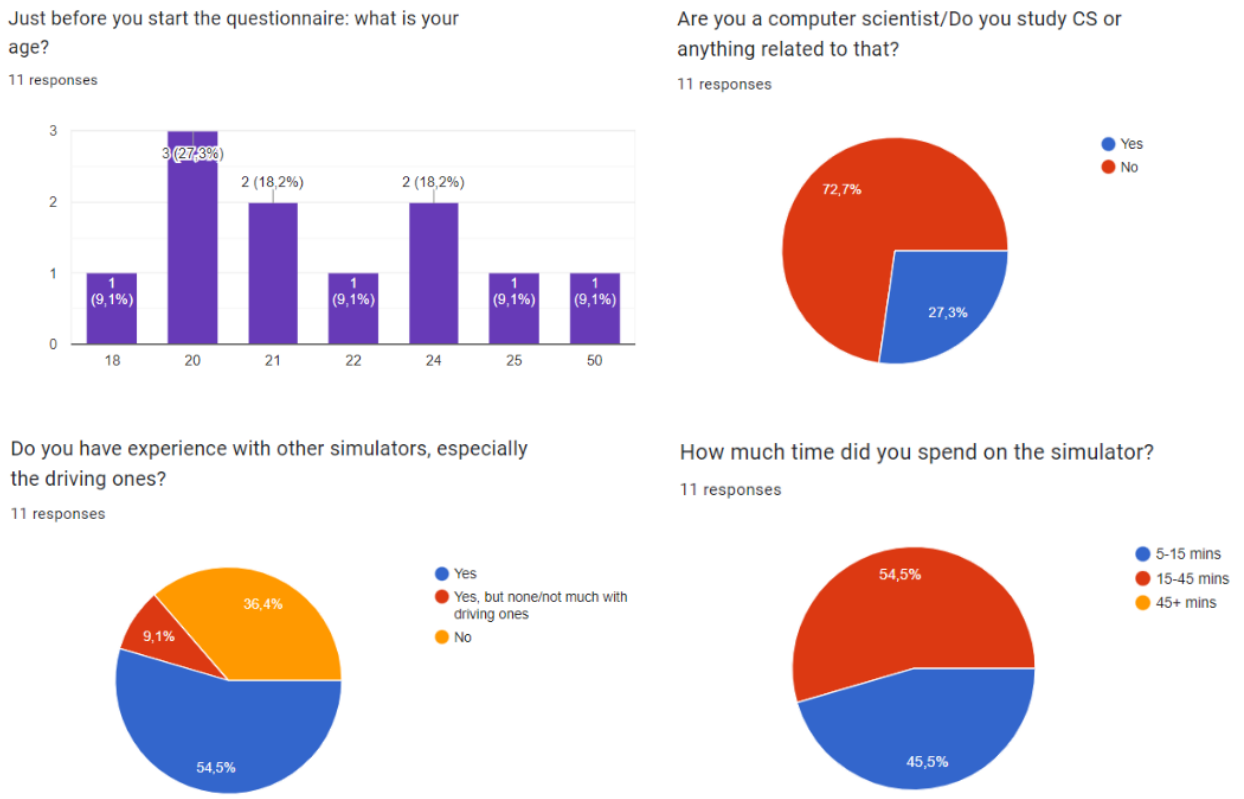



**Fig. 26:** Answers to questions 1-4.

**Q6:** There were mixed feelings about recommendations appearing on screen. While everybody agreed that the language was understandable, most of responders complained about the speed of recommendations showing up. Advice was disappearing from the screen too quickly, leaving people with not enough time to move their eyes and read. This definitely leaves room for improvement – the timer for recommendations has been implemented using a counter which increments with every Update function, rather than using real-time timer. The change of method would allow for a fixed period in which a recommendation would be displayed, however, there is still no perfect answer to that problem as for some people it still would be too fast, while for others – too slow. There was also a single complaint that the recommendation's box was too small and had a disturbing choice of colours.

**Q7:** No one reported a situation in which recommendations irrelevant to a road situation appeared. However, there were some answers that matched the next question more than the current one, so they will be talked about in the next paragraph.

**Q8:** This question asked users to describe situations in which some advice should appear but did not. As responders were not informed about the set of recommendations that can be displayed, some answers talked about unimplemented recommendations.

(1) It was pointed out that no warning gets displayed when the car hits a pedestrian or the other car. Although there exists a highway code on what to do in case of an accident, it has not been implemented to RoTRA. One way of solving the issue would be to implement these rules to RoTRA. Another way is that we can consider these rules as consequences of not following recommendations of giving way to a pedestrian or keeping the distance from the other car. Therefore, proper warning could get displayed when missing a recommendation.

(2) A few people mentioned that there is no information when a car moves the wrong side of the road. These have not been implemented in the simulator though. RoTRA defines a rule to always keep on the left rather than to return when moving the wrong side. Since it would be too annoying too constantly receive a recommendation to stay on the left side, I did not implement that rule. However, as in (1), further work might be done to implement this feature when breaking the rule.
(3) As shown in Fig. 27, no recommendation shows up on which way to follow when at a junction. This simply has not been implemented.
(4) Some recommendations suggested by responders are not encoded in RoTRA, such as not to hit a pedestrian on pavement (however, it could be implicit in *must-not_drive_dangerously* rule), when a map ends (which is not possible in real world) or when a car moves backwards.

What is important, nobody has reported that one of the already implemented recommendations does not get displayed when it should.

**Fig. 27:** No recommendation to turn left showing up.

**Q9:** A couple of minor bugs have been reported.

(1) Pedestrians *walk through* the car that is on their way, as if they pushed it away with their physical force.
(2) Some issues with the map, such as pink textures on buildings or mirrored text on signboards (mentioned in chapter 3.1.2.).
(3) The void being treated as road – this has been implemented on purpose, so users do not fall off the map.
(4) Issues with the camera flashing, already mentioned in Q5.

Bugs (1) and (2) have been spotted, however, since they are only details that are not relevant for the final outcome of the project, not much time was devoted to cosmetic fixes. However, some work could be possibly done in the future to make the simulator look more realistic.

**Q10:** The general feedback is positive. After adapting to the controls, it was easy for users to steer the car and drive along the map, only one person still struggled a bit with the car's physics.

**Q11:** The simulator is thought to be quite good compared to others.

**Q12:** Some suggestions for further work have been mentioned not only in this question, but in previous ones as well. The simulator could do better with features such as car's speed being displayed, minimap, welcome menu, quit button and driving instructions.

It was noticed that people who spent more time in the simulator have spotted more bugs or reported more recommendations not showing up when they should, than those who spent less than 15 minutes. However, no other correlations between some demographic groups and their general feedback have been observed. 11 is a number of responders too small to perform more analysis.

The general feedback was positive and responders found the simulator easy to adapt, good in use and readable. Improvements were pointed out, as well as bugs spotted and additional features suggested.

# Chapter 5

# Development

This chapter gives an overview of the project's development phase, showing the initial plan and indicating changes made.

The first plan of the project's development phase was entirely different from the eventual approach taken. The initial schedule is presented in a form of Gantt Chart, in Table 5 for Semester 1 and Table 6 for Semester 2.

The first draft of the project included preparing 5 different scenarios. The user would have been able to freely choose between maps and test different driving situations. After starting the scenario, the user would have to get through a situation. Once finished, the simulation would be over. My decision to drop this approach and choose one bigger map combining different road situations happening simultaneously enables to test the road advisor more precisely. Apart from that, this approach gives a better insight into real-world conditions – no driver would have to face just one driving situation on a journey.

| Semester 1 | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Week 3** | **Week 4** | **Week 5** | **Week 6** | **Week 7** | **Week 8** | **Week 9** | **Week 10** | **Week 11** | **Week 12** | **Christmas Break & Exams** |
| Conduct research | | | | | | | | | | |
| Prepare descriptions of 5 scenarios | | | | | | | | | | |
| Develop scenarios and implement static elements into the simulator | | | | | | | | | | |
| | | Implement dynamic elements into the simulator | | | | | | | | |
| | | | Implement car controls | | | | | | | |
| | | | | | Implement beliefs reading | | | | | |
| | | | | | | | Implement intentions reading | | | |
| | | | | | | | | | Implement communication between RoTRA and the simulator | |

**Table 5:** Initial Gantt chart of activities scheduled (Semester 1).

The first plan saw two weeks devoted to implementing dynamic elements into Unity. However, this step needed much more time in total as the logic behind traffic lights, pedestrians and the NPC cars had to be encoded. On the other hand, implementing car's controls (with three weeks to be spent on it originally) turned out to be a quicker step – the car's physics was downloaded from Unity Assets Store and it only required me to experiment with parameters to choose the most suitable ones for city centre environment.

The initial plan included two weeks devoted for implementing beliefs and then two weeks for intentions. As it quickly became clear, a much reasonable approach was to work on specific rules, one by one. It enabled me to catch errors faster and make sure that every rule was correctly returned. Therefore, communication between RoTRA and the simulator was implemented before starting to work on recommendations.

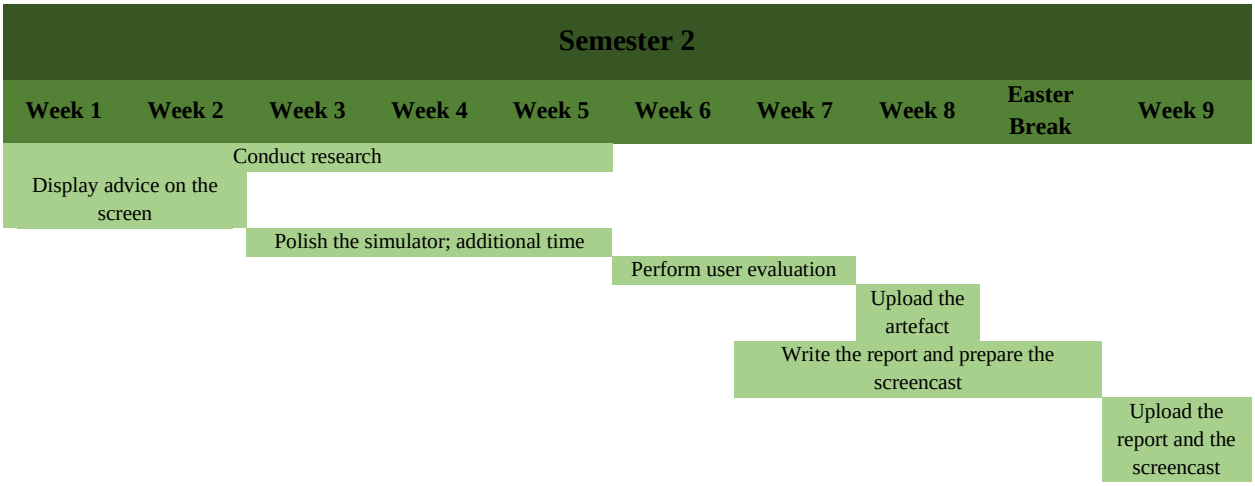



**Table 6:** Initial Gantt chart of activities scheduled (Semester 2).

As I took rule-by-rule approach, the order of actions performed was entirely altered. I started with having the map and car ready and then I worked on every rule one by one. Therefore, the activities of implementing the communication of beliefs and intentions, and encoding logics behind dynamic elements were performed simultaneously.

A huge drawback of my initial plan was the lack of time devoted to exploring different simulators, choosing the best one and learning its features and limitations. It took around three weeks of installing already existing simulators and checking their possibilities. Eventually, I decided to use Unity as my development tool – it is intuitive in use, and it has a wide base of users, which allowed me to download assets from the store, rather than spending too much time on creating graphics, which is not within aims of the project.

As it was set in the plan, research was being conducted along the whole development process to support me with a good theoretical background.

A Gantt chart reflecting the final sequence of activities taken has been presented in Table 7.

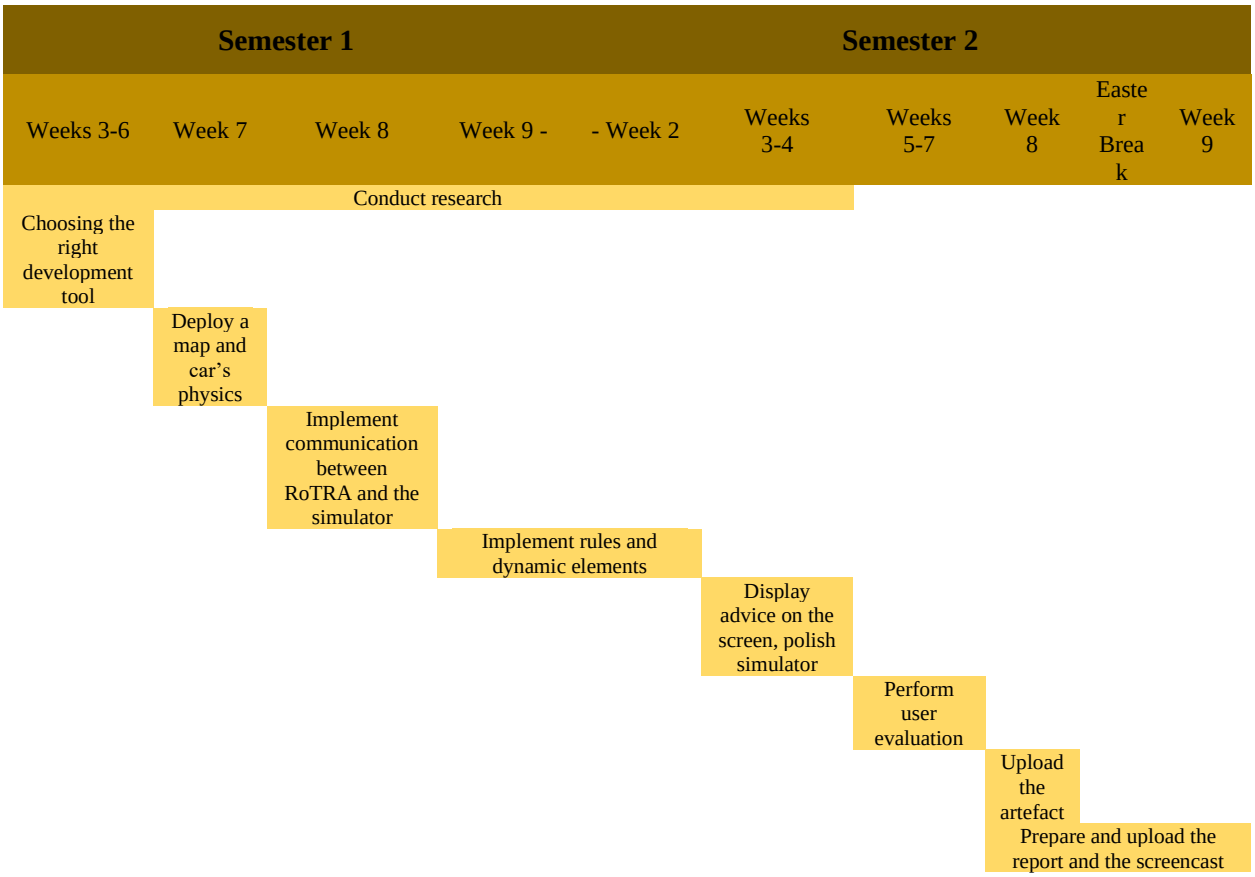



**Table 7:** Gantt chart of final activities taken.

# Chapter 6

# Discussion

This chapter presents the final conclusion of the project, including a summary of achievements, suggestions for further improvements and challenges.

## 6.1. Summary of achievements

The project activities were started around October 2022 and finished in March 2023. Following is the description of achievements, with respect to aims and objectives described in chapter 1.2.:

**(1) To explore further the subject of autonomous vehicles, paying particular interest to the safety of such devices and existing assistance features.** A thorough analysis of the topic supported by a deep literature survey has been presented in chapter 2. Knowledge acquired and reflections made were considered at every step of the project's implementation.

**(2) To develop a 3D Unity-based simulator which enables a user to freely control the car.** A high-quality 3D simulator was developed in Unity. It uses simple controls that allow to freely move the car around the map.

**(3) To enrich the car with methods that enables it to properly analyse the environment.** A wide range of methods with different requirements has been developed for the car. The vehicle, when moving around the map, correctly scans the environment and converts that into right sets of beliefs and intentions.

**(4) To convert the environment analysis into a set of law recommendations by establishing communication between Unity and RoTRA.** Connection between Unity simulator and RoTRA has been successfully developed. Correct sets of recommendations are being returned.

**(5) To perform a user evaluation which will check if the simulator is user-friendly and if the recommendations displayed are relevant.** The evaluation has been performed on a group of 11 people. The general feedback was eminently positive, reflecting some minor issues and suggestions of possible enhancements.

Therefore, the aims and objectives of the project were greatly fulfilled.


**Fig. 28:** Car driving on a cycling path (left image), and on the same line on the crossroads (right image).

## 6.2. Challenges

The aspect of the legal advisor I found the most challenging was on defining and implementing the intentions. Since the computer does not know what direction the user wants to go on the crossroads until they press one of keyboard keys, it is hard to give a proper advice in advance. But talking about the recommendations that have been actually implemented: we do not know if the user wants to enter the cycling path, turn left or just simply move the car slightly to the left, until they really perform that. In my simulator intention *beInCycleLane* is activated as soon as the car enters the cycling path (but not when it does it on the crossroads), as shown in Fig. 28.

Regarding intention *approachingTrafficLight*: while it is not hard to recognise when the car gets close to traffic lights, a hypothetical situation (not implemented in the simulator but possible in a more extended map) could include two crossroads, A and B. Crossroad A is not equipped with traffic lights but crossroad B, which is just after when A is passed, has traffic lights. A car needs to be informed to stop at red light far before reaching A, however, the computer does not know if the car is going to go through B or turn left on A. Therefore, if the car turns left on A and gets recommendation to stop at red light, the advice is unreliable. The situation has been illustrated in Fig. 29.

Intention *brake* has a similar problematic nature: the computer will know if a user wants to brake only when they hit key *S*. Therefore, the recommendation to brake lightly and early can be displayed after the activity has been initialized.

This problem could be solved, for instance, with question boxes appearing on the screen asking for user's intention, however, that would be found really annoying to answer the same question all the time. Another approach to that problem, a more realistic one, is to extend the controls to allow signalling the intentions: one key for left turn signal, another for right turn signal and another one to signalise the intention to brake. This solution makes steering far more complicated and unfriendly: while drivers are used to have tens of differently activated controls in their real-world vehicles, the game cars are usually steered with just 4 keys.

Intentions' implementation seem to be a less of a problem when considering vehicles with more advanced systems: cars with navigation on and route set will be able to recognise the reason behind entering a cycling path and will know if a car should turn left on A or go through B in the crossroads hypothetical situation.

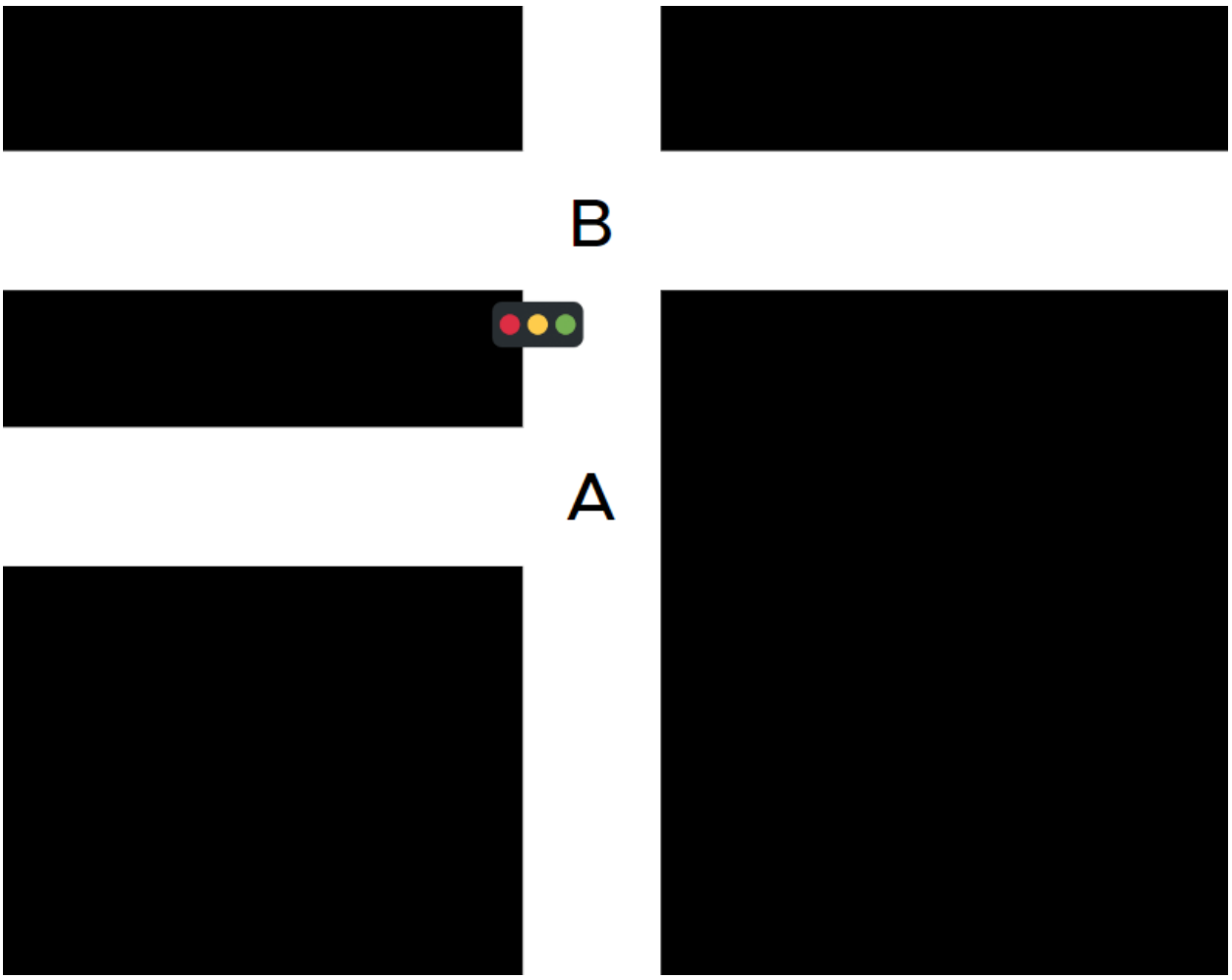



**Fig. 29:** Hypothetical problematic situation for intention *approachingTrafficLight*.

# 6.3. Improvements

As described in chapter 4.2., there are several improvements to be made in order to convert the simulator and the legal advisor into a more realistic tool. These enhancements include:

(1) The way recommendations appear was too fast for some people. The Update function counter should be replaced with real-time timer and some more evaluation should be performed to find the closest-to-perfect time period of a recommendation's display.
As some people pointed out, it takes time to move eyes to read recommendations. Therefore, when altering the way of advice display, a great inspiration could be taken from Automotive Head-Up (shown in Fig. 30) which allows to show information on the line between driver's eyes and their sight [27]. In the simulator, recommendations could be shown either as thin glowing letters or half-transparent boxes in the middle of the screen.


**Fig. 30:** Automotive Head-Up Display.

(2) Another suggestion inferred from the user evaluation concerns implementing more recommendations into the base. While having a more detailed legal advice would work well for a smart vehicle, it could possibly harm a human driver's attention. Since the UK Highway Code is a big set of rules, loads of recommendations would show up in a short time. A human driver would not manage to read or listen to all of them, therefore, they would quickly lose focus. When enlarging the advice set, methods of choosing the most important recommendations to display at a time should be thought through and implemented.
(3) It has been pointed out a few times that the camera sometimes flashes or even shows the wrong direction of driving. While it does not happen that often, some improvements or even a complete change of camera's methods should be considered.
(4) A system that detects when a recommendation was not followed (so to display a proper eye-catching warning) should be examined.
(5) Additional propositions include cosmetic changes – improving pedestrians' physics, additional elements of a simulator, cosmetic changes of a map.

A bug, which was not spotted by evaluation responders, concerns pedestrians and the NPC car route following. Very rarely it is the issue that they miss a corner collision sphere and keep going straight, and later they either stop on an obstacle or end up outside the map. An improvement of the route following methods could be considered.

Aside from the user experience side, an improvement to the code could be introduced as well. As mentioned in chapter 3.3.3, the reasoning behind activating the belief *notAccessProperty* is equivalent to those for beliefs *seenSign* and *signNoConflictsWithAuthorisedPersons* and intention *approachingTrafficLight*, however, two different methods were used. Applying general methods to equivalent or similar problems would make the code more readable and easier to adapt with for a programmer.

## 6.4. Further work

The further work that could be done with an accurate legal advisor includes connecting it to other ADAS technologies in order to increase the car's autonomy. Apart from the Automotive Head-Up Display mentioned earlier, Adaptive Cruise Control systems could take a great advantage of RoTRA. These are systems that are used to adjust the car's velocity and distance to vehicles in front [28]. This technology can be implemented for a car using a RoTRA-based advisor – the car can be moving with a speed just below overspeeding, and always keeping safe distance from another road user.

Another work that could be done is to make my legal advisor comply with jurisdictions of other countries. There are a few possible ways of achieving the goal:

- By developing a RoTRA version applicable to a country of interest. The challenge here is to make the advisor agent deal with beliefs, intentions and recommendations which are not present in the UK Highway Code (that would especially be the case for countries using right-hand traffic systems). It could be done by extending the script responsible for connection with RoTRA.
- If there is a tool that encodes the highway code of a country of interest and it receives beliefs and intentions as input and returns recommendations as output, it could be used. In order to make successful connection between Unity and the tool, the script must be adjusted to be able to call and receive output from agents written in different languages.

If a tool works in a different way than described above, adjustments would be major as they would include a complete change of logics behind the advisor.

## 6.5. Final comments

As it was presented in chapter 5, the initial plan has really changed throughout the development, since it came out to have some ineffective strategies (such as working on the communication with RoTRA separately from implementing beliefs and intentions). Next time on projects on such a large scale I will put more effort into planning of methodologies. Apart from that, the first plan did not assume time devoted to research into practical tools. This major drawback has shown me how important it is to spend much time into looking for right tools and learning on how to use them.

Undertaking this project has been a challenging and time-consuming endeavour that required a significant amount of dedication and passion in both the research and development phases. Through this project, I have gained valuable experience and skills, such as learning how to work on large-scale tasks and how to identify effective frameworks to support my work. As described in the previous paragraph, the project has highlighted some critical shortcomings in my planning process, which has served as an important lesson for my future work. In particular, it has taught me the importance of careful planning and the need to consider all possible factors that may impact the success of a project.

In addition, the project has provided me with a deeper theoretical understanding of the challenges facing the AVs industry, such as the need to ensure safety, reliability, and regulatory compliance. This knowledge will be invaluable in my future endeavours, as it has equipped me with the ability to identify and analyse key trends and issues in this field. Furthermore, I have gained practical experience in the development of driving assistance features.

Overall, this project has been a significant learning experience that has provided me with a broad range of skills and knowledge that will be invaluable in my future work. The project has met its aims and objectives: a good quality simulator for a car legal advisor has been produced.

# References


[1] Jones, B.: (Feb. 2019). *Learning to be Decisive Drivers.* [online] Available at: https://www.lymanrichey.com/news/2019/2/4/decisive-drivers [Accessed 16 Apr. 2023].

[2] Collenette, J., Dennis, L.A. and Fisher, M. (2022). Advising Autonomous Cars about the Rules of the Road. Electronic Proceedings in Theoretical Computer Science, 371, pp. 62–76. doi: 10.4204/eptcs.371.5.

[3] M. Elbanhawi, M. Simic and R. Jazar, *In the Passenger Seat: Investigating Ride Comfort Measures in Autonomous Cars,* IEEE Intelligent Transportation Systems Magazine, vol. 7, no. 3, pp. 4-17, Fall 2015, doi: 10.1109/MITS.2015.2405571

[4] Cunningham, M.L., Regan, M.A., Horberry, T., Weeratunga, K. and Dixit, V. (2019). Public opinion about automated vehicles in Australia: Results from a large-scale national survey. *Transportation Research Part A: Policy and Practice*, 129, pp. 1–18. doi: 10.1016/j.tra.2019.08.002.

[5] SAE (2021). J3016C: *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles - SAE International.* [online] www.sae.org. Available at: https://www.sae.org/standards/content/j3016_202104/.

[6] Bagloee, S.A., Tavana, M., Asadi, M. and Oliver, T. (2016). Autonomous vehicles: challenges, opportunities, and future implications for transportation policies. *Journal of Modern Transportation*, 24(4), pp. 284–303. doi: 10.1007/s40534-016-0117-3.

[7] GOV.UK (n.d.). *Self-driving vehicles listed for use in Great Britain.* [online] Available at: https://www.gov.uk/guidance/self-driving-vehicles-listed-for-use-in-great-britain.

[8] The Engineer. (n.d.). *The Engineer - Comment: Autonomous vehicles are coming, is UK law ready?* [online] Available at: https://www.theengineer.co.uk/content/opinion/comment-autonomous-vehicles-are-coming-is-uk-law-ready/ [Accessed 16 Apr. 2023].

[9] GOV.UK (n.d.). *General rules, techniques and advice for all drivers and riders (103 to 158) - The Highway Code - Guidance - GOV.UK.* [online] Available at: https://www.gov.uk/guidance/the-highway-code/general-rules-techniques-and-advice-for-all-drivers-and-riders-103-to-158#rule149.

[10] GOV.UK (n.d.). *Introduction - The Highway Code - Guidance - GOV.UK.* [online] Available at: https://www.gov.uk/guidance/the-highway-code/introduction.

[11] Pinsent Masons. (n.d.). *New legislation crucial to retain UK's leading position in self-driving car industry.* [online] Available at: https://www.pinsentmasons.com/out-law/news/new-legislation-crucial-to-retain-uks-leading-position-in-selfdriving-car-industry.

[12] GOV.UK (2022). *Self-driving revolution to boost economy and improve road safety.* [online] Available at: https://www.gov.uk/government/news/self-driving-revolution-to-boost-economy-and-improve-road-safety.

[13] BBC News. (2022). *Robot grocery delivery service launches in Leeds.* [online] 30 Nov. Available at: https://www.bbc.co.uk/news/uk-england-leeds-63809836.

[14] UKTN | UK Tech News. (2023). *Starship expands delivery robots to Greater Manchester.* [online] Available at: https://www.uktech.news/mobility/starship-delivery-robots-greater-manchester-20230316 [Accessed 16 Apr. 2023].

[15] Tennant, C., Stares, S. and Howard, S. (2019). *Public discomfort at the prospect of autonomous vehicles: Building on previous surveys to measure attitudes in 11 countries.* Transportation Research Part F: Traffic Psychology and Behaviour, [online] 64, pp. 98–118. doi: 10.1016/j.trf.2019.04.017.

[16] Stilgoe, J. (2022). *Survey reports.* [online] Driverless Futures? Available at: https://driverless-futures.com/2022/05/09/survey-reports/.

[17] Lydall, R. (2023). *1,600 miles, 0 crashes: Three-year London test run of driverless cars concludes.* [online] Evening Standard. Available at: https://www.standard.co.uk/news/transport/nissan-driverless-cars-test-woolwich-london-b1060935.html [Accessed 16 Apr. 2023].
[18] Othman, K. (2021). *Public acceptance and perception of autonomous vehicles: a comprehensive review. AI and Ethics*, pp. 355–387. doi: 10.1007/s43681-021-00041-8.
[19] Cugurullo, F. and Acheampong, R.A. (2023). *Fear of AI: an inquiry into the adoption of autonomous cars in spite of fear, and a theoretical framework for the study of artificial intelligence technology acceptance.* AI & SOCIETY. doi: 10.1007/s00146-022-01598-6.
[20] Dagan, E., Mano, O., Stein, G.P. and Shashua, A. (2004). *Forward collision warning with a single camera.* [online] IEEE Xplore. doi: 10.1109/IVS.2004.1336352.
[21] Jansson, J., Johansson, J., & Gustafsson, F. (2002). *Decision Making for Collision Avoidance Systems*. SAE Transactions, 111, 197–204. http://www.jstor.org/stable/44699413
[22] Car ADAS. (2021). *Understanding ADAS: Backup Camera Systems & Related ADAS Systems.* [online] Available at: https://caradas.com/backup-camera/.
[23] de la Escalera, A., Armingol, J.Ma. and Mata, M. (2003). *Traffic sign recognition and analysis for intelligent vehicles. Image and Vision Computing,* [online] 21(3), pp. 247–258. doi: 10.1016/s0262-8856(02)00156-7.
[24] GOV.UK (n.d.) *The Highway Code.* [online] GOV.UK. Available at: https://www.gov.uk/guidance/the-highway-code.
[25] GOV.UK (n.d.). *The Highway Code, road safety and vehicle rules - GOV.UK.* [online] Available at: https://www.gov.uk/browse/driving/highway-code-road-safety.
[26] GOV.UK (n.d.). *Updates - The Highway Code - Guidance - GOV.UK.* [online] Available at: https://www.gov.uk/guidance/the-highway-code/updates.
[27] Patterson, S., Farrer, J. and Sargent, R. (1988). *Automotive Head-Up Display. SPIE Proceedings.* doi: 10.1117/12.947724.
[28] Xiao, L. and Gao, F. (2010). *A comprehensive review of the development of adaptive cruise control systems. Vehicle System Dynamics,* 48(10), pp. 1167–1192. doi: 10.1080/00423110903365910.

# Appendix A

## Explanations of beliefs, intentions and recommendations in human-understandable language

| *State* | *Explanation* |
|---|---|
| **Beliefs** | |
| *seenSign* | A road sign was spotted. |
| *signNoConflictsWithAuthorisedPersons* | There are no authorized persons whose signals prioritize road signs, |
| *exceedingSpeedLimit* | The speed limit was exceeded. |
| *cantStopBeforeCarInFrontStops* | It is impossible to stop before the car in front does. |
| *cycleLaneAvoidable* | The cycling path is avoidable. |
| *pavement* | The car is on a pavement, |
| *notAccessProperty* | The car is on a property with forbidden access. |
| *pedestriansInRoad* | There are pedestrians on road ahead. |
| *stopSign* | A road sign is a stop sign. |
| *whiteLineAcrossRoad* | There is a white across line ahead. |
| *giveWaySign* | A road sign is a give way sign. |
| *dottedWhiteLineAcrossRoad* | There is a dotted across line ahead. |
| *lightRed* | A traffic light is red. |
| *lightAmber* | A traffic light is amber. |
| *ableToStopByWhiteLine* | The car is able to stop by white line. |
| **Intentions** | |
| *brake* | The driver wants to brale. |
| *beInCycleLane* | The driver wants to enter a cycling path. |
| *approachingTrafficLight* | The driver is approaching traffic lights. |
| **Recommendations** | |
| *must-consideration_others* | You must take others into consideration. |
| *must-drive_care_attention* | You must pay attention. |
| *must-not_drive_dangerously* | You must not drive dangerously. |
| *should-give_way_other_roads* | While turning, you should give way to other roads users. |
| *should-keep_left* | While turning, you should keep left. |
| *must-follow_sign* | You must follow the sign. |
| *should-brake_early_lightly* | You should break early and lightly. |
| *must-reduce_speed* | You must reduce speed. |
| *should-increase_distance_to_car_infront* | You should increase distance to car infront. |
| *must-use_road* | You must use road. |
| *must-avoid_parking* | You must avoid parking here. |
| *must-road_surfaces-avoid_non* | You must avoid non-road surfaces. |
| *should-give_way_to_pedestrians* | You should give way to pedestrians. |
| *must-stop_at_white_line* | You must stop at white line. |

| | |
|---|---|
| *should-wait_for_gap_before_moving_off* | You should wait for gap before moving off. |
| *must-give_way_at_dotted_white_line* | You must give way at dotted white line. |