Bachelor's thesis

Information and Communications Technology

2021

Karl Lahdenranta

AI BEHAVIOR IN EQUAL INTERSECTIONS

– Modification of Simulandia traffic AI

TURKU AMK

TURKU UNIVERSITY OF APPLIED SCIENCES

BACHELOR’S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information and Communication Technology

2021 | number of pages: 33, number of pages in appendices: 1

Karl Lahdenranta

AI BEHAVIOR IN EQUAL INTERSECTIONS

- Modification of Simulandia traffic AI

When creating a simulation of any sort, one of the most important aspects is immersion and when the player is put into a completely lifeless environment it is rarely very immersive. To solve these issues developers cannot simply plant people, animals, etc. to our simulation, these objects also need to interact with the player and the interactions should feel real. This is where we need to implement AI as part of our simulation.

AIs are usually fairly complex because a simple AI is often not enough to create an illusion of interacting with a living being. This complexity can, unfortunately, become an issue when there is a need to alter existing AI behavior.

This thesis will discuss methods for incorporating new behavior for existing AI, specifically iTS Intelligent Traffic System (Unity asset) used by TTS Työtehoseura ry’s Simulandia project (made by Turku University of Applied Sciences and ADE Oy) and the new behavior being comprehension of equal junctions.

Although iTS has many build-in options for modifying AI behavior, it has no understanding of equal intersections. This will increase the complexity of our problem, as our task changes from simply modifying existing behavior to creating a completely new one and incorporating it as a functioning part of the AI. The AI consists of multiple fairly large scripts, which also brings a problem of accidentally creating new bugs and unwanted behavior when editing said files. For this reason, the thesis will partly cover how the AI works currently and lists some possible issues that may arise when trying to modify it in a certain way.

This thesis presents three possible methods on how to implement equal intersections to the driving environment and discusses strengths, weaknesses, and optimal integration environments for each method. All methods are then compared on which one would be the best choise to implement to the current version of Simulandia driving environments and what possible aspects these methods have for further development.

KEYWORDS:

simulation, AI, game engine, traffic, programming

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tieto- ja viestintätekniikka

2021 | 33 sivua, 1 liitesivua

Karl Lahdenranta

TEKOÄLYN KÄYTTÄYTYMINEN TASA-ARVOISISSA RISTEYKSISSÄ

- Simulandia-liikennetekoälyn muokkaus

Minkä tahansa simulaation eräs tärkeimmistä ominaisuuksista on immersio, ja täysin eloton ympäristö on hyvin harvoin kovin immersiivinen. Ongelman ratkaisemiseksi kehittäjät eivät voi vain lisätä ympäristöihin ihmisiä, eläimiä yms., vaan kyseisten objektien tulisi myös reagoida pelaajaan ja reaktioiden olisi tunnuttava mahdollisimman luonnollisilta. Tällaisessä tapauksessa on tuotava tekoäly osaksi projektia.

Tekoälyt ovat usein hyvin monimutkaisia, sillä yksinkertainen tekoäly pystyy harvoin luomaan illuusion kanssakäymisestä elollisen olennon kanssa. Tämä monimutkaisuus voi kuitenkin koitua suureksi ongelmaksi projektin mahdollisessa jatkokehityksessä, kun tekoälyn käyttäytymistä halutaan muokata.

Tässä opinnäytetyössä tarkasteltiin mahdollisia metodeja uusien käyttäitymismallien lisäämiseksi valmiiseen tekoälyyn. Kyseessä oleva tekoäly on TTS Työtehoseura ry:n Simulandia-projektin (Turun ammattikorkeakoulun ja ADE Oy:n luoma projekti) liikennetekoäly iTS Intelligent Traffic System (Unity lisäosa). Lisättävä käyttäytymismalli, johon opinnäytetyö keskittyy, on autojen käyttäytyminen tasa-arvoisissa risteyksissä.

Vaikka iTS sisältää useampia sisäänrakennettuja työkaluja liikenteen käyttäytymisen muokkaamiseen, ei tämä tekoäly tunne tasa-arvoisten risteysten konseptia. Tämä vaikeutti tehtävää huomattavasti, sillä tekoäly ei sisällä pohjaa jonka ympärille voisimme rakentaa oman käyttäytymismallimme. Kyseessä oleva tekoäly koostuu useasta kohtuullisen suuresta tiedostosta, mikä vaikeuttaa sen muokkaamista ilman toiminnassa olevien ominaisuuksien häiritsemistä. Tästä syystä opinnäytetyö kesittelee osittain myös iTS-tekoälyn yleistä toimintaa ja mahdollisia ongelmia sen muokkaamisesta.

Opinnäytetyössä esitetään kolme mahdollista tapaa tasa-arvoisten risteyksien lisäämiselle ja käydään läpi niiden vahvuudet, heikkoudet ja optimaaliset käyttöympäristöt. Lisäksi verrataan mikä kyseisistä metodeista olisi parhaiten sopiva nykyiseen Simulandia-ajoympäristöön ja mitä mahdollisia jatkokehitysideoita opinnäytetyön aikana tuli esille.

ASIASANAT:

simulaatio, tekoäly, pelimoottori, liikenne, ohjelmointi

CONTENT

LIST OF ABBREVIATIONS (OR) SYMBOLS

AI	Artificial intelligence
iTS	Intelligent traffic system
iTSTA	iTS Traffic AI
iTSMM	iTS Main Manager
iTSTS	iTS Traffic Spawner
VR	Virtual reality

1 INTRODUCTION

The purpose of this thesis was to find ways to improve iTS (Intelligent Traffic System) traffic AI used in TTS Työtehoseura's Simulandia project (made by ADE Oy and Turku Game Lab), more specifically the AI cars behavior in equal intersections. The goal was not necessarily to create a working system but compare different solutions and make suggestions on what type of changes to iTS AI would be most beneficial to Simulandia.

The original interest to study the addition of equal intersections to Simulandia driving simulations came from a larger list of possible features that could be used to enhance the overall simulation experience. Equal intersections were chosen because out of all the considered subjects it was more geared towards enhancing the already existing experience rather than adding completely new functionalities.

In the theory section of this thesis, we discuss the necessary logic of AI and traffic rules that will serve as a foundation for the development of equal intersection systems. We also familiarize ourselves with the iTS, Intelligent Traffic System, by going through all the components that will be relevant to the development process and list missing features that would be needed for the implementation of equal intersections.

In its original state iTS AI can not apply right of way rules on its own, they have to be set beforehand on each intersection. All possible intersections created with this method would be variations of a junction to the main road. This method could not work on its own to create equal intersections since the right of way not only depends on the turn a car is about to make but also on other cars present at the intersection. The inclusion of equal intersections would not only add a new element to the list of usable building blocks when creating a driving environment but the logic could possibly be used further in other road scenarios.

To achieve the correct type of behavior for AI cars in equal intersections we have to first decide how we want to approach the process of modifying the AI. The logic of equal intersections would require two main parts to function correctly: a way for cars to pass info, and a system to process this info and give the right of way correctly. The first part would heavily depend on what type of info processing system is created and would more likely be a factor in optimization, so we will focus more on the actual info processing systems. In game technology the system that keeps track of various types of data is called "a manager". These managers can control multiple objects or they can be placed as a part of a singular object. In this thesis, we propose two separate methods using both of these styles and one additional style to cover as many types of possible driving environments and types as possible. All methods were tested in a versatile city environment and we will take a look at how each one affects the overall traffic flow and performance of the simulation.

After all of the methods have been presented we will discuss the strengths, weaknesses, and optimal integration environments for all of them, and draw a conclusion on which method would be suited best for the current version of Simulandia driving environments.

2 THEORETICAL BACKGROUND

2.1 AI

Before we start to go through iTS AI and its functions, we should discuss the definition of AI and its different variations. The term “artificial intelligence” refers to a machine’s ability to mimic characteristics that humans associate with human mind’s intelligence, such as learning and improvising. Any machine that takes actions based on its environment and increasing its chances of achieving its goals shows the behavior of an AI. [1]

The three main cognitive skills of AI are learning, reasoning and self-correction. All of these skills are AI’s methods of using its algorithms to achieve its goals. Learning is a process where AI gathers information about its surroundings and turns it into usable data for the algorithms available. [2] Reasoning refers to the state where AI chooses the correct algorithms for the task in question. Self-correction does not affect the current task the AI is trying to complete but is meant to alter its algorithms so that similar tasks in the future can be completed more efficiently.

As AIs become more complex, some functions that were considered needing intelligence have been removed from list of tasks that require AI. One of such functions is character recognition. This pattern is called the AI effect and is also catch from Tesler’s theorem: “AI is whatever hasn’t been done yet.” By this standard, it could be said that video games do not have true AI, which is true in most cases. [3]

Game AI is focused on enhancing the player experience rather than attemting to show human-like intelligence and is by this definition always held back. [4] Games also favor simple AIs to achieve better overall performance. When designing an AI for a game (or simulation) it is necessary to assess first how simple can we make the AI while maintaining an immersive gaming experience. [5]

All of this taken into consideration, it can be said that iTS AI is much closer to being game AI rather than a true AI, and therefore terms regarding AI will more specifically refer to game AI.

2.2 Traffic systems

Traffic laws and etiquette depends heavily on which country they apply to. Simulandia being a Finnish simulation envinronment, we will naturally follow regional traffic practice and our subject being quite narrow – equal intersections – the amount of variables we need to consider for real-life perspective is very manageable.

We have three main situations to go through to create a realistic representation of equal intersections: [6]

1. Priority to the right
2. Priority to going straight
3. Exception

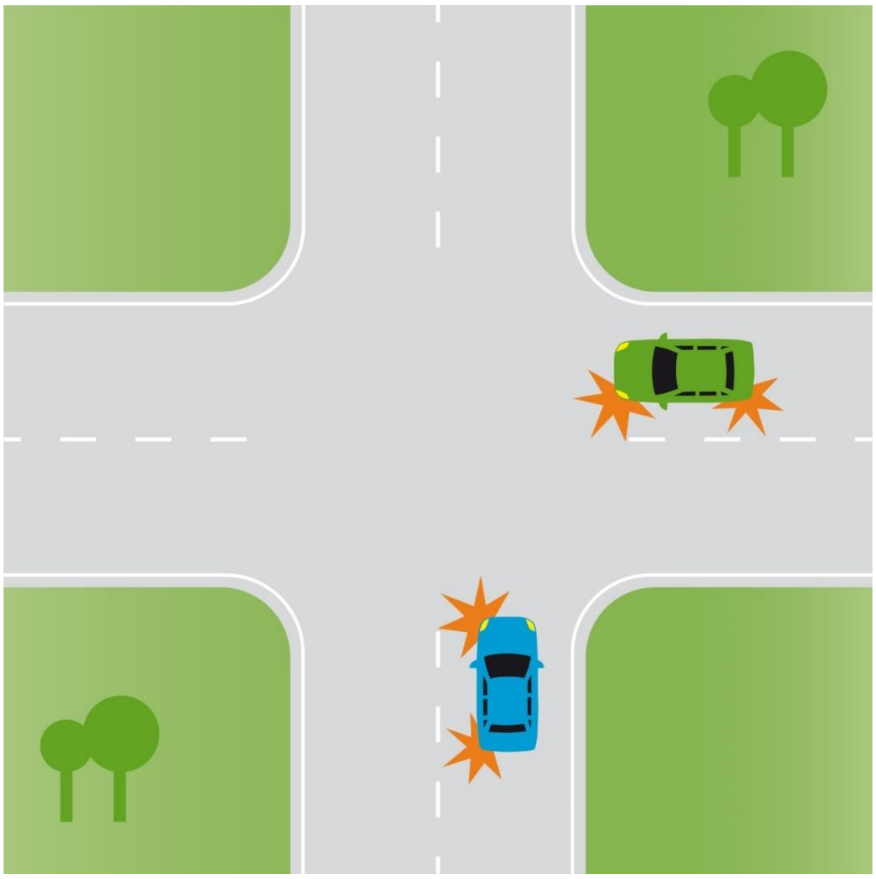

Figure 1. Priority to the right [7]

In Figure 1 above we have an example of case 1: priority to the right. In this situation, the blue car is obligated to let the green car go first since it is approaching the intersection from blue car's right side.

In Figure 2 the priority is to go straight. Here the two cars entering the intersection are driving on parallel lanes and therefore our first rule will not affect either of them. In this situation, the green car is obligated to wait until the orange car has passed the intersection. Keep in mind that these rules only stand if there is no right of way on any of the lanes like main road etc.

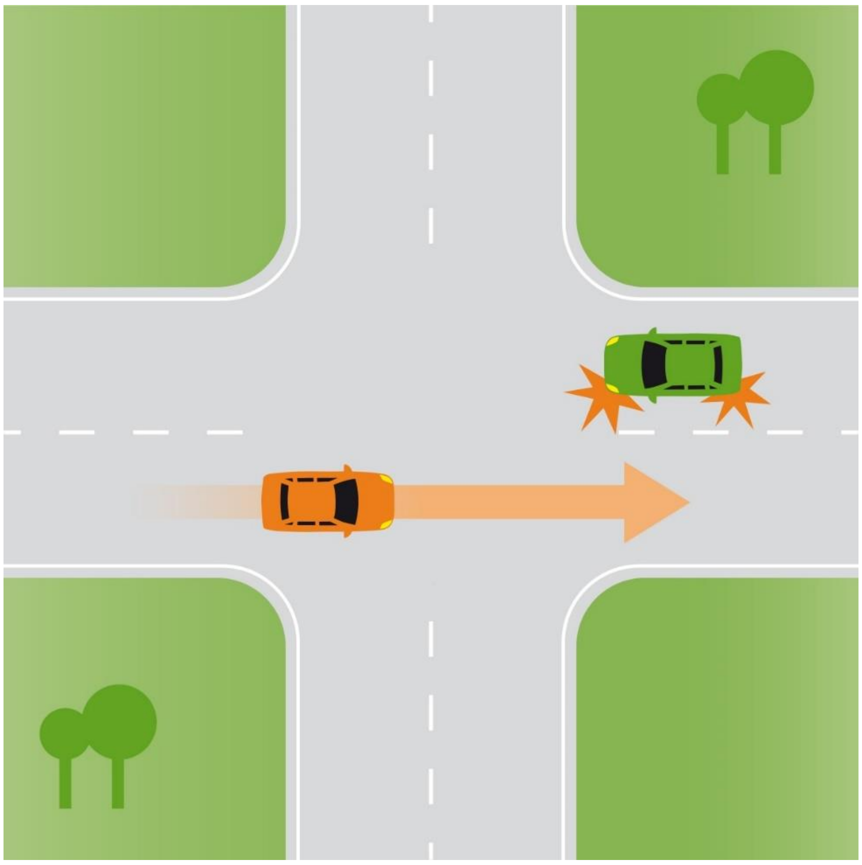

Figure 2. Priority to go straight [7]

The last case we need to be able to recreate in the simulation, is easily the most complicated one and that is an exceptional situation. In this situation the cars enter an intersection in a way that the first two rules override each other creating an unresolvable situation, where one of the cars is forced to break our previous rules. In Figure 3 we have a very simple example of this type of situation. The orange car should be giving way to its right, the blue car. The blue car should be giving way to its right too, the green car. Lastly, the green car is not allowed to go first either since it is obligated to give way to the traffic going straight, in this case, the orange car.

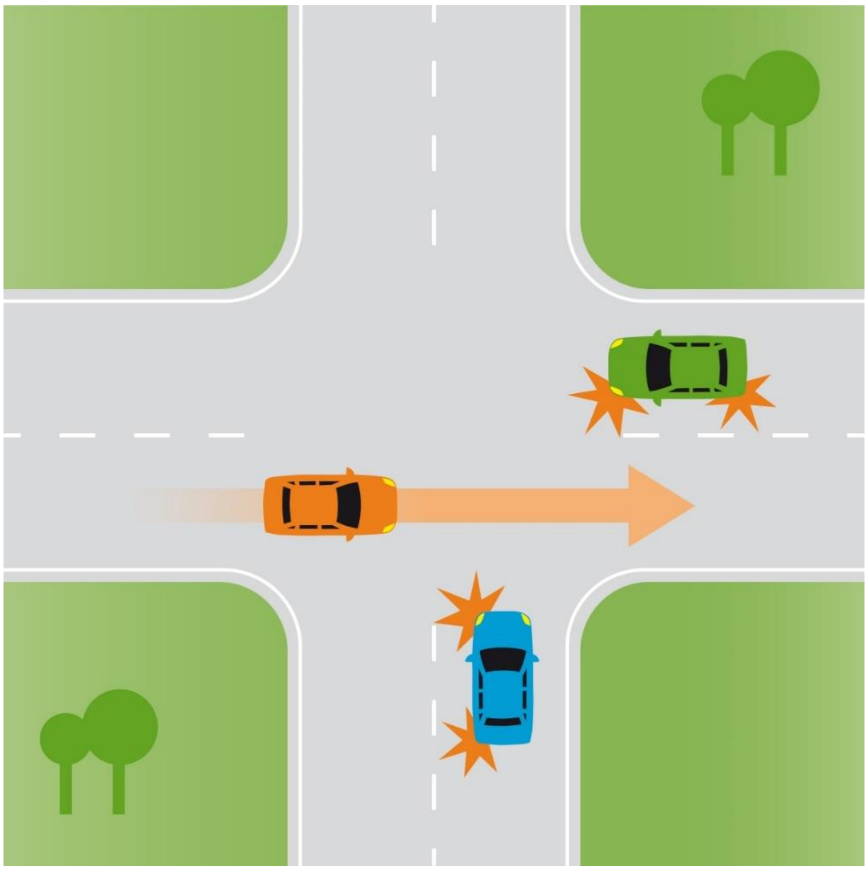

Figure 3. Unresolvable situation [7]

The correct order of cars, according to Finnish driving instructor Erkki Savolainen, would be that the car driving straight (orange) would pass the junction first. This by no means should be interpreted that in similar situations the car going straight has the right of way. [7]

2.3 Simulating realistic behavior with AI

The main goal of this thesis is not only to find a solution so the AI cars can behave relatively correctly in equal intersections, it is also to make them seem less like machines and more like fellow drivers. If the AI car's behavior is too regular it will not feel realistic and since all crashes etc. are counted as player errors, the AI needs to follow traffic rules most of the time so the driving experience stays engaging but does not become frustrating or too unrealistic. Even though it is not very difficult to make the AI cars obey all traffic rules and act very considerately towards the player it wouldn't create a good driving experience in the context of learning how to act in traffic. The only time when AI should not make any mistakes is when the player is not present and the AI is driving on its own.

Adding before-mentioned behavior patterns is very difficult when using a before-made AI. Currently, the AI cars of iTS have no behavior profiles and act identically regardless of the vehicle type, excluding limitations that the vehicles have because of their size (e.x. buses never drive on small roads). Because iTS is lacking this type of behavior system, methods studied in this thesis will keep AI car behavior differences to a minimum.

3 ITS – INTELLIGENT TRAFFIC SYSTEM

Before discussing methods of modifying the iTS car AI, we should go through the basic functions and limitations that the original version of the AI has. [8]

3.1 General behavior/use

The AI of iTS Unity asset consists of roughly twenty scripts that work in unison. Most of these scripts do not interact with each other but rather have three “master scripts” that serve as a base for communication between the rest of the scripts. Most of the modifications that have happened during the making of this study have been made to these three scripts. The rest of the changes have been minor and lean more to the side of being fine-tuning.

“ITS Main Manager” (iTSMM) is the main component of iTS system, and holds all information of lanes, connectors and their subsystems and passes this information to the other two main scripts.

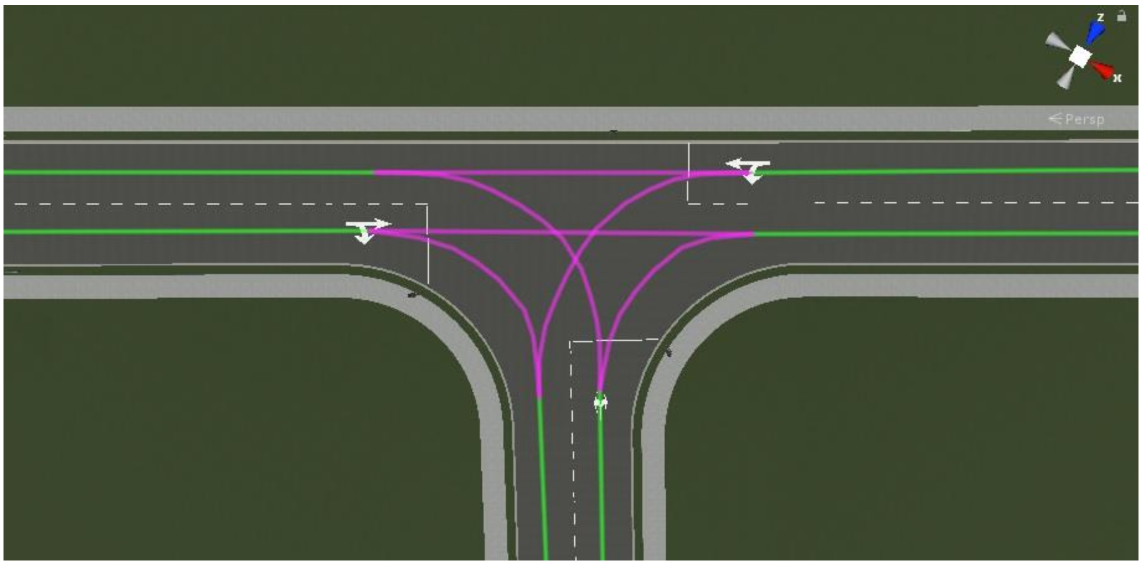

Figure 4. iTSMM lanes in the editor view

In Figure 4 we have a visualization of partial iTSMM lane (green lines) and connector (purple lines) data. Visually lanes and connectors are represented as lines but as data element, they are considered as a list of points. Lanes’ and connectors’ general logic is very similar and the whole network could be created by using only connectors, but the added amount of data of this would have a huge impact on the simulation performance. The main reason why lanes are not as performance-heavy is that iTS assume that lanes will never cross each other. Therefore when cars more on lanes they do not have to be wary of cars of other lanes blocking their way.

Lane data contains info like speed limit, lane density, and width. It also defines what type of vehicles are allowed to travel on their path. Another main function of ITSMM is lane linking, which enables cars to overtake each other and pass an object that blocks their current lane.

“ITS Traffic Spawner” keeps track of the functionality of each car, specifies areas the AI cars are allowed to use and stores unused cars (to prevent traffic jams, etc.). Traffic spawner does not hold any type of specific info of the AI cars and communicates with only a few other scripts (compared to ITSMM and ITSTA that work with multiple scripts and pass a lot of data between each other).

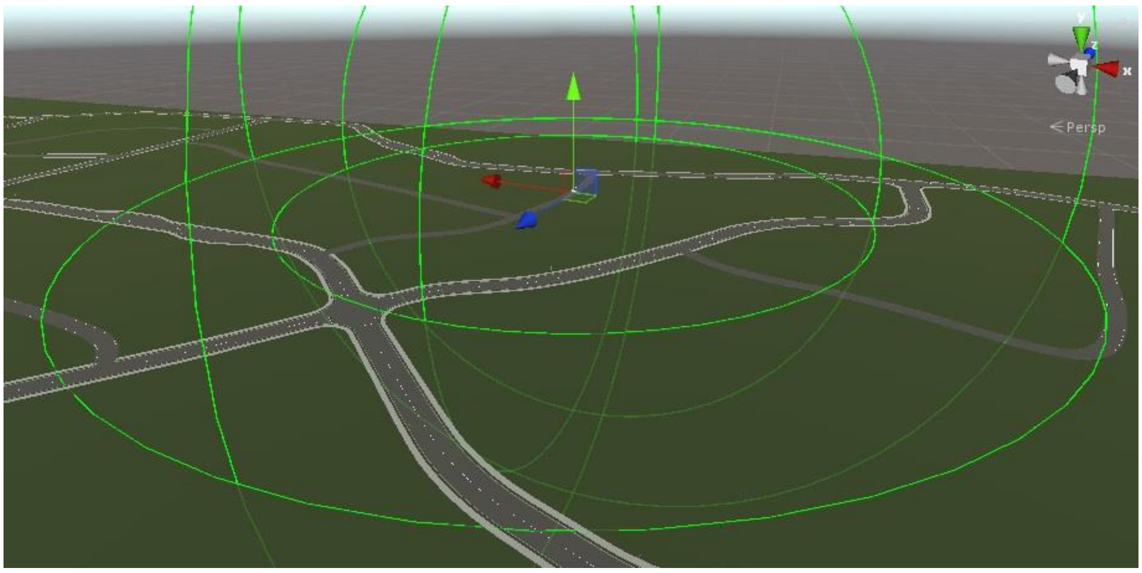

Figure 5. iTSMM AI car spawn area

Figure 5 shows the standard spawn area system that is used in iTS asset. AI cars spawn only between the two green circles and despawn if they move out of the range of the outer circle. This is to prevent the player from seeing the spawning and despawning of the AI cars.

3.2 Lane reservation in iTS

As mentioned in paragraph 3.1 lanes are coded as lists of points. Each car has three key point types in this system: waypoint, reserved points, and reserver point. Reserved points can also be divided into two categories: critical and non-critical. Critical points are reserved points that cover the area of the cars physical game object and they cannot be removed by other cars or the traffic system. Non-critical points are points that have been reversed according to calculations where the car will be in a couple of seconds. These points can be removed e.x. by a traffic light turning red.

Reserver point is considered as a non-critical point and it will always be the furthest point in front of the car. Systemwise the main difference of reserver point is that it can overlap other points. This is very useful when testing different lane reservation situations.

Waypoint is one of the reserved points and also a critical one. It is the only point that is directly communicating with the car’s game object. The purpose of the waypoint is to represent the exact location where the car’s game object is currently heading.

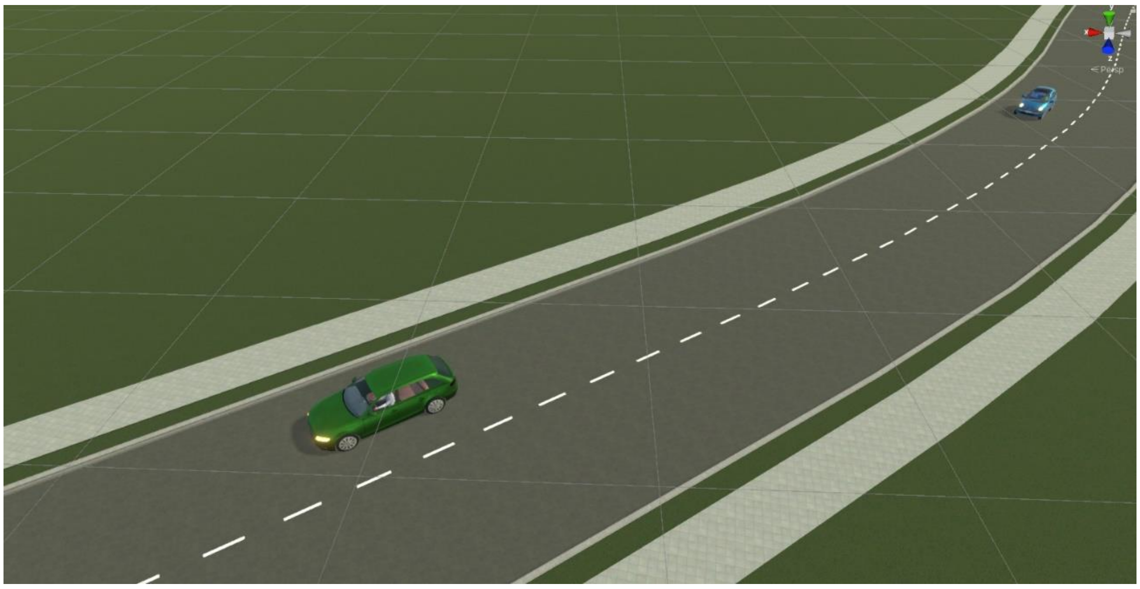

Figure 6. AI cars in editor view with no gizmos

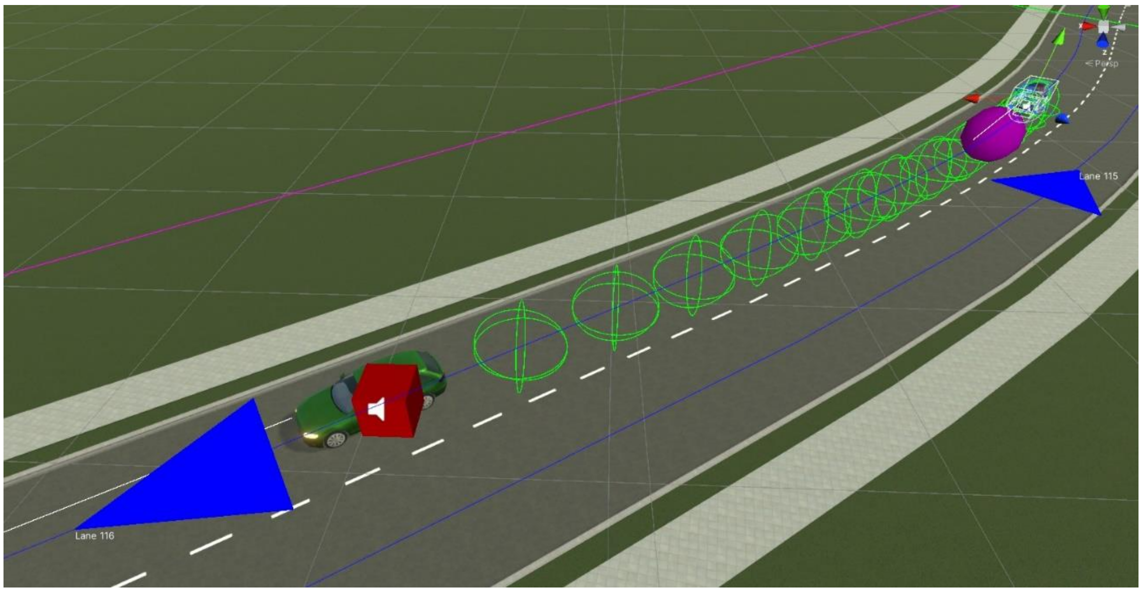

Figure 7. AI cars in editor view with gizmos

In Figure 6 we have a standard editor view of two cars driving on a road in simulation. Figure 7 has been taken from the exact same situation except it includes the visual debug information of the reserved points. Waypoint is highlighted as a magenta sphere and reserver point as a red cube. These types of visualization objects are called gizmos in Unity. Reserver points gizmo will switch color depending on the reservation status of the points ahead of it. The rest of the reserved points are drawn as light green colored wire spheres.

3.3 Problematic/missing features

Before starting to discuss different solutions on how to implement equal intersections in iTS we need to take a closer look at some of the key features that the system lacks and what will affect the planning stage of our development.

The first issue we have to tackle is the communication between cars. Currently, some connectors can have priority over the other but these values are static and only work if all involved cars are already waiting at a junction. In essence this system works as follows:

1. Two cars are waiting in a junction and are trying to reserve connector points from connectors that overlap each other
2. Eventually, these cars will try to reserve connector points that are too close to each other (or even overlap)
3. Since connectors can have priorities the connector point in question is given to the car that is on a higher priority connector.
4. The second car will have to wait until the first one has unreserved the before mentioned connector point.

In the case of equal intersections, this system cannot be used as such, since the priorities of cars depend on the other cars in equal intersections. For example, in Figure 4 the connector priority is given to the straight connectors but in an equal intersection, this would not work, as demonstrated earlier by the situation in Figure 3. Therefore our solution on equal intersections cannot utilize the existing priority system of iTS, at least not in its current form. A system similar to this could be created if cars were able to determine connector priorities depending on their own entry point to the intersection and pass this information to other cars.

The next two limitations we need to take in consideration are route memory and unreserving connector points. Latter already exist in iTS but has very limited uses and the former is completely neglected in the current form of the traffic system. Unreserving connector points will also be very closely tied to the first issue of communication between cars and will likely stay as the most relevant (and maybe even only) piece of information that the cars need to pass between each other.

Fortunately, route memory is not a vital component in an equal intersection system since even if connectors can overlap each other, they always have only one direction and cannot branch further. There are situations where route memory could serve a purpose, e.x. when the connector’s first point would overlap with another connector and a car would have to unreserve all of its connector points and therefore also “forgetting” what connector it was trying to reserve. All of this can be avoided with careful construction of lanes and connectors, so instead of creating a memory system for equal intersections, it’s better, for now, to plan the lanes and connectors in a way that this will not become an issue.

Unlike the route memory, unreserving points is a feature that cannot be neglected no matter how the lanes and connectors are built. As mentioned in paragraph 3.2 some points reserved by cars can be unreserved, but the existing function for this needs a static point on the map to work. A very similar problem that we have with the connector priority. Not only is a function hard to switch to utilize variables compared to static values, not to mention multiple variables sent between multiple moving objects, but to keep said function from becoming too performance heavy.

The function iTS uses for unreserving points is tied mainly to junctions with traffic lights. When a car has to give up on some of its connector points the points are unreserved until the car has no more points on the connector where to unreserving process started i.e. all of the reserved points will be lane points. This system can work in equal intersections on the surface but could lead to confusing situations when observed from the outside. An example situation of this could be a variation of Figure 2 where the green car would have to give priority to the orange car, but afterward it would turn right completely negating the need for priority check in the first place and it could cause further confusion on the cars of other lanes if the traffic would be more substantial.

4 METHODS OF MODIFYING AI

Like in most cases of coded systems that consist of a network of different scripts the most difficult part of adding new features is not the actual creation process of said functions, but to make sure that the already existing features will not receive any unintended side effects. In the case of iTS this is extremely important since the simulation is supposed to be of educational use and is grading the users according to their actions in the environment. A badly implemented function could skew the scoring that is given to the user and in the worst case create unwanted reactions to similar situations in its real life counterpart. Therefore the most important aspect of these solutions will be their probability of not affecting anything outside of their jurisdiction.

4.1 Modifications to TTS Työtehoseura driving simulator

Like mentioned before, the most important part of any modification is its ability to not affect anything outside of its given area. The current version of iTS used in Simulandia has not been given any drastic changes, outside of gathering data on users' behavior, compared to the original version but is more of a refined version of the existing asset. As such the inclusion of equal intersections would easily be the biggest single addition to the iTS.

In this section, we go through three different solutions that tackle different combinations of lacking features mentioned in paragraph 3.2. The solutions are listed in order from the most likely to cause issues to the most simple. Of course, this also means the most realistic to the most unrealistic. The performance costs and effects on the traffic flow of each solution will be discussed with test results after we have familiarized ourselves with all of these cases.

4.2 Method A (Cars share info between each other)

The option that would need the least amount of changes to the existing code, when counting as new lines of code. In this solution, we simply create an extra method for identifying reserved connectors and the car ID that reserved them. After this, we would determine which car should go first and the seconda car would have to wait. This method would require a solution for two of the limitations that iTS has. Firstly, communication between cars, and secondly a method of unreserving points without removing all connector points from the current reserved point list. Both of these solutions however they are made would likely have the biggest likelihood of causing bugs considering the alteration of the already existing system and implementation of a completely new one.

Communication between cars would need a new system for iTS and route memory even if implemented could cause more issues because the AI navigation system works extremely inconsistently when connectors are reserved only partially. Without any additions or modifications to this systems, cars can only reset their driving routes and get a completely new one. On the surface level it would not matter even if the cars change their destination after giving way to another car, assuming they will not change the turn they are making in the junctions, but multiple cars arriving into the junction simultaneously could cause issues that would require further testing and development. This issue was also mentioned in paragraph 3.3.

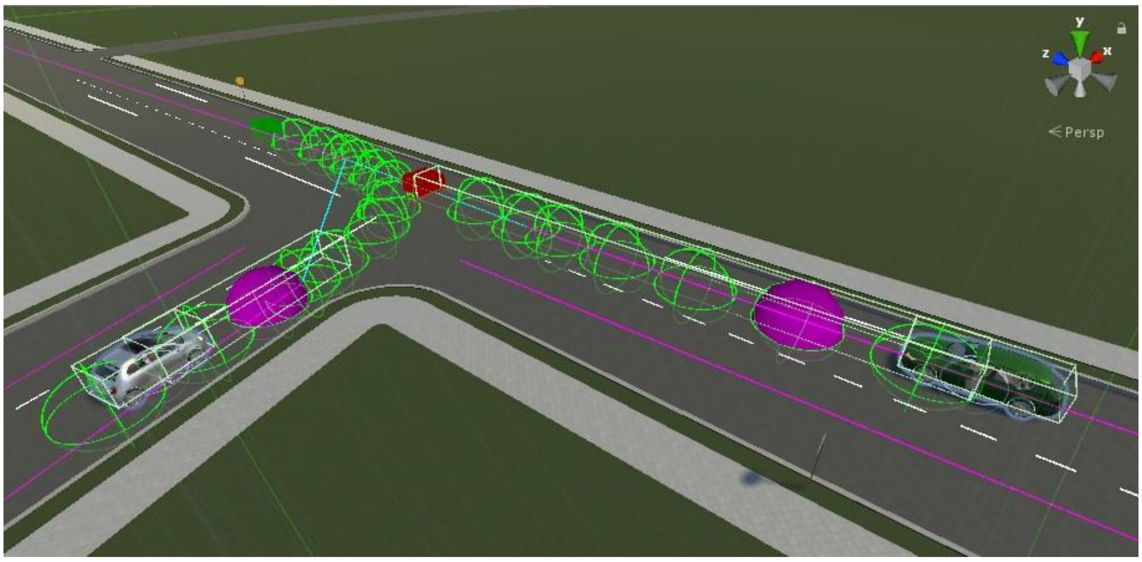

Figure 8. Lane reservation visualized with gizmos in the editor view

In Figure 8 we have a situation that can happen in the current version of iTS traffic AI. The grey car should give way to the incoming traffic from the right, but the lack of priorities makes the cars operate currently on basis of first come, first served. Of course, these types of violations could also be avoided by setting main road statuses for certain roads but these intersections could not then be considered as equal intersections.

The logic the cars in Figure 8 will follow with method one:

1. The grey car enters the intersection normally and reserves the needed connector points
2. The green car enters the intersection and tries to reserve connector points to advance straight through
3. The green car receives values that tell if the necessary connector points are reserved and the ID of the car that reserved them
4. Green car checks if it should have the priority
5. If the green car has priority it sends the grey car’s ID to the AI car manager and requests this car to unreserve connector points

All of this sounds pretty straightforward and simple, but determining priorities between two cars that arrive at a intersection is the simplest example of how the system works. In the actual simulation, there will be situations where multiple cars would have priority over another car (e.x. Figure 3). The problem of looping commands of giving or receiving priorities becomes very apparent.

4.3 Method B (System that oversees car behavior)

Our second option for determining priorities is to create an observation system that registers cars as they enter intersections. These types of “big brother” objects are set in every intersection and work separately from each other. Every car that enters a junction will be set to queue where they wait for permission to proceed further in said junction.

This solution logic is a modification of already existing traffic light system. Every lane of every intersection will have an extra traffic light that is invisible to the player. These traffic lights are red by default and will be changed to green for each car on the queue. While proving to be quite stable, this solution is by far the most performance heavy out of all the suggested solutions. Not only because we would have an extra system on guiding the AI cars through the intersections but also for needing to add this system for every intersection separately (excluding the intersections that are not equal). Another downside for this system is that it does not function very well when used in unison with the existing traffic light system.

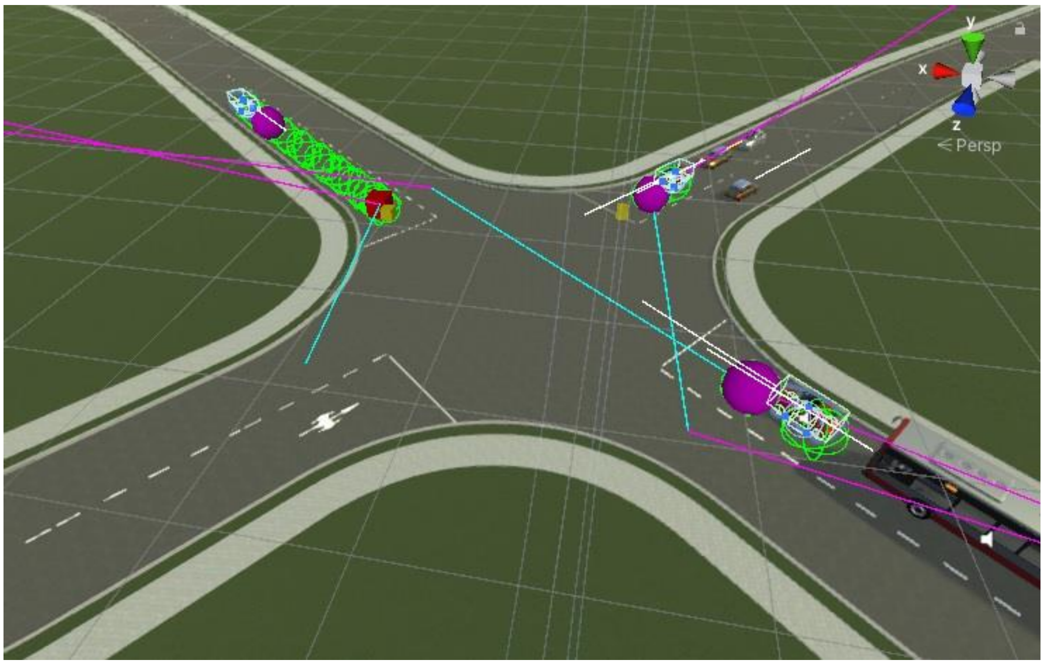

Figure 9. AI car driving destinations visualized with gizmos in the editor view

Figure 10. Same situation as in Figure 9 without gizmos

The way the equal intersections priority system works is visualized in Figure 9. In this situation, we have three cars waiting to pass an intersection. Figure 10 is taken from the exact same situation, but we have left out Unity gizmos for better overall visibility of the area and vehicles involved. By default, they all act like there was a red traffic light stopping them from entering the junction (also for purpose of taking this picture, no car is given priority to ensure multiple cars would wait at the same time at an intersection). When a car is reserving points the reserver point turns red when it reaches the last point of the current lane. After this it requests a path through a connector to the next lane, which is visualized with a cyan line. In Figure 9. The system then checks if any of the cyan lines cross each other and gives priorities accordingly.

In the following explanation, we will call the blue car in top left car A, the red car in the top right car B and the red car in bottom right car C. Priorities given to these three cars would therefore go as follows:

1. Route request from car A is not crossing paths with any other car, so it is given a right to enter the intersection.
2. Route requests of cars B and C cross each other and priority should be given to one of these cars.
3. System notes that car B is arriving in the intersection from car C’s right side, so it is given priority in this situation.

Afterward the same process would repeat as long as there are cars entering the junction. Unlike with method A, the problem of looping commands (of giving priority) is much easier to solve since we have a single object determining the order in which the cars would pass the intersection. For the traffic to bring the flow of cars to a complete halt, it would need no more than three cars with crossing route requests and similar arrival times to the entrance of the junction. This type of situation was already mentioned in paragraph 2.2 and could be used for the logic of solving the looping issue in this method, i.e. give priority to the car going straight if priority has been given four times without any cars passing the intersection. The only way for cars to create this type of loop without any one of them going straight, would for four cars to arrive at the same time to the junction and request a turn to the right. Even if such a combination of path requests would be highly unlikely to happen it could be prevented beforehand quite easily by giving the system special orders for such an occasion. Shortest would be to give the priority to the car that arrived at the junction first.

4.4 Method C (Only consider the player car)

Regardless of the nature of modifications that have been made to iTS throughout the development of TTS Työtehoseura driving simulations, the traffic system has not been the ideal AI system considering the ease of system changes. Another critical issue that has surfaced, not only with the AI traffic but also with the lane system, is iTS being relatively performance heavy on larger scenes. Conurbation areas roughly the size of 100 hectares will not suffer from any performance issues with iTS, but in larger areas, any additions of extra cost on update loops should be avoided. For this reason, the last method discussed will mainly focus on trying to keep performance costs as low as possible.

The main purpose of this thesis was to find the best way to implement equal intersection as a part of the city driving simulations environments and these simulations were mainly created for grading user's driving and observation skills in traffic. Even though creating as realistic a simulation as possible would be ideal, often some features will be unnoticed by the user of the simulation. Cutting said features can be a good way of cutting performance costs and keeping the overall simulation more manageable.

During test runs of various driving simulations, user feedback on AI car activity was in general always connected to the user or case of a singular misbehaving AI car. Because the AI cars operate with a hive mind type of control scheme, it is understandable that AI cars will not create peculiar situations with one another outside of bugs. For this reason, a possible solution to the implementation of equal intersections would be to only make them a function at junctions where the user is present. For grading the user's actions in traffic it is not strictly necessary for the AI traffic to abide by all the rules, especially if the user is not present. Thus method C will focus on losing as much complexity from the junction behavior as possible while maintaining the grading capabilities needed for equal intersections.

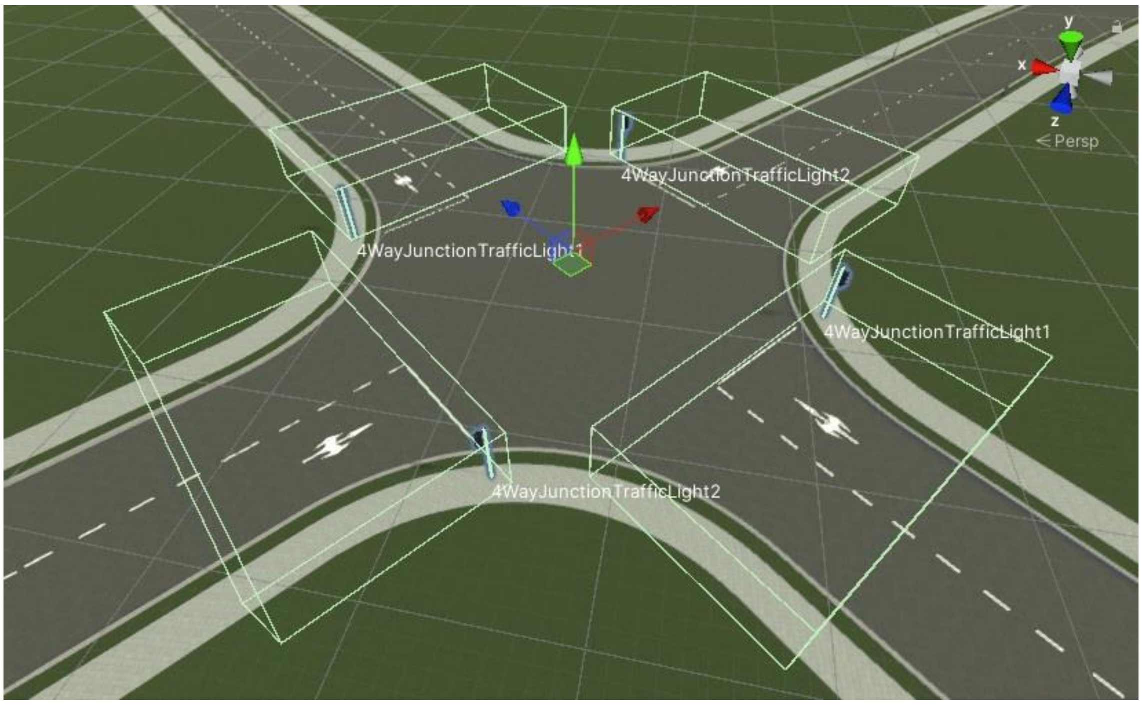
4WayJunctionTrafficLight2
4WayJunctionTrafficLight1
4WayJunctionTrafficLight1
4WayJunctionTrafficLight2
y
x
z
< Persp

Figure 11. Intersection scanners visualized with gizmos in the editor view

Figure 11 shows, with Unity gizmos, the general stucture of the intersection surveillance. The transparent, green-lined boxes represent scanners that register user car’s entrance and exit points when it drives through a junction. In Unity, these types of objects are called collider boxes. Data collection begins when the car enters one of these boxes and continues until it exits another. After this, the user’s action are graded depending if they used indicators correctly, obeyd traffic signs, and so on. Method C aims to only grade user’s actions and alter the behavior of AI cars that are nearby and would crash into the user car if no precautions are taken.

While the particular colliders in before-mentioned image are far too short for detecting any arriving cars, the system itself can be integrated quite easily as a part of a driving environment with some changes on the existing colliders. Another change that this method requires is a communication arrangement for the user to inform their chosen route to the AI cars. To solve this issue, we will be using the lane reservation system. Normally the lane reservation determinates where a car is heading, but this system can also be used in reverse to predict where a car would end up if it keeps its current course. This combined with the user cars indicator, to narrow down the path on connectors, we can easily inform the user’s route to the AI without any further extensions to the traffic system.

Method Cs equal intersection behavior works as follows:

1. User car enters the first collider. Indicator usage determines which connector would be reserved for the player.
2. Colliders that could contain cars with higher priority are checked.
3. User is graded for their actions.

Using the lane reservation system to inform AI cars of user’s chosen path can be considered either beneficial or harmful for the overall simulation. Since iTS has no understanding of equal intersections, unless it’s static i.e. main road, the AI will give way to the user if they have reserved connectors even when they do not have the right of way. Toggling reservations depending on the user braking do allow AI cars to pass but affect the overall flow of the traffic. Alternatively, the AI cars can be given a slightly longer reaction range and not be given any information on the users driving routes. This solution will force the user to be more observant but could lead to a crash even if the player makes no errors at all.

4.5 Effects on traffic (performance and traffic flow test)

While dicussing different solutions on how it would be possible to add equal intersections as a part of the TTS Työtehoseura driving simulations, we have mentioned few downsides for each of the methods and what was the main reason why this method was proposed in the first place. In this section, we take a slightly statistical approach to comparing these methods. We will study how each of these solutions affects the traffic flow and frame rate of the simulation. Since user behavior is an exterior variable that can not be perfectly replicated, these tests consist only of the AI traffic.

The tests were performed on a computer with the following specs:

CPU: AMD Ryzen 5 2600 Six-Core

RAM: 16 GB

GPU: Radeon RX 580 Series

Disk: Microsoft Storage Space Device HDD

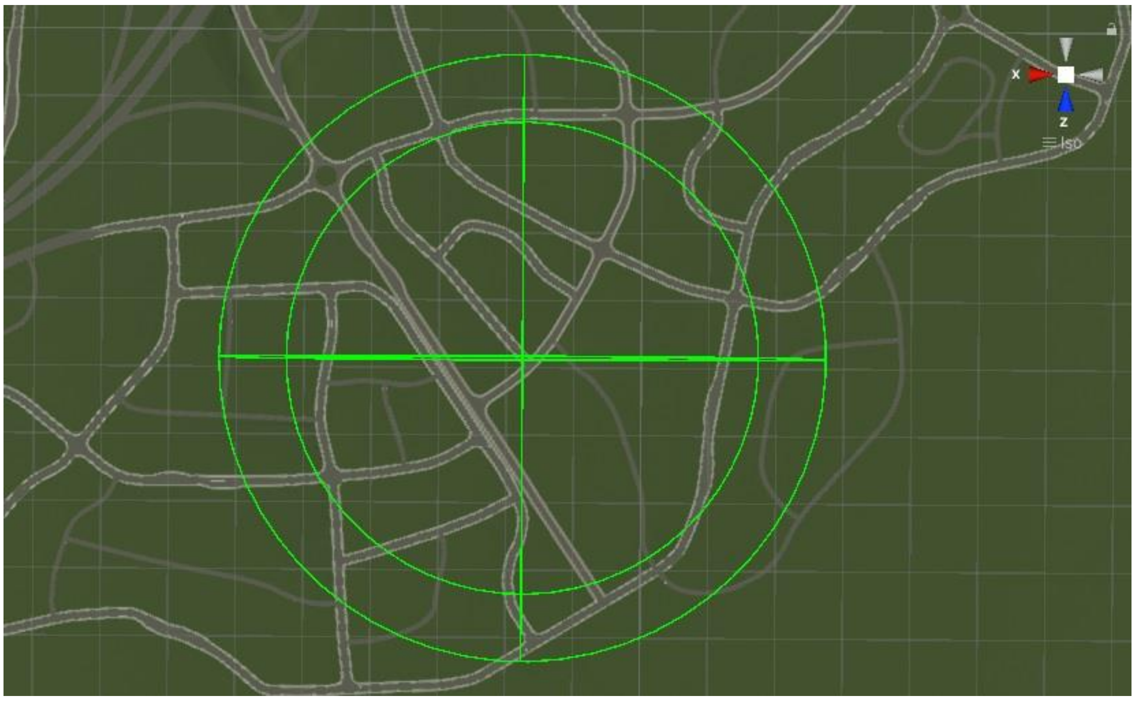

Figure 12. Test area used for frame rate test visualized with gizmos in editor view

Table 1. Frame rate depending on the chosen method and amount of AI cars

Cars in simulation	20	25	30	35	40	45	50
Original	62	58	55	52	50	45	41
Method A	61	58	53	49	47	43	39
Method B	59	54	50	47	43	39	34
Method C	62	57	54	53	50	44	40

The first test was used to compare the performance costs of each method. The test area is shown in Figure 12. The area between the two circles is reserved for spawning and despawning cars and the area limited by the inner circle is the main operating zone for the AI cars. Table 1 shows how the overall frame rate changes with each method when more cars are added to the environment. The margin of error in this test is one to two frames.

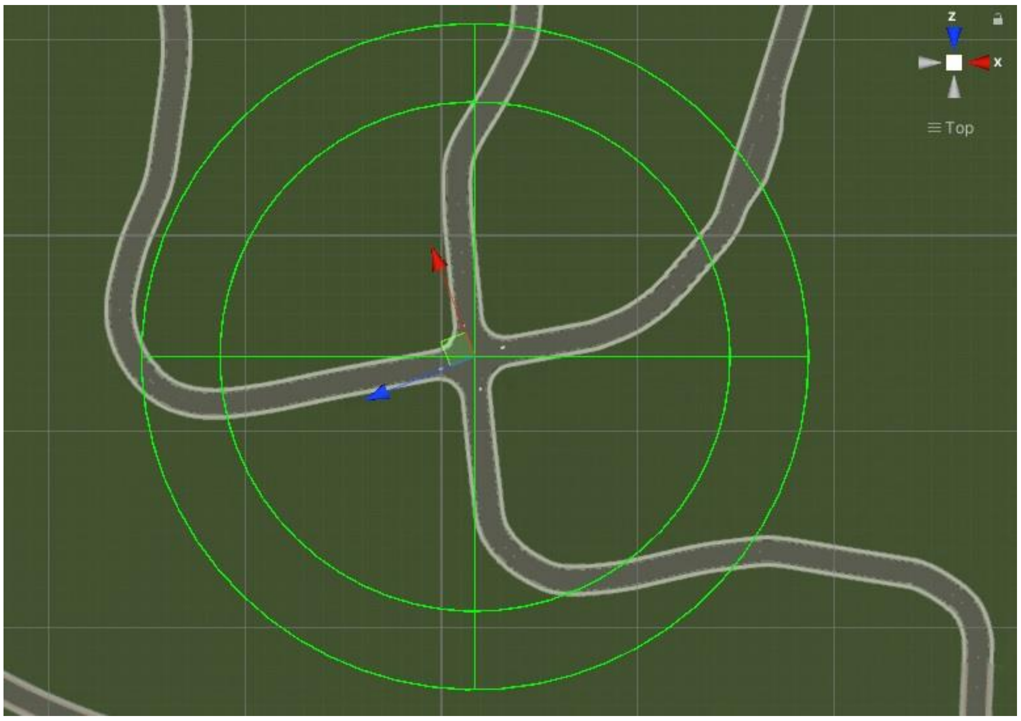

Figure 13. Test area used for traffic flow test visualized with gizmos in editor view

Table 2. Traffic flow depending on the chosen method

Cars in simulation	5	7	10	15	20
Original	45	56	74	118	151
Method A	43	50	62	73	89
Method B	44	51	68	102	119
Method C	44	53	76	117	153

Another important aspect that would be affected by the introductions of equal intersections would be the general traffic flow. For this test, we restricted the driving area to a single intersection, for the most consistent results (area shown in Figure 13). Table 2 shows the average amount of cars that pass the intersection in five minutes. The margin of error in this test is two to three cars.

5 FINDINGS

Now that we have covered each method’s strengths and weaknesses, we can compare these findings and decide which method would be the best choice for different types of driving simulations. Naturally, the benefit offered by each solution will depend on the scenario in question and the current number of different driving scenarios being quite low, some methods will appear much more appealing than the others. Therefore we will also discuss some possible future scenarios that could benefit from each method.

5.1 Comparison of methods

The core idea of method A was to use as many already existing functions of iTS as possible. The only, albeit not a small, addition this method had was the introduction of communication between AI cars. Performance-wise this method is not too demanding compared to the current version of iTS, but the big weakness this method had was the effect on traffic flow when enough cars were forced into an intersection. The main reason for this being the modification in question made the cars extremely considerate towards other cars in the junction. More technically speaking, they developed the issue of looping commands. Since this system has no outside observer or manager to control the situation it was very difficult to create exception handling for these types of situations. This method would be the optimal solution in scenarios that have vast environments and low traffic density, like countryside environments.

Method B was mostly an attempt to fix the general issues that method A had, specifically the issue of looping commands. By adding a singular object that is in charge of driving priorities this was an easily achievable goal. The traffic flow test shows that method B slows is slower than the original solution, which was predictable considering the fact that in original iTS intersections cars would not stop unless there were traffic lights or they were about to crash into each other. Still, taking into account that most of the existing scenarios would not exceed the amount of ten cars in such a small area, the traffic flow can be considered as a non-issue. Unlike method A, method B suffers more from overall additions of cars to the environment. Current versions of iTS simulations use an environment with twenty cars. Even though the performance with this number of cars is not a critical issue, it must be noted that the driving environment hasn't been completed and would add more stress to the overall simulation. During the development of the simulations, twenty cars on a rather small area are sufficient for testing but if this number would be raised in the future, the simulation might become too taxing for the current computers used to run the simulations. By this we mean the frame rate of the simulation dropping below the recommended value to not cause nausea. Method B would for these reasons be best suited for environments with densely packed traffic that operates at slow speed and the total amount of cars could be kept relatively low, like a city center.

Method C was created to fill the purpose of grading equal intersection behavior while avoiding traffic flow and performance issues at a cost of simplified functions and cut features. As mentioned in paragraph 4.4 method C does not operate unless the user car has arrived at the intersection in question. Therefore both of the tests show very similar results on method C and the original iTS system, only having differences that are the cause of the AI acting slightly differently in each test. Even when the player car would pass an intersection there would be no noticeable difference in performance and traffic flow would be only affected by the user's behavior. This method does not have the issue that the previous methods displayed but is by no means the optimal solution in every situation, but it is likely the most appealing for the current version of iTS driving simulations. The biggest missing feature in this method compared to the two previous ones is the lack of interaction between the AI cars. Even if these cars give way to the user when needed, they operate by the rule of first come first served which might cause some confusion for a user waiting for their turn to pass the intersection.

5.2 Recommendations for further research

5.2.1 Other type of intersections

Because this thesis was aiming to find possible solutions as a base for equal intersection these methods have not seen a throughout testing in other types of junctions. Namely the ones that do not have traffic lights but have priority systems tied to them, the most common being roundabouts. Method A does not support roundabouts and would need to be excluded in these situations or require additional modifications. But considering this method would work best in environments with very low-density traffic, it is unlikely that roundabouts would be set to those environments. Method B can be used to work with roundabouts but because these junctions consist of multiple individual intersections the performance issue becomes even more noticeable. For this reason, it might be best to further develop a custom manager for roundabouts if they appear in an environment that uses method B as a solution for equal intersections. Although iTS can be used to determine priorities in static situations like main road and roundabout, it might be beneficial for future development to chose a singular priority logic to be used in a simulation environment.

5.2.2 Car behavior profiles

In its current form, the AI cars of iTS drive very mechanically always obeying good traffic behavior etiquette. Not only does this lower the overall immersion of the simulation but it also leads to very predictable interactions with the AI cars. In method A each car gives priority individually, this leads to a test where some cars would be set to never give priority even if they should. Method As basic features worked well even when there were these so-called bad eggs mixed into the traffic. By expanding this type of individual behavior for cars they could be set to drive slower or faster than average, not give priority, and possibly even sometimes ignore traffic lights. Well implemented behavior profiles would lead to a much unpredictable user experience and could improve the immersion of these types of simulations. Some AI cars not following traffic rules would likely create problematic situations and require heavy testing to acquire a staple functionality.

5.2.3 Rivaling AI systems

Unity is one of the most used game engines and having a large number of third-party solutions, like the iTS, to suit the developers’ needs it is also worth further discuss if iTS can provide everything that is demanded from the simulations now and in the future.

“Road and Traffic System” is Unity third-party plugin that can be used to add AI to your scenes to create more immersive environments. The main difference for this system compared to iTS is the collision-based system. A collision based system means that AI cars act more reactively rather than having planned routes. This keeps performance costs down but can lead to traffic jams more easily. “Road and Traffic System” is considerably harder to set up and work with than iTS but it comes with source code so it is as modifiable as iTS.

“Simple Traffic System” is another more simplified version of traffic system compared to iTS. Unlike “Road and Traffic System” this plugin is rather easy to set up and use but quite time-consuming when creating larger road networks. “Simple Traffic System” works like a slate where one can add their traffic system needs as there are no existing additional features to create mixed signals with your custom methods. This can be seen as positive or negative depending on the priorities of the project’s development.

Most other traffic AI solutions that could be implemented to Unity, asset store plugins, and open source solutions included, are more bare-bones versions of these two solutions and therefore seem even less likely as a possible replacement for iTS.

6 CONCLUSION

For driving simulations, the AI traffic can be considered the most essential part of the user experience. It would be ideal to strive for a system where AI cars act very humanly and unpredictably enough to keep the user engaged and observant, but these properties need to be balanced with program performance and AI stability. Even if an environment could have AI cars that behaved very realistically, it would be counterproductive to use resources to produce such an AI if the simulation performance costs rise too high or the AI would develop bugs that would create unwanted actions and situations.

The main goal of this thesis was to provide different solutions to the implementation of equal intersections and to make this achievable all the solution concentrate on solving the issue from a different angle and by having different priorities on what features of the simulation could be sacrificed to create a working system for these types of junctions.

The hardest part of this development process was the existing environment where the driving simulations would take place. This environment was created to present large scale of different road and traffic types in a single entity. As discussed previously, this quickly leads to the unavoidable issue that the system in use would need to cater to multiple types of traffic densities and road types.

Since the TTS Työtehoseura driving simulations are mainly used for grading the users' behavior rather than creating as immersive a simulation as possible, method C seems to most beneficial addition. Even if this method lacks some features that the other two possess and does not have as much further development possibilities, the overall stability, and low-performance cost make it the best choice in its current form.

7 REFERENCES

[1] Artificial Intelligence (2018), retreivied May 10, 2020, from

https://plato.stanford.edu/entries/artificial-intelligence

[2] Linda Luccy, Definition: Artificial Intelligence, retrieved May 10, 2020, from

https://searchenterpriseai.techtarget.com/definition/AI-Artificial-Intelligence

[3] Artificial Intelligence (2020), retrieved May 10, 2020, from

https://en.wikipedia.org/wiki/Artificial_intelligence

[4] Artificial Intelligence in Video Games (2020), retrieved May 12, 2020, from

https://en.wikipedia.org/wiki/Artificial_intelligence_in_video_games

[5] Laura E Shummon Maass, Artificial Intelligence in Video Games (2019), retrieved May 12, 2020, from

https://towardsdatascience.com/artificial-intelligence-in-video-games-3e2566d59c22

[6] Risteys edessä – kuka väistää? (2017), retrieved May 18, 2020, from

https://www.liikenneturva.fi/fi/ajankohtaista/liikennevinkki/risteys-edessa-kuka-vaistaa#61034b3b

[7] Turvallinen risteysajaminen koostuu liikennesäännöistä ja yhteispelistä (2017), retrieved May 19, 2020, from

https://www.liikenneturva.fi/fi/ajankohtaista/liikennevinkki/turvallinen-risteysajaminen-koostuu-liikennesaannoista-ja-yhteispelista#1d21f282

[8] Jose Garrido, iTS – Intelligent Traffic System: Developer Manual (2014)