



Federal University of Pernambuco
Informatics Centre
Master in Computer Science



ITRANS – 3D TRANSIT SIMULATOR

by

Paulo Gonçalves de Barros

Recife, March 2005.



Federal University of Pernambuco
Informatics Centre
Master in Computer Science



ITRANS – 3D TRANSIT SIMULATOR

by

Paulo Gonçalves de Barros

Dissertation submitted to the Informatics Centre of the Federal University of Pernambuco as a partial fulfilment of the requirements for the degree of Master in Computer Science.

Advisor: Judith Kelner

Examining committee:

Dr. Cláudio Kirner;
Dr. Veronica Teichrieb;
Dr. Fernando da Fonseca;
Dr. Judith Kelner.

Chair of the Committee:

Dr. Fernando da Fonseca.

Recife, March 2005.

Barros, Paulo Gonçalves de
ITRANS – Simulador de trânsito 3D / Paulo
Gonçalves de Barros. – Recife : O Autor, 2005.
x, 83 folhas : il., fig., tab., quadros.
Dissertação (mestrado) – Universidade Federal
de Pernambuco. Cin. Ciência da Computação, 2005.
Inclui bibliografia e apêndice.
1. Ciência da computação – Realidade virtual. 2.
Tráfego urbano – Simulação de semáforos e veículos
– Modelos matemáticos de microsimulação. 3.
Interface tridimensional – Biblioteca gráfica OpenGL.
4. Vias de tráfego – Configuração de Dados – C++
(Linguagem de programação). I. Título.
004.94 CDU (2.ed.) UFPE
006.8 CDD (22.ed.) BC2005-106

ABSTRACT

Urban traffic planning has a fundamental role in our society, for it improves the use of traffic roads and optimizes the flow of both vehicles and pedestrians. Traffic planning reduces traffic jams, trip times and pollution, helping not only the economy of a city or state, but also citizens by assuring them a better quality of life.

In order to solve the problem of urban traffic, different mathematical models have been created. With the advent of computers, these models passed to be virtually built and simulated. Such models were denominated macro-simulation models, since they represented transit by flows in each street instead of representing the state of each vehicle separately.

With the increase of the processing capacity of computers, new computational models were developed and the micro-simulations appeared. These built a transit simulation model by simulating the behaviour of each transit entity, be it either a vehicle or a pedestrian.

Between these two above-mentioned groups, it might be said that micro-simulations are receiving more attention nowadays. It is in this group the present work is inserted.

The primary goal of this work is to develop a Virtual Reality (VR) interface for transit simulators. To accomplish this objective, a basic simulator was developed with 3D Virtual Reality desktop interface.

The simulation was structured so that it could be later used as a basis for the development of a transit simulator with a distributed architecture, similar to the ones currently being proposed in the area of Distributed Systems.

This work attempts to define an interface for transit simulation in three-dimensional virtual environments capable of exploring the potential obtained by the addition of a third dimension. It demanded not only the profound study of simulation models, but also the analysis of VR interfaces.

RESUMO

O planejamento de tráfego urbano tem um papel fundamental na sociedade atual, pois permite um melhor aproveitamento das vias de tráfego e otimiza o fluxo de veículos e pedestres. Este planejamento também reduz engarrafamentos, períodos de viagem e poluição, ajudando não só a economia, mas a população, através da garantia de uma melhor qualidade de vida.

A fim de solucionar os problemas do tráfego, diversos modelos matemáticos foram criados. O advento do computador fez com que esses modelos passassem a ser não somente construídos, mas também simulados. Tais modelos são chamados de modelos de macro-simulação, pois apresentam o trânsito através dos fluxos em cada rua, ao invés da representação de cada veículo.

Com o aumento da capacidade de processamento dos computadores, novos modelos de simulação computacional foram desenvolvidos e surgiram as micro-simulações, que constroem um modelo para o trânsito baseado na simulação de comportamento de cada entidade de trânsito: os veículos e pedestres.

Dentre esses dois grandes grupos, pode-se dizer que a micro-simulação tem atualmente recebido uma ênfase crescente, e é neste contexto que está inserido o presente trabalho.

O objetivo primordial deste trabalho é o desenvolvimento de uma interface de realidade virtual (RV) para simuladores de trânsito. Para atingir este objetivo, um simulador básico foi desenvolvido com uma interface de realidade virtual tridimensional desktop.

A simulação também possui uma estrutura que servirá de base para o desenvolvimento de um simulador de trânsito com arquitetura distribuída semelhante às desenvolvidas na área de Sistemas Distribuídos.

Procura-se também definir uma interface para simulação de trânsito em ambientes virtuais 3D que busca explorar o potencial obtido de uma terceira dimensão. Essa tarefa demandou não só o estudo aprofundado de modelos de simulação, mas também a análise de interfaces de realidade virtual.

CONTENTS

CHAPTER 1 : INTRODUCTION	1
1.1. HISTORICAL SUMMARY	1
1.2. OBJECTIVE.....	2
1.3. COMMON FEATURES IN TRANSIT SIMULATORS.....	2
1.4. TECHNOLOGIES USED IN TRANSIT SIMULATORS.....	6
1.5. PROJECT CONTEXTUALIZATION.....	8
1.6. DOCUMENT STRUCTURE	9
CHAPTER 2 : DATA CONFIGURATION.....	10
2.1. TRAFFIC WAYS	10
2.1.1. Manual Construction of the Traffic Ways	10
2.1.2. Automatic Construction of the Traffic Ways	11
2.1.3. Traffic Structures Mapping	12
2.2. TERRAIN.....	16
2.2.1. Terrain Generation for a Traffic Graph Manually Defined.....	16
2.2.2. Terrain Generation for a Traffic Graph Automatically Defined	17
2.2.3. Buildings and Urbanization.....	18
2.3. VEHICLES	18
CHAPTER 3 : SIMULATOR.....	20
3.1. SIMULATOR ARCHITECTURE.....	20
3.2. VEHICLES' PATHS	27
3.2.1. Initial Configuration for Vehicles with a New Path.....	29
3.2.2. Updating the Vehicle to its Next Destination Point	30
3.2.3. Vehicle Final Destination Arrival Verification and Finalization	31
3.2.4. Possible Vehicle States.....	31
3.3. TRAFFIC LIGHTS SIMULATION	32
3.3.1. Semaphore Configuration for Manually Generated Graph.....	33
3.3.2. Semaphore Configuration for Automatically Generated Graph	34
3.3.3. Graphical representation	35
3.4. VEHICLE BEHAVIOUR	36
3.4.1. Vehicle Position Update Based in its Current Target Point	36
3.4.2. Red Light Vehicle Parking.....	37
3.4.3. Vehicle Collision Calculation	38
3.4.4. Calculation of the Separation Force between Vehicles.....	39
3.5. OPTIMIZATIONS.....	41
3.5.1. Relative Time Unit Alteration.....	41

3.5.2.	Graphical Scene.....	42
3.5.3.	LOD.....	42
3.5.4.	Data Structure Optimization	43
3.5.5.	Algorithmic Improvements	44
CHAPTER 4 : INTERFACE.....		45
4.1.	AXLES SYSTEM	45
4.2.	INTERFACE COMPONENTS.....	45
4.3.	VIEWPOINTS.....	49
4.4.	SIMULATION CONTROL COMMANDS	49
CHAPTER 5 : PERFORMANCE ANALYSIS.....		51
5.1.	EXPERIMENTS	51
5.1.1.	Vehicle LOD Performance Experiment.....	52
5.1.2.	Performance Experiment according to the Number of Junction.....	54
5.2.	RESULTS AND ANALYSIS	56
5.2.1.	Vehicle LOD Performance Experiment.....	56
5.2.2.	Performance Experiment according to the Number of Junctions	60
CHAPTER 6 : CONCLUSION		65
6.1.	DIFFICULTIES	66
6.2.	CONTRIBUTIONS.....	67
6.3.	FUTURE WORK.....	67
6.3.1.	Vehicle Behaviour Improvement.....	68
6.3.2.	Simulation Distribution Modification.....	68
6.3.3.	Topographic Information Improvement	69
6.3.4.	Auto-adjustable LOD System	69
6.3.5.	Final Considerations.....	70
BIBLIOGRAPHY		71
	REFERENCES	71
	RECOMMENDED BIBLIOGRAPHY.....	75
APPENDIX		76
A.1.	CLASS DIAGRAMS	76
A.1.1.	Class Relations Diagram.....	76
A.1.2.	Detailed Class Diagram.....	76
A.2.	USER COMMAND KEYS.....	78
A.2.1.	Viewpoint Control Commands	78
A.2.2.	Navigation Commands.....	78
A.3.	ALGORITHMS	79
A.3.1.	Vehicle Path Following Algorithm.....	80
A.3.2.	Vehicle Collision Detection Algorithm	80

A.3.3.	Vehicle Separation Algorithm	81
A.3.4.	Vehicle Path Generation Algorithm.....	82

FIGURES

<i>Número</i>	<i>Página</i>
<i>Figure 1: Automatically generated traffic graph example.....</i>	<i>12</i>
<i>Figure 2: Types of possible connections in a traffic mesh</i>	<i>14</i>
<i>Figure 3: Calculations and results of the shifts applied between lanes' end and start points.....</i>	<i>15</i>
<i>Figure 4: Traffic network representation using a bidirectional graph.....</i>	<i>20</i>
<i>Figure 5: Edge mapping in different transit situations in the traffic mesh.</i>	<i>21</i>
<i>Figure 6: Simulator architecture.....</i>	<i>23</i>
<i>Figure 7: Sequence method diagram of the initialize method.....</i>	<i>24</i>
<i>Figure 8: Sequence diagram of the update method.</i>	<i>25</i>
<i>Figure 9: Sequence diagram of the draw method.....</i>	<i>26</i>
<i>Figure 10: Sequence diagram of the deinitialize method.....</i>	<i>26</i>
<i>Figure 11: Application general procedural structure.</i>	<i>27</i>
<i>Figure 12: Probability interval calculation for outgoing connections located in a junction during vehicle path generation.....</i>	<i>28</i>
<i>Figure 13: Connection selection process during vehicle path creation.....</i>	<i>29</i>
<i>Figure 14: Vehicle following points and state update.....</i>	<i>31</i>
<i>Figure 15: Textures applied to the traffic light according to its state.....</i>	<i>35</i>
<i>Figure 16: Vehicle position adjustment using an adapted path following algorithm (Reynolds, 1999).</i>	<i>37</i>
<i>Figure 17: Vehicle collision region definition.</i>	<i>39</i>
<i>Figure 18: Vehicle position adjustment using a modified separation algorithm (Reynolds, 1999) for vehicles in the same lane.</i>	<i>40</i>
<i>Figure 19: Vehicle position adjustment using the modified separation algorithm (Reynolds, 1999) for vehicle in adjacent lanes.....</i>	<i>41</i>
<i>Figure 20: Position of the three coordinate axes with respect to the user and the monitor.....</i>	<i>45</i>
<i>Figure 21: Interface providing simulation information to the user.</i>	<i>46</i>
<i>Figure 22: Traffic ways visualization for a manually generated traffic graph.</i>	<i>47</i>
<i>Figure 23: Visualization of semaphore functioning and higher LODs of vehicle avatars for a manually generated traffic graph.</i>	<i>47</i>
<i>Figure 24: Visualization of terrain and vehicle LODs for an automatically generated traffic graph.....</i>	<i>48</i>
<i>Figure 25: Visualization of terrain and vehicle LODs for an automatically generated traffic graph.....</i>	<i>48</i>
<i>Figure 26: Lista de pontos de vista do usuário.....</i>	<i>49</i>
<i>Figure 27: Possible user movements' illustration.....</i>	<i>50</i>
<i>Figure 28: User positioning for experiment 1.....</i>	<i>52</i>
<i>Figure 29: User positioning and graph shape presentation for test 7 of experiment 1.....</i>	<i>53</i>

Figure 30: Graphical results of the frame rates for each test in experiment 1.....	56
Figure 31: Frame rates for experiment 1 with LOD 0 applied to vehicle avatars.	57
Figure 32: Frame rates for experiment 1 with LOD 1 applied to vehicle avatars.	57
Figure 33: Frame rates for experiment 1 with LOD 2 applied to vehicle avatars.	57
Figure 34: Frame rates for experiment 1 with LOD 3 applied to vehicle avatars.	57
Figure 35: Frame rates for experiment 1 with LOD 4 applied to vehicle avatars.	58
Figure 36: Frame rates for experiment 1 with LOD 5 applied to vehicle avatars.	58
Figure 37: Frame rates for experiment 1 with all LODs applied to vehicles and the user located in the centre of the graph.	58
Figure 38: Frame rates for experiment 1 with all LODs applied to vehicles and the user located in the further extremity of the graph.	58
Figure 39: Frame rates for experiment 1 with all LODs applied to vehicles and the user located in one of the corners bounding rectangle of the graph.....	59
Figure 40: Average frame rates based on the samples of the three measurements of the seventh test of experiment 1.....	59
Figure 41: Histogram with data obtained from experiment 1 and vehicles represented only by LOD 3..	59
Figure 42: Graphics presenting frame rate and memory consumption results for experiment 2 tests.....	60
Figure 43: Frames rates of experiment 2 using graph 0.	61
Figure 44: Frames rates of experiment 2 using graph 1.	61
Figure 45: Frames rates of experiment 2 using graph 2.	61
Figure 46: Frames rates of experiment 2 using graph 3.	61
Figure 47: Frames rates of experiment 2 using graph 4.	62
Figure 48: Frames rates of experiment 2 using graph 5.	62
Figure 49: Frames rates of experiment 2 using graph 6.	62
Figure 50: Histogram with test data of experiment 2 with graph 4.....	64
Figure 51: Main system classes and their relationship.	76
Figure 52: Description of classes' methods and attributes.	77

TABLES

<i>Número</i>	<i>Página</i>
<i>Table 1: File line format for traffic graph point specification.....</i>	<i>11</i>
<i>Table 2: File line format for traffic graph edge specification.....</i>	<i>11</i>
<i>Table 3: File line format specification of the automatic traffic graph generation.....</i>	<i>12</i>
<i>Table 4: File line format for terrain area specification.</i>	<i>16</i>
<i>Table 5: File line format for avatar LOD specification of a terrain area.....</i>	<i>17</i>
<i>Table 6: Levels of detail with their activation distances and texture colours.</i>	<i>18</i>
<i>Table 7: File line format for specification of the maximum number of vehicles simulated.....</i>	<i>18</i>
<i>Table 8: Velocity and mass value variation for vehicles in the simulation.</i>	<i>30</i>
<i>Table 9: Standard traffic light time values.</i>	<i>34</i>
<i>Table 10: Different vehicle LODs information.....</i>	<i>43</i>
<i>Table 11: Automatically generated graph configuration for the vehicle LOD performance test experiment.....</i>	<i>52</i>
<i>Table 12: Automatically generated traffic graph configurations for performance test according to the number of junctions.</i>	<i>55</i>
<i>Table 13: Frame rates results for each test in experiment 1.....</i>	<i>56</i>
<i>Table 14: Frame rates and levels of memory consumption for each tested graph in experiment 2.....</i>	<i>60</i>
<i>Table 15: Number of entrance connection for each graph used in experiment 2.....</i>	<i>62</i>
<i>Table 16: User viewpoint control commands.</i>	<i>78</i>
<i>Table 17: User navigation commands.....</i>	<i>79</i>

ACKNOWLEDGEMENTS

I would like to thank my family, of which I am very proud to be part of. I thank them from the bottom of my heart for all their support during my Master course.

I also would like to thank my friend and advisor, Dr. Judith Kelner who has been giving me professional support and persistently teaching me how to become a real researcher since 1999.

I thank my friends for their help, patience and comprehension for all the times I couldn't be present in neither the RPG or cinema sessions nor parties.

Specially, I would like to thank my beloved Isabella, the one who nourishes my soul with tenderness and love more than anyone and to whom I dedicate this simple, but laborious work.

I would like to thank Dr. Alejandro Frery, Dr. Geber Ramalho, Dr. Hermano Perrelli and Dr. Alex Sandro Gomes, who provided me great support, inspiration and wisdom during my technologic-scientific journey in the last two years. I also would like to thank the Federal University of Pernambuco, its Informatics Centre and CAPES for turning into reality this opportunity of academic improvement.

Lastly, I would like thank *.* , everything and everyone whom, even with unconscious actions and, perhaps, imperceptible as the clapping wings of a butterfly, bring breezes and gales to the extraordinary reality in which I live. Without it, I would never be who I am.

*“May the time given by life take away sorrow and bring me joy
for it is clear this time is the light in the boundaries of an empty dawn”.*

(P.G.B.)

GLOSSÁRIO

A

Avatar: entity responsible for representing the user in the virtual world. In this work, it will be defined as the representation of any object or entity, be it the user or not.

C

CAVE: a room that allows the projection of virtual environments in its walls, as well as their manipulation with advanced Virtual Reality interfaces.

CITU: **U**rban **T**ransit and **T**ransport **C**ompany (**C**ompanhia de **T**rânsito e **T**ransportes **U**rbanos).

D

Dead-reckoning: recognition of a common state for all players during a game involving more than one player. This term is used only for network games, when synchronism between PCs is required.

DEVS: **D**iscrete **E**Vents Description **S**ystem.

DIS: **D**istributed **I**nteractive **S**imulation.

H

HLA: acronym for **H**igh **L**evel **A**rchitecture. It is a general-purpose architecture for simulation with reuse and interoperability.

I

ISA: acronym for **I**ntelligent **S**peed **A**daptation System: autonomous systems that automatically adjust their speed according to the road speed limit.

ITS: **I**nstitute for **T**ransport **S**tudies. Institute dedicated to the study of transportation located in the city of Leeds, United Kingdom.

K

KQML: acronym for **K**nowledge **Q**uery and **M**anipulation **L**anguage. Language used for knowledge and information exchange between computational entities such as agents.

L

LOD: acronym for **L**evel **O**f **D**etail. It indicates the level of detail among the many graphical representations an object may have in the virtual world.

O

Stereoscopic glasses: glasses that provide virtual environment views slightly deflected for each eye, giving to the human brain a three-dimensional impression of the virtual world being presented.

P

Pathfinding: path search done by the animated entities in a game or virtual world.

R

Replay: capacity of going back to specific moments in the simulation and reviewing them, such as in a video.

S

SACI: **S**imple **A**gent **C**ommunication **I**nfrastructure.

SLX: **S**imulation **L**anguage with **E**Xtensibility.

U

UTC systems: **U**rban **T**raffic **C**ontrol systems. They consist of electronic systems which measure and control urban traffic flow in specific roads.

W

World-Up: simulation tool that allows the use of advanced and immersive Virtual Reality interfaces in applications.

Chapter 1 :

Introduction

Intense and disordered transit is one of the main problems of large Brazilian cities. Not only does it promote traffic jams, but it also pollutes atmosphere and causes accidents. According to **CTTU** (Urban Transit and Traffic Company) 11,722 accidents were recorded in Recife, causing 57 deaths and leaving 2,288 people injured in 2003. The solution for the transit problem is still considered to be a great challenge, which tends to increase along with the city vehicle fleet.

Taking as an example the city of Recife, during the year of 2004, around a thousand vehicles were added to the total fleet in each month. Should the metropolitan region be considered, this number becomes even larger, as expected. According to CTTU, the city had a 44% increase in its fleet in the last decade. Besides, due to the polarity and centrality of services, Recife daily receives part of the vehicular volume from the metropolitan region. The union of all these factors are causing an enormous pressure in the vehicular system, pressure whose tendency is only to increase.

Urban traffic planning allows the better use of traffic ways and optimizes vehicle and pedestrian flow, promoting the reduction of traffic jams, travel times and pollution. Therefore, traffic planning not only helps not only the economy, by providing more efficient transportation of goods and workmanship, but it also guarantees to citizens a better quality of life.

This chapter will give an approach to how traffic planning evolved along History. It will also present the technology used and the features contained in transit simulations nowadays, insert the developed research in this context and, lastly, explain how the content of the work is organized in the chapters to follow.

1.1. Historical Summary

A diversity of Mathematical models was created envisaging the minimization of traffic problems. Such models basically represent vehicle flow by arrows and points in a graph. The arrows or edges represent roads. The points or nodes represent junctions, also called crossings. Extra features are added to each edge, such as flow, number of lanes, lane width, speed limit and

length. The same happens to nodes. Thus, it is possible to build with pen and paper a model of any road structure.

The advent of computer caused these models to be not simply built, but also simulated in computers. Such models were called macro-simulation models, since they represented transit by the traffic flow in each street. With computational simulation, it was possible to determine the best way to build a new road in an urban area (Clark, 1997). The road plan, its number of lanes and traffic lights positioning could be re-evaluated as many times as necessary and until the expectation which motivated its design were reached without the necessary monetary expenditure for its construction (Macredie, 1996).

With the increase of the processing capacity of computers, new computational simulation models were developed, leading to the appearance of the micro-simulations, which built a model for transit based on the behaviour simulation of each transit entity as, for example, vehicles and pedestrians. The vehicles follow distinct paths inside of the traffic network according to the average traffic flow in each street (Owen, 2000).

Simulation aids the work of traffic engineers, by allowing the visualization and analysis of critical or adverse situations in important traffic ways, and is projected without the necessity to be physically implemented.

1.2. Objective

The primary objective of this work was to develop a Virtual Reality (VR) interface for transit simulators. In order to accomplish this, a basic simulator were developed with a VR interface. This application, composed by this interface, the simulator and a flexible system of traffic mesh configuration, was called ITranS.

As a secondary objective, an application structure was developed in order to serve as a basis for the future development of transit simulators with distributed or parallel architectures, similar to the ones developed in (Cameron, 1994) and (Klein, 1998).

1.3. Common Features in Transit Simulators

Innumerable are the research groups and companies developing software in the area of traffic simulation. A research carried by the ITS (Institute of Transport Studies), in the University

of Leeds, United Kingdom (SMARTTEST, 2000), lists 57 tools which deal with traffic situations in the most varied manners.

According to ITS, these tools can be classified, basically, in 4 groups:

- Urban models: they simulate situations in internal areas of cities, where traffic flow is slower and suffers more interruptions;
- Free-way models: they simulate roads with higher speed located in the periphery of urban areas, where traffic lights are sparser and where vehicle speeds are higher;
- Combined models: they mix features of the two above mentioned models, being used for both urban simulations and road simulations;
- Highway models: they are used to test vehicles and autonomous transport systems, where flow is simulated without the interruption of traffic lights.

These models generally have features in common. The first of them is the fact that almost all of them have their update time based on discrete units, in one-second intervals.

Another feature is the definition of the vehicles' routes, where the origin and destination points are specified. In the traditional method, the intermediate points are chosen according to the percentage of contribution of the vehicle flow in each road to the total outgoing flow of a junction to which they are connected. For example, if 60% of the traffic flow in a point A goes to point B, the probability of a vehicle, being in point A, go to point B is 0.6. Since this method is not flexible enough to provide route changes, another method, where each vehicle defines its own route during path traversal, initially knowing only its destination point, is becoming each time more popular.

The operation of traffic lights is also a common feature to most models. Its implementation in the analyzed models vary as follows: they are either defined in a separate module, with a proper specific description language, internal to the simulation or based on data extracted in real-time from an urban region by Urban Traffic Control Systems (UTC systems).

Few are the models that consider pedestrians and cyclists as an influential part of traffic flow. Similarly, in most models the behaviour of public transports is not specifically defined (Liu, 2000).

Generally, models only worry about the continuous flow of vehicles, not taking into consideration the possibility of them to park, enter or leave stores or houses. This characteristic could modify the flow in diverse points of a vehicle's path.

The majority of the models present some simulation state information as output, such as trip times of vehicles, the variation of these trip times, vehicles' speed and traffic jams formation, localization and length.

On the one hand the simulation generally has configuration parameters, which give flexibility to the models being simulated. On the other hand, the integration with databases and other tools rarely occurs. In order to improve the simulation performance, event execution happens up to five times faster than real time. Furthermore, vehicles are generally put into motion with a higher speed than its normal average and traffic lights do not take as long to change their state, as they normally should.

Model calibration - necessary for a trustworthy reproduction of reality - exists, but the data used in this process varies from model to model. Among the analyzed data, there are statistics of average fuel consumption, pollutants emission rates, acceleration, deceleration and speed limits of the vehicles, as well as information of the flow of vehicles in each road link.

These models are validated comparing their output data to the real data obtained in the region by field research. Currently, little emphasis is given to this feature, which makes the increase of confidence in these tools difficult.

With respect to graphical interfaces there is still much to be improved, although some applications already use three-dimensional visualization models (Seneviratne, 2001). No simulation model using VR equipment was found, except for those used in pilot training. Models without an immersive interface also exist as well as models in which the user drives one of the vehicles inside the modelled traffic (Bayarri, 1996).

In most of the models, the speed limit of the vehicles is fixed during its entire course, which does not occur in reality where each road section in a single street may have different speed limits. Some projects had success in testing speed limits simulating vehicles with the ISA system (Intelligent Speed Adaptation System) (Liu, 2000).

Lane changing is already well mapped in some models and occurs when there is need for one vehicle to pass another, change route, stop, etc. In other models, vehicles have the capacity to learn, accumulate experience and improve their behaviours from simulation to simulation.

The definition of the vehicles behaviours is based on psychophysical models (Schulze, 1997). They try to shape the movement of automobiles and the sociological behaviour of drivers. Therefore, they allow the creation of drivers with different levels of aggressiveness or passivity in transit. Consequently, vehicles are presented with different speeds, accelerations and dispositions for passing (Al-Shihabi, 2001).

With regard to VR, models have evolved to become each time more realistic, either by the use of game engines for processing the environment (DeLeon, 2000), or by the definition of improved level of detail techniques (Of Floriani, 2002). However, the full potential of the 3D interface is still not completely explored, owing to the insistence of researchers in using the same implementation paradigms for bidimensional interfaces in three-dimensional interfaces (Stappers, 2000). To define an interface for transit simulation in three-dimensional virtual environments is a task that demands not only the thorough study of simulation models, but also the analysis of VR interfaces.

The game industry has enormously contributed to the development of applications to simulate transit in realistic way. Problems such as obtaining replay of the simulation (Wagner, 2004), dealing with delay in the communication between players (Pantel, 2002), allowing dead-reckoning and executing pathfinding quickly (Smith, 2002) have been solved in a satisfactory way by the digital entertainment industry. Although these solutions have limited complexity to the benefit of the performance, they may be partially inherited and adapted for transit simulations (Adzima, 2001).

Another similarity between these two types of application is the great number of interactive entities possible, generating problems in the use of monolithic processing (Roehl, 1995; Brutzman, 1995). To solve them, virtual environments are using technologies such as the IEEE DIS standard (Macedonia, 1995) and the HLA architecture (Fullford, 1996), which make possible the distribution of entities processing of the virtual world and the transmission of visual and behavioural information between different systems.

Auxiliary informative visual features in the interface are a basic concept to help the user orientate in the virtual world, but it must be parsimoniously applied so that its use brings benefits

to the user instead of confusion (Morar, 2002). The sense of immersion must not be merely restricted to the sense of vision. It must involve the five senses. The influence of each one of them in the perception of the virtual world and the proper movement of the user has been studied in the university of York (Harris, 2002). The higher the number of impressions felt by the user senses via a virtual environment interface, the more immersive and convincingly this interface will be.

1.4. Technologies Used in Transit Simulators

The transit simulation tools currently developed use macro-simulation, micro-simulation models or both of them. Cellular-automaton-based models (Blue, 2003), be it associated or not with DEVS (Lo Tartar, 2001; Díaz, 2001), are also under development, but with smaller notoriety. Amongst the two larger types of simulation above mentioned, it may be said that the micro-simulations are currently receiving more emphasis. However, despite the availability of more powerful machines, it is still not possible to build micro-simulation representing very extensive regions or with a great amount of vehicles without implying in the reduction of the model update rates.

Architectures for applications executed in only one machine have been implemented. They are coded with optimized algorithms to improve performance, are aided by supercomputers in information processing (Manouselis, 2001; Jayakrishnan, 1990) and are implemented in languages with high extensibility, such as SLX (Schulze, 2001; Lemessi, 2001), to guarantee the fast extension and adaptation of the application.

Multi-agent systems (SMAs), using the Gaia formal language (Manouselis, 2001), for example, have also been considered. In these systems, each agent represents a vehicle or traffic light. Thangiah et al. (Thangiah, 2001) proposed a flexible agent-based architecture where each entity uses a different algorithm for definition of vehicle routes. Furthermore, tests with SMAs, using SACI and KQML languages (Schmitz, 2002) to control urban traffic where agents simulated traffic lights and vehicles have also achieved positive and interesting results (Schmitz, 2002; Schmitz, 2002-2). Moreover, a SMA developed by Paruchuri et al. (Paruchuri, 2002) using C++ and the Qt graphical library (Qt, 2005) simulated an example of unordered traffic caused by drivers with different states of mood, each one of them driving and reacting to transit in a different manner.

In order to deal with the problem of the computational load in the processor, distributed architectures, such as parallel computing systems, have been used (Hsin, 1992). An example of the

effectiveness of such approach is the PARAMICS (Cameron, 1994), a system capable of simulating hundreds of thousands of vehicles passing through thousand of roads. The IEEE HLA interoperation standard, developed by the Department of Defense of the United States, is yet another architecture developed for such applications. It allows the distributed control of entities, denominated federates, via a standardized interface. Projects using HLA presented positive results, as it is the case of Klein (Klein, 1998), where a simulation of vehicles, pedestrians and traffic lights was successfully developed.

Efficient algorithms are essential for achieving high performance simulation. One of the greatest problems faced nowadays is the definition of better routes for vehicles that deliver goods. They have to pass by a set of specific geographic locations. Similar to the problem of the travelling salesman, this problem is NP-hard, not allowing methods capable of reaching an accurate result. Algorithms of evolutive computing have been the best option in the solution for this type of problems (Tavares, 2003; I read, 2002; Sun, 2002). Despite the great commercial importance this problem has (Schulze, 2001), it runs out of the scope of the present work and will not be argued in depth.

Apart from efficiency, flexibility is another important point in a transit simulation, for the reconfiguration of roads and their flows has to be made for every new simulated region. Solutions for generation of roads based on bi-dimensional images have been used in the construction of urban models of cities (Marson, 2003) and of its streets (Sun, 2002), but a previous preparation of these images is still necessary in most cases.

Virtual environments interaction is a controversial area of study. Despite the innovative results presented by electronic games interfaces, each application requires a different interface, according to its functionalities (Reisman, 2003). Because of that, there is a great difficulty in defining rules for the development of three-dimensional interfaces. Moreover, a deep understanding of what is necessary for the people interact with them routinely must be obtained, so that the best form of interaction in a virtual world is found and provided to the user (Sheridan, 2000). The guarantee of immersion, by the use of advanced interfaces such as CAVEs or stereoscopic glasses, is still under discussion and analysis, although some methodologies have already been developed to prove it (Raja, 2004, Raja, 2004-2). Research in these areas must be further formalized before results can be taken as reference (Roberson, 1997). More important than the hardware interface is how the virtual world is presented to the user and how easy is for him to feel oriented in the three-dimensional environment.

1.5. Project Contextualization

The model considered in the present project is of the combined type, where situations of intense and slow traffic in urban regions can both be simulated as well as situations of sparse and more rapid traffic, as in highways and freeways. Its execution time is limited only by the frame rate with which the computer is capable of processing the simulation.

The application developed in this project is organized in three parts, each with a fundamental role:

- Data Configuration System, where the parameterization and initiation of the basic structures of the application is done;
- Simulator, where all system functioning is implemented, including vehicles, traffic lights and the user;
- Interface, responsible for drawing the graphical scene and presenting it to the user.

Interaction between these components happens in the following way. The road mesh is created from a set of points and edges that represent junctions and road sections or links. Each edge has specific values for vehicle flow per minute and traffic light times. The compilation of this information is done during data configuration. After that, the simulator is activated, performing the actions for each entity of the virtual environment. In each frame, both the simulation and the user interface are updated. The relation between the application parts will be thoroughly discussed in chapters 2, 3 and 4.

The vehicle routes are generated by the traditional method and at once, before the vehicle starts to traverse it, making them inflexible to traffic incidents such as traffic jams. The maintenance of this traditional model occurred in order to reduce the complexity of this initial version. A vehicle does not stop until it finds its point of destination. The maximum speed of each vehicle is fixed during all the simulation and the lane changes are random and happen only in road link transitions. Fuel consumption and pollutants emission measurements are not part of the model.

Initial parameters are configured in different input archives. Some numerical data are extracted from the simulation and presented on screen, including frame rates and position and orientation of the user's avatar. The prototype validation is given by the measurement of the

vehicles trip times. This version does not include simulation of pedestrians, cyclists and public transports.

Model calibration is based on the vehicles speed and acceleration. The interface is three-dimensional, using OpenGL (OpenGL, 2005), and levels of detail (LOD) are applied to the graphical models of vehicles and terrain. Moreover, the structure of the simulation was based on current distributed simulation models, which allows its distribution and expansion in future versions.

Finally, vehicles movement is given by the use of modified artificial life algorithms, which are generally applied to the movement of groups of characters in games, as part of the path following and separation mechanisms (Reynolds, 1999).

This project is the continuation of other works in scientific initiation levels and final graduation work developed by the author in the period from August 2001 to March 2003. During this period, initial tests with vehicles movement were conducted and initial versions of transit simulators were developed in an experimental model, which simulated the transit of the Salgadoinho Complex region, in Recife, Pernambuco, using the World-Up simulation tool (World-Up, 2000; Barros, 2003). Although the algorithmic ideas have been partially inherited, this case study was not considered in the current work.

1.6. Document Structure

The work is organized in four chapters. Chapter 2 explains how the initial configuration of the system occurs. Chapter 3 describes the development methodologies used during the creation of the simulator, from an architectural standpoint to the functioning of the vehicles and traffic lights. Chapter 4 approaches the user interface and its different manipulation features. Chapter 5 details experiments carried for testing certain aspects of the application performance. It also presents their results. Finally, Chapter 6 presents the conclusions achieved after analysis of the results and shows a perspective on what must still be done and what can be optimized in the simulation.

Chapter 2 :

Data Configuration

This chapter describes the first part of the simulator, which is the data configuration system. In order to simulate the traffic of a region, the application needs three types of input data: the traffic ways, the terrain and the vehicles data. The first type is used to configure the streets and junctions, whereas the second loads the objects that represent the terrain surface, and the third defines the number of vehicles in the simulation. These data are loaded during the construction of the Simulation class and are described below.

2.1. Traffic Ways

The traffic ways can be specified one by one by archive input or generated automatically in a homogeneously distributed grid. The first type is used in the construction of specific and realistic models. The second is used in the simulator performance tests. Below, the information processing for either case is described.

2.1.1. Manual Construction of the Traffic Ways

In the manual construction of the traffic ways, the information is extracted from a text archive, containing nodes and edges data of a directed graph, cyclical or not, which is manually generated by the user according to the traffic mesh of a region. The translation of edges and points data contained in orto-photo letters directly involves the use of complex image processing algorithms and, thus, there is no general applicable formula. The images need to pass by a pre-processing stage before their data can be used in the model. The development of such translation functionality is out of the scope of this first version of the work.

The archive containing data organizes, for each line of text, information about a single point or edge separated by spaces. The specification formats of the archive for points and edges are illustrated in Tables 1 and 2, respectively, with arbitrary example values. Likewise, arbitrary values will be defined in all subsequent format specification tables to exemplify the possible values in each field of a determined line of archive. The fields of Table 2 referring to the configuration of traffic lights will have their meaning clarified in section 3.3.

Table 1: File line format for traffic graph point specification.

<i>C</i>	<i>Id</i>	<i>X</i>	<i>Y</i>	<i>Z</i>
p	0	-50.2	0.2	20.3

The fields in Table 1 are the following: *C* - code informing that a point is being described in this line of archive, *Id* - identification number of the point, *X* - position of the point in the x coordinate axle, *Y* - position of the point in the y coordinate axle and *Z* - position of the point in the z coordinate axle.

Table 2: File line format for traffic graph edge specification.

<i>C</i>	<i>Id</i>	<i>F</i>	<i>Oid</i>	<i>Did</i>	<i>NL</i>	<i>WL</i>	T_g	T_y	T_r	H_{it}
E	0	12	0	1	3	2.5	10.3	13.5	23.3	2.5

The fields in Table 2 are the following: *C* - code informing that an edge is being described in this line of archive, *Id* - identification number of the edge, *F* - flow of vehicles per minute, *Oid* - identification number of the origin point, *Did* - identification number of the destination point, *NL* - number of lanes, *WL* - width of the lanes in meters, T_g - final time of the green light in seconds, T_y - final time of the yellow light in seconds, T_r - final time of the red light in seconds and H_{it} - initial halted wait time of the traffic light in seconds.

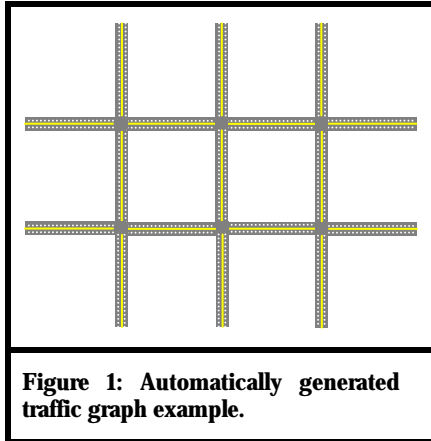
A data structure containing the exact archive content is created in the Simulation class using specific classes. This structure is public and has its content accessed through Simulation methods by any class. Such structure is necessary due to the high degree of dependence between connections themselves, as well as between junctions. It consists of a reduced model of the traffic region and must be present in each junction, so that they do not depend on data contained in other junctions and, thus, avoid losing autonomy, which is necessary to their future distribution.

2.1.2. Automatic Construction of the Traffic Ways

In order to run tests in diverse levels of traffic complexity, a simple but efficient automatic traffic ways generator was developed. Its purpose is to generate traffic graphs in the form of rectangular grids.

This generator creates a grid with $N \times M$ points homogeneously distributed in space, separated from a distance d of each other and connected to their neighbour points by two edges,

one in each possible direction. All the edges of this graph have the same fixed number of lanes n_l as well as a fixed width l and flow f of vehicles per minute. Moreover, the configuration of state times for the traffic lights is calculated based on the distance d between the points of the connection it belongs to. This includes the time for the green, yellow, red light and also the initial wait time, as it will be described in section 3.3.



Nodes located in the four edges of the grid are not connected to any edge. Moreover, the nodes located in the grid borders are not connected to other nodes also located in the borders.

The result is a traffic mesh similar to the one presented in Figure 1. This example presents 5×4 junctions and $(5-2) \times (4-1) \times 2 + (4-2) \times (5-1) \times 2 = 34$ connections with 2 lanes each. From each junction located in the borders, f vehicles per

minute depart from one side of the road, while on the opposite side other vehicles arrive at their destination point.

To request automatic traffic graph generation, an extra configuration line is enclosed in the same archive where the edges and nodes are described for the manual configuration. This line specifies the variables used for automatic graph generation and can be enclosed in any point of the archive. If it is enclosed, other points and edges configurations are ignored. The format of this new line can be seen in Table 3.

Table 3: File line format specification of the automatic traffic graph generation.

C	N	M	d	F	n_l	l
A	5	5	100	10	3	3

The fields in Table 3 are the following: C - code informing that the graph must be generated automatically, N - number of lines containing nodes in the traffic graph, M - number of nodes per line in the traffic graph, d - distance between adjacent nodes in the traffic graph, f - flow of vehicles per minute in each connection, n_l - number of lanes in each connection and l - width of these lanes in meters.

2.1.3. Traffic Structures Mapping

After the traffic ways information is loaded in a graph inside of the Simulation class, this information is mapped to specific traffic structures contained in this class: the junctions and the connections. As a consequence of the high degree of dependence between a junction and the connections it contains, the initial configuration of these entities is done in three steps instead of in only two. First, all the connections of a junction have the variables whose values are dependent only on the values of the traffic graph initiated. They are temporarily stored in a list of connections in the Simulation class. Second, the junctions are created with its lists of connections. Finally, the constructor of the junction initiates, for each of its entrance connections, the variables dependent on the junction configuration. At this moment, each connection finishes the configuration of its internal variables and initiates its traffic light, if it exists.

According to the disposal of the edge that represents itself in the traffic graph, a connection can be classified as:

Entrance connection: it represents a road section that intercepts the borders of the traffic mesh. It simulates the traffic preceding from neighbouring areas that arrives in the region of the mesh. Connections of this type are the starting points of any vehicle in the simulation;

Exit connection: it also represents a road section that intercepts the borders of the traffic mesh. It simulates the traffic that leaves the simulated region and moves towards neighbouring areas. Connections of this type define the end of a path traversed by a vehicle;

Intermediate connection: it represents a road section that does not intercept the borders of the traffic mesh. It is, therefore, a road link completely contained in the simulated region and composes the path that takes a vehicle from an entrance to an exit connection.

Figure 2 illustrates each one of the three types of connections.

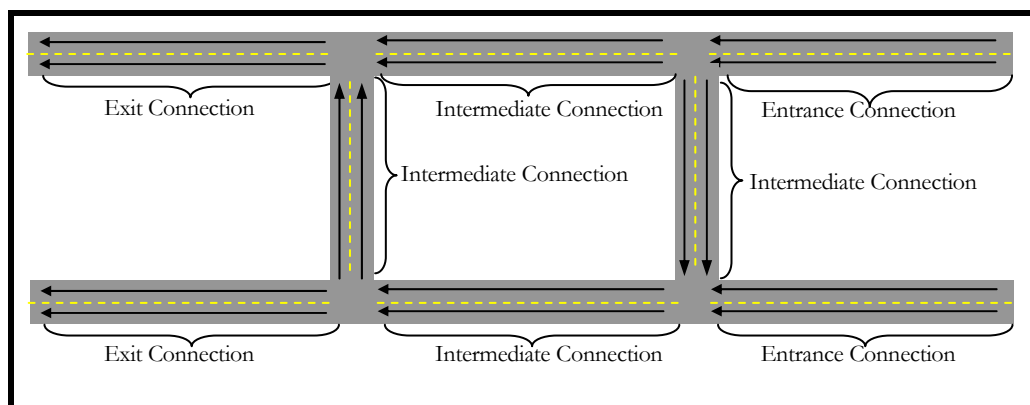
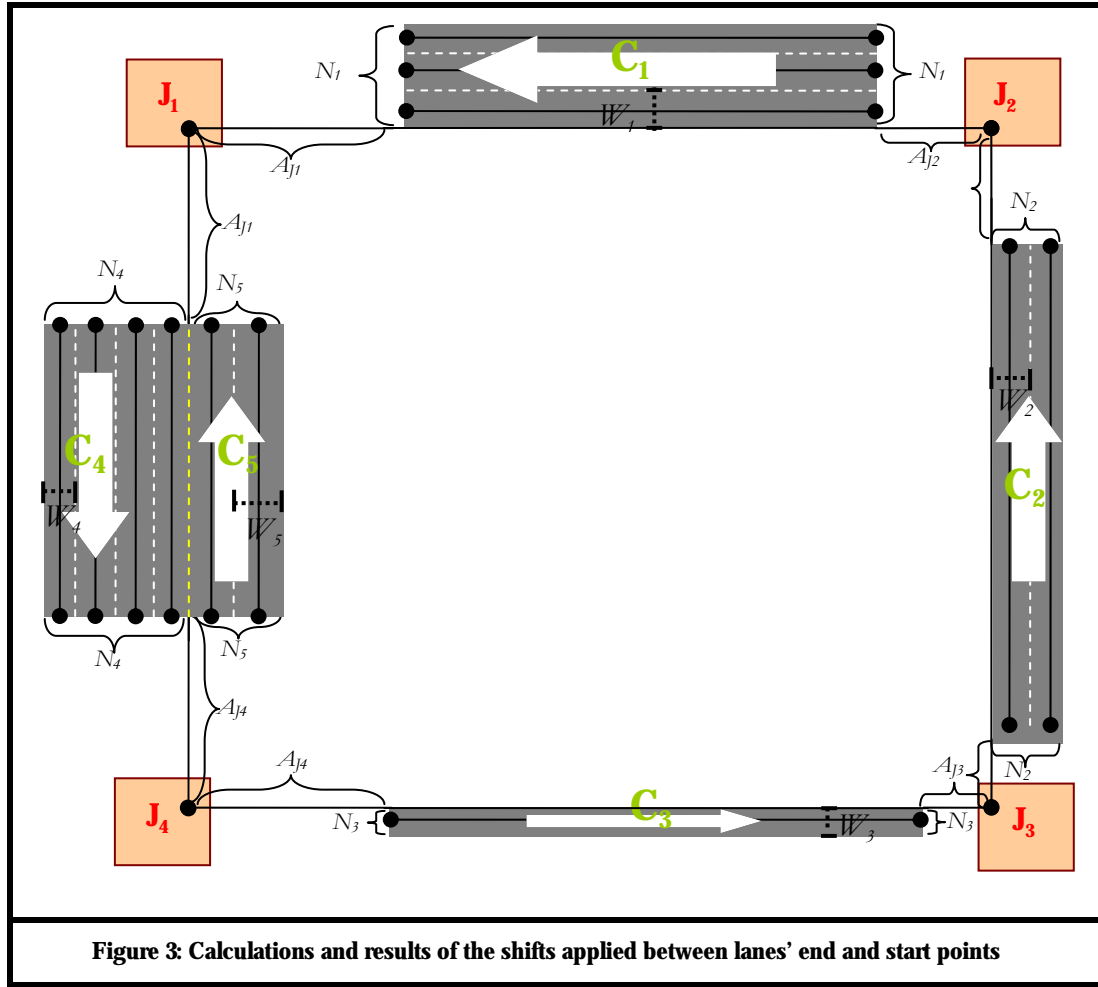


Figure 2: Types of possible connections in a traffic mesh.

Each connection has a certain number of lanes with a specific width. Lanes of the same connection have the same width. This information is obtained directly from the edges of the traffic graph. In the case of an automatically generated graph, each edge has the same configuration, except for the initial wait time of its traffic light.

Based on the information passed in its construction, a connection generates start and end points for each one of its lanes. Vehicles pursue these points during the traversal of its paths. The initial points of lanes in the same connection are lined up in a straight line perpendicular to the direction of the connection. However, this straight line does not cross the initial point of the connection, because a shift of a distance \mathcal{A} is applied to it. Being N the maximum number of lanes found among entrance and exit connections of a junction of which the current connection is part of and \mathcal{W} the maximum width found amongst the lanes of the connections in this same set of connections, the following equation is derived: $\mathcal{A} = \mathcal{W} \times N$. This shift is necessary so that the distance between the end point of the lane of a connection and the initial point of the lane chosen in the next connection is far enough for the vehicles to be able to make curves softly. The calculations and the results of these shifts can be seen in Figure 3.



In Figure 3, the following variables are presented: C_i - connection i , N_i - number of lanes in a connection i , W_i - width of lanes of a connection i , J_i - junction i and A_{ji} - lane point shift for J_i . The calculation of the shift for lanes in each connection presented in the above figure is shown below:

- $W_5 \geq W_1 \geq W_4$ & $N_4 \geq N_1 \geq N_5 \rightarrow A_{J1} = W_5 \times N_4$
- $W_2 \geq W_1$ & $N_1 \geq N_2 \rightarrow A_{J2} = W_2 \times N_1$
- $W_2 \geq W_3$ & $N_2 \geq N_3 \rightarrow A_{J3} = W_2 \times N_2$
- $W_5 \geq W_4 \geq W_3$ & $N_4 \geq N_5 \geq N_3 \rightarrow A_{J4} = W_5 \times N_4$

During the construction of the Simulation class, besides the configuration of the road mesh by input archive reading and structure adapting, a list containing inactive and available vehicles for insertion in entrance connections is created. According to its traffic flow, each

entrance connection removes vehicles from this list, generates their paths, activates and inserts them in the simulation. Likewise, vehicles that reach the end point of an exit connection are removed, deactivated and placed back in the list of inactive vehicles, being available for reappearance in the simulation when other vehicle solicitations in entrance connections occur. The function of the list is to limit the number of vehicles in the simulation. Moreover, it accelerates processing during the insertion and removal of vehicles in the simulation, since vehicles are already partially configured.

2.2. Terrain

The terrain is configured in two different ways, depending on how the traffic graph is specified. On the one hand, the manual specification allows the definition of any traffic network. On the other hand, the automatic specification generates rectangular meshes of traffic with different levels of complexity easily and faster than with the manual specification. The meshes generated by this second technique were used during the experiments presented in Chapter 5. Below, both methods of terrain surface creation are described.

2.2.1. Terrain Generation for a Traffic Graph Manually Defined

If the traffic graph is defined manually in an archive, by specifying each edge and node, the terrain must also be generated by a manual specification in a special archive. For each terrain area, this archive specifies, in a line, an identification number, a position, an orientation and the number of levels of detail. The format of this configuration line can be seen in Table 4.

Table 4: File line format for terrain area specification.

C	X	Y	Z	O_x	O_y	O_z	N
T	10.0	0.0	20.0	0.0	90.0	0.0	3

The fields in Table 4 are the following: C - code informing that a terrain is being described, X - coordinate of the terrain in the x axle, Y - coordinate of the terrain in the y axle, Z - coordinate of the terrain in the z axle, O_x - rotation angle of the terrain in degrees around the x axle, O_y - rotation angle of the terrain in degrees around the y axle, O_z - rotation angle of the terrain in degrees around the z axle and N - number of levels of detail of the terrain.

In the archive lines following the above specified terrain description line, the directory paths for avatars and textures are specified, including the activation distances for each of its level of detail. For each of these LODs, a new line is inserted. The LOD line format is shown in Table 5.

Table 5: File line format for avatar LOD specification of a terrain area.

<i>C</i>	<i>3D Object</i>	<i>Texture</i>	<i>D</i>
L	3dobjects\\terrain_default.3DS	textures\\terrain_defaultTexture_LOD0.bmp	100

The fields in Table 5 are the following: *C* - code informing that a level of detail of the avatar of a terrain area is being described, *3D Object* – directory path for the three-dimensional object representing the level of detail of the terrain avatar, *Texture* - directory path for the texture to be applied in the terrain avatar for this level of detail and *D* - minimum distance needed between the terrain and the user for activation of this level of detail.








It is important to highlight the use of two inverted bars to represent the archive of a subdirectory. Thus, the reference "3dobjects\\terrain_default.3DS" seen in Table 5, for example, specifies a three-dimensional object whose directory path is "/3dobjects/terrain_default.3DS".

The specification of the levels of detail must follow an increasing order according to the activation distances or, in other words, a decreasing order of detail of the three-dimensional model that represents the terrain. Therefore the model of the avatar with more details will be described in the line located below the other line that describes the terrain, followed by a less detailed LOD model description in the next line and so on, until the description of the model with fewer details is reached.

2.2.2. Terrain Generation for a Traffic Graph Automatically Defined

The terrain created in the traffic graph automatic generation is represented by simple avatars consisting of a square of fixed size. The applied texture, however, is configurable according to the LOD. The number of LODs was defined subjectively, by analyzing the minimum amount of LODs that would guarantee the perception of their change in diverse visualization situations of the scene. It was concluded that seven levels of detail were more than enough to guarantee this perception for most situations. However, some other number could have been chosen. The textures apply different colours for each LOD, which results in concentric circles painted in the terrain around the user. They confirm the terrains LODs correct application. The colours and minimum activation distances for each level of detail are presented in Table 6.

Table 6: Levels of detail with their activation distances and texture colours.

<i>Level of Detail</i>	<i>0</i>	<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>	<i>6</i>
Distance	0	100	150	200	250	300	350
Texture Colour							

The standard models and textures archives for the automatic terrain generation may be modified. The avatar, in the 3ds¹ format, can be substituted or have its size adjusted in a three-dimensional (3D) modelling tool. In the same way, textures can be substituted or modified in a graphic or image editor which supports the bitmap format. The visualization of the terrain levels of detail can be seen in the application figures 24 and 25 in section 4.2.

2.2.3. Buildings and Urbanization

Since no interaction between vehicles and terrain has yet been defined, other objects which are not terrains can be inserted in the scene to better represent a region. They are treated as if they were a terrain and are drawn in the scene with a position, an orientation and with levels of detail for its avatar.

Due to little importance of these objects in the scene, it was not created a specific class to represent them. An improved use of these entities and the specification of their classes are out of the scope of this work.

2.3. Vehicles

Besides the archives for specification of the traffic graph and its region terrain, another archive was created to store configuration for the vehicles in the simulation. At the moment, however, this archive is only used to define the maximum total number of vehicles possible inside the simulation. This is done by defining an identification code, followed by a space and the desired number of vehicles, as presented in Table 7.

Table 7: File line format for specification of the maximum number of vehicles simulated.

<i>C</i>	<i>N</i>
N	100

¹ 3ds: exportation format of the 3D modeling tool 3D Studio Max®, produced by Discreet© (Discreet, 2005).

The fields in Table 5 are the following: C - code for definition of the maximum number of vehicles and N - value for this number.

Chapter 3 :

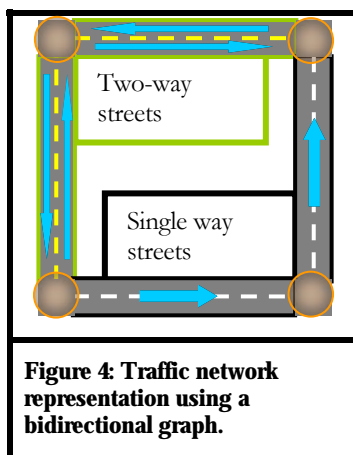
Simulator

The simulator is the application core. It is responsible for making all entities of the virtual world be put into motion and interact between themselves. This chapter describes its architecture and its internal functioning in terms of the behaviour of its virtual entities, which are the vehicles and the traffic lights.

3.1. Simulator Architecture

The ItranS simulator was developed with an architecture whose processing is done junction by junction, similar to the one suggested by Cameron and Kleins (Cameron, 1994; Klein, 1998). This architecture allows the extension of the system to a distributed model, once each junction has the processing of its vehicles carried independently, allowing the distribution of each junction in different machines. Theoretically, this processing distribution also allows the geographic scope of the virtual world to be increased without performance loss. Therefore, the increase of data to be processed is limited only by the number of available machines to be used in the distribution.

The traffic network mapping of a region is done using a bidirectional graph, as illustrated in Figure 4.

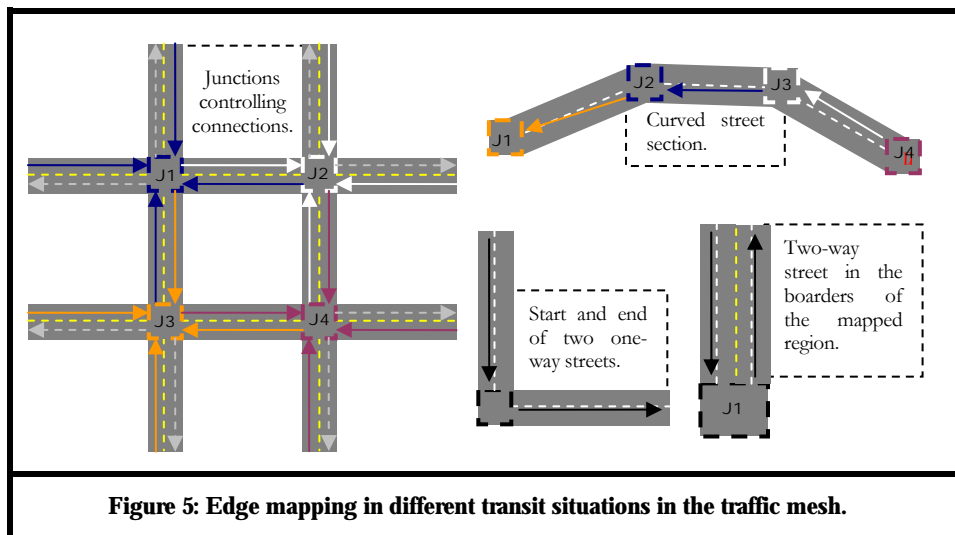


Graph nodes, which represent junctions between streets and also exit and entry points for the traffic of a region, are present in every beginning and end of each contiguous street section, that is, in a street section that does not intercept a junction. Each node is then transformed into a traffic junction.

The graph edges represent a set of one-way lanes in a contiguous street section. A one-way street section is represented by a single edge. However, a two way street section is represented by an edge for the lanes going in one direction and by another for the lanes going in the opposite direction.

Contiguous street sections that are not rectilinear can be represented by a set of edges, forming a poly-line to simulate its curvature. Each edge will later be transformed into a traffic connection during the traffic graph traffic mapping to the application classes. The traffic connection description is located in this section. As a result, any type of street can be represented in the traffic graph by an edge and two nodes or a set of edges and nodes.

The proposed architecture divides the simulation model traffic among junctions according to the graph nodes. Each junction controls traffic contained only in its edges of incoming traffic. If a junction contains only one incoming edge and one outgoing, it represents a segment of a poly-line that defines a curved street, the meeting point between a street start and another's end or a two-way street section located in the borders of the mapped traffic mesh. These representations are illustrated in Figure 5.



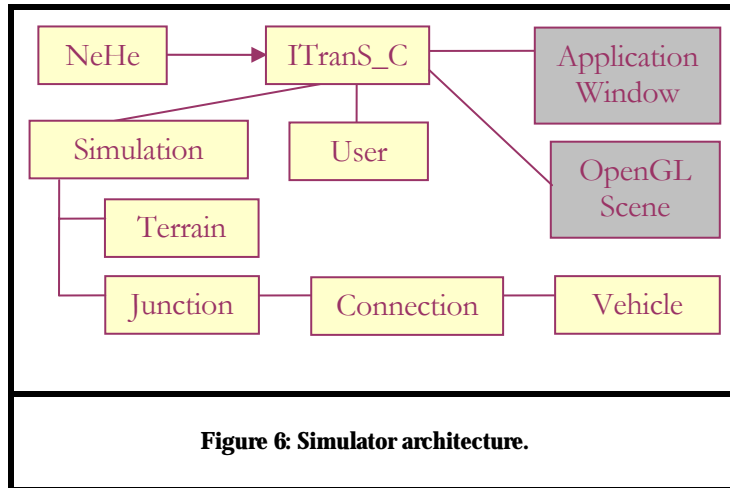
A junction has incoming edges, from which the junction traffic arrives, and outgoing edges from which the junction traffic flows out leaves it. The traffic in the incoming edges is controlled by the junction that receives its traffic. The outgoing edges of a junction are controlled by adjacent junctions that consider them as incoming edges and receive their traffic flow. Thus, amongst the edges to which it is connected, a junction only controls its incoming edges.

By organizing the traffic in this way, the mesh has its traffic uniformly distributed between the junctions. As a consequence of this organization, it is expected that the distribution of the traffic processing becomes possible, as well as the expansion of the model without changes in its structure.

The construction of the three-dimensional scene is made by obtaining the graphical information of the objects being controlled by the junctions. Before sending this information to the user, each junction makes the adjustment of the level of detail of each object according to the distance between them and the user. The idea is to reduce graphical processing of the visualized scene as much as possible.

The classes which compose the system were created based on this nodal model of distributed processing. Figure 6 represents the simulator architecture and the relation between the application main entities, represented by the following classes:

1. User: avatar and point of view with which the user navigates and observes the scene. Its interface is described in Chapter 4;
2. Simulation: contains all the junctions and the terrain. It controls virtual environment functioning as a whole;
3. Junction: the entity which represents the crossings and most of the independent processing units;
4. Terrain: three-dimensional topographical map of the region. It represents the remaining portion of the independent processing units;
5. Entrance connection: streets sections with pre-defined vehicle flow controlled by a junction. They may have one or more lanes. They control the vehicles traversing it;
6. Vehicle: each mobile entity which dislocates along the lanes of the street sections which are part of its path.



In the same way that graph nodes are mapped into junctions, the edges are mapped into connections. Thus, all the structures represented in the traffic graph can be stored inside the simulation classes.

As computer processing is centred in this version, an extra entity was created. The ITranS_C class contains: the entire simulation environment, initiating and controlling the scene in OpenGL, the terrain, avatars and the user interaction, as well as all junctions, each with a list of entrance connections containing their respective vehicles and traffic lights. This class may be considered the basis of the application. The NeHe graphical library (Molofee, 2004) was used to simplify the configuration of a window in the operational system containing the OpenGL environment. It can be seen as a set of methods supporting the application.

Four main methods compose the ITranS_C class, defined in the interface supplied by NeHe graphical library: initialize, update, draw and deinitialize. The first one of them, initialize, is used to initiate the application. In it, the window, the graphical scene, the user and the simulation are sequentially initiated and configured. The internal functioning of this method can be seen in Figure 7.

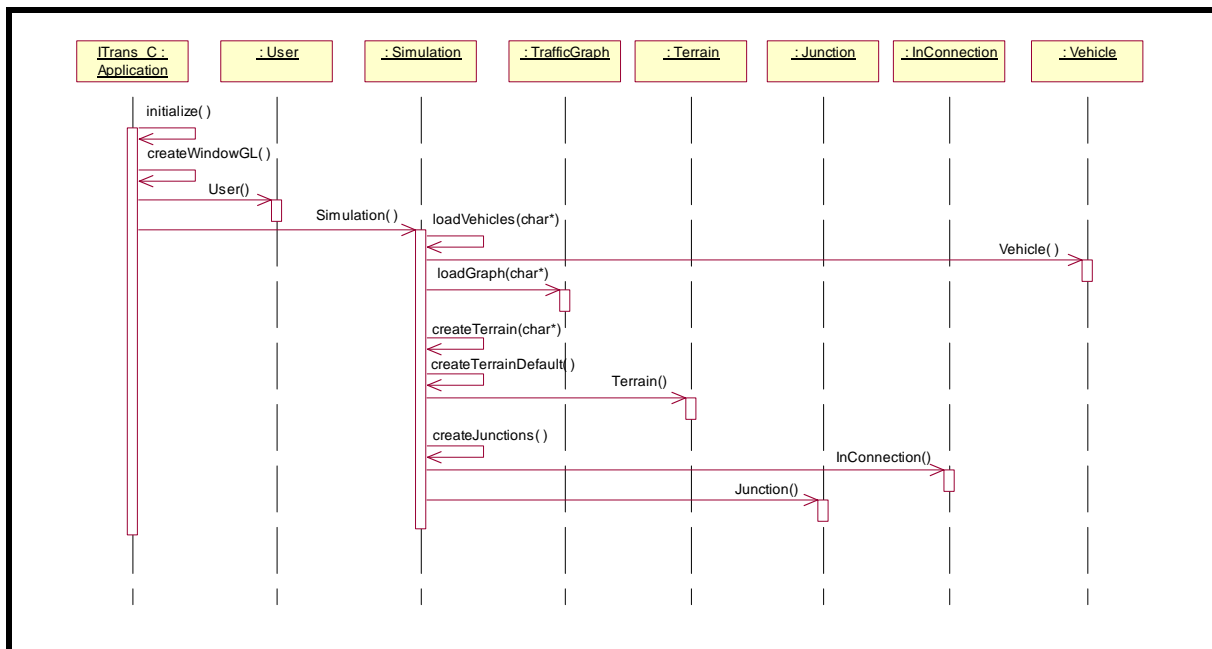


Figure 7: Sequence method diagram of the initialize method.

The method update is responsible for processing input commands and updating the scene. Its internal functioning is described in Figure 8.

This method calls the method simulate in the Simulation class to simulate junctions. These, in turn, simulate their incoming connections. These connections simulate active vehicles moving along their lanes. If the connection is an exit connection, the vehicles which reach its end point are set to inactive and transferred to the simulation's list of inactive vehicles, by calling the method pushVehicle(). If it is a start connection, that is, if it is the first connection in a vehicle path, it will insert, close to its initial point, new vehicles to traverse new paths in the simulation. Otherwise, if the connection is located out of the borders of the map, it is set as a intermediate connection. Such connection obtains vehicles from other connections and transfers them to the next connection which composes the path of each vehicle, by calling the method transferOutGoingVehicles(). In a distributed version of the simulation, it would be in this point that the transmission of information of vehicles between connections would occur. The start connection generates entrance times in an one minute interval according to its traffic flow, by calling the method generateVechileEntranceTimes(). This is more thoroughly explained in section 3.2.1. When the entrance time of a new vehicle is reached in a specific one minute interval, the vehicle is removed from the simulation's list of inactive vehicles by calling the method

popVehicle(), has its path generated by the method generatePath() and is set to active by calling the method activate() in order to be simulated in the next frame update.

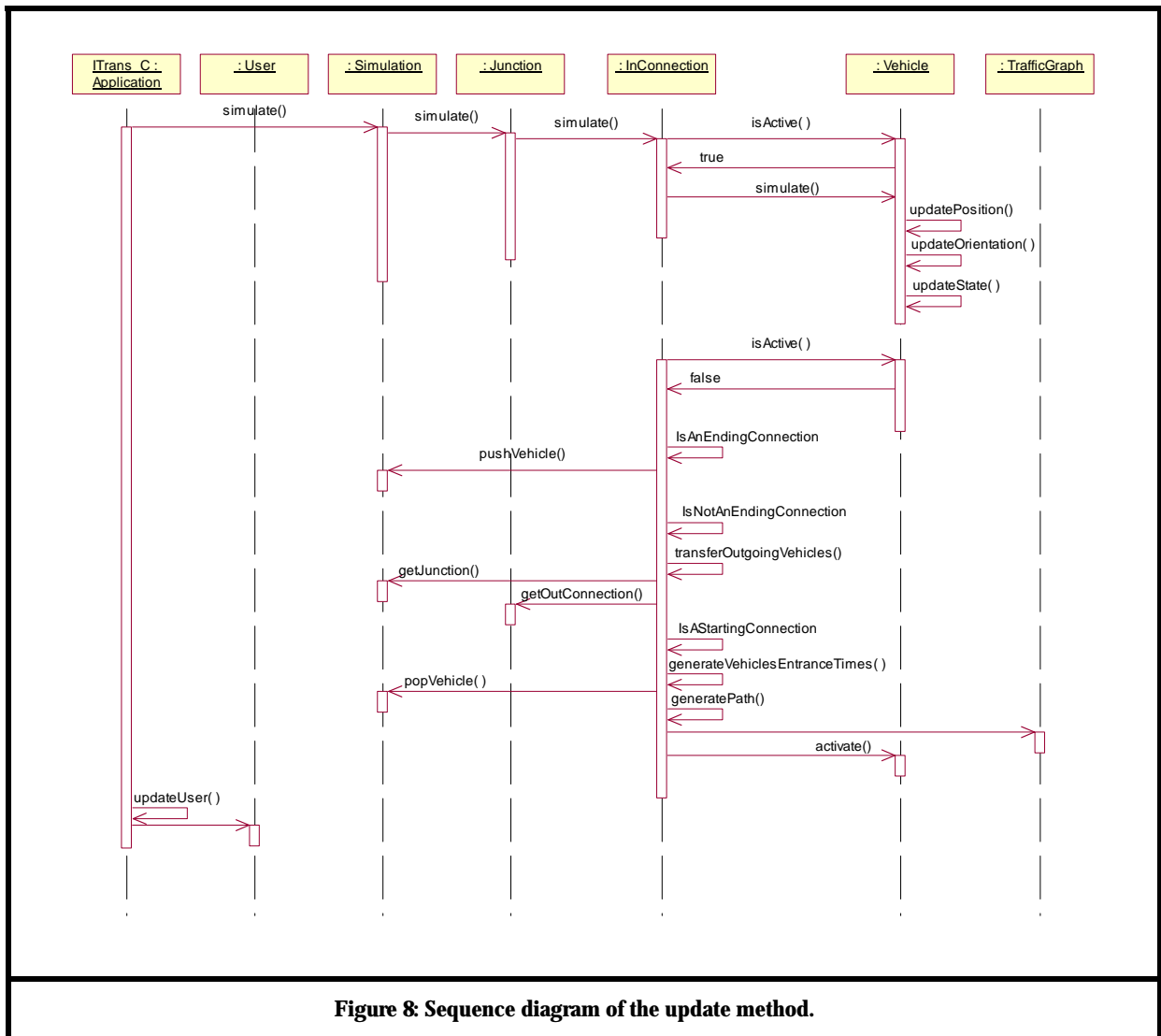


Figure 8: Sequence diagram of the update method.

The third method, draw, graphically builds the already updated scene. The internal functioning of this method can be seen in Figure 9. First, this method configures the point of view of the user in the scene according to its position, by calling the method updateUserView(). Then, the method draw() of the Simulation class is called, and requests the junctions and terrains to be drawn. Then, junctions themselves draw their incoming connections. These connections, in turn, draw the vehicles contained in their lanes and which they control. After this stage, the user interface is drawn.

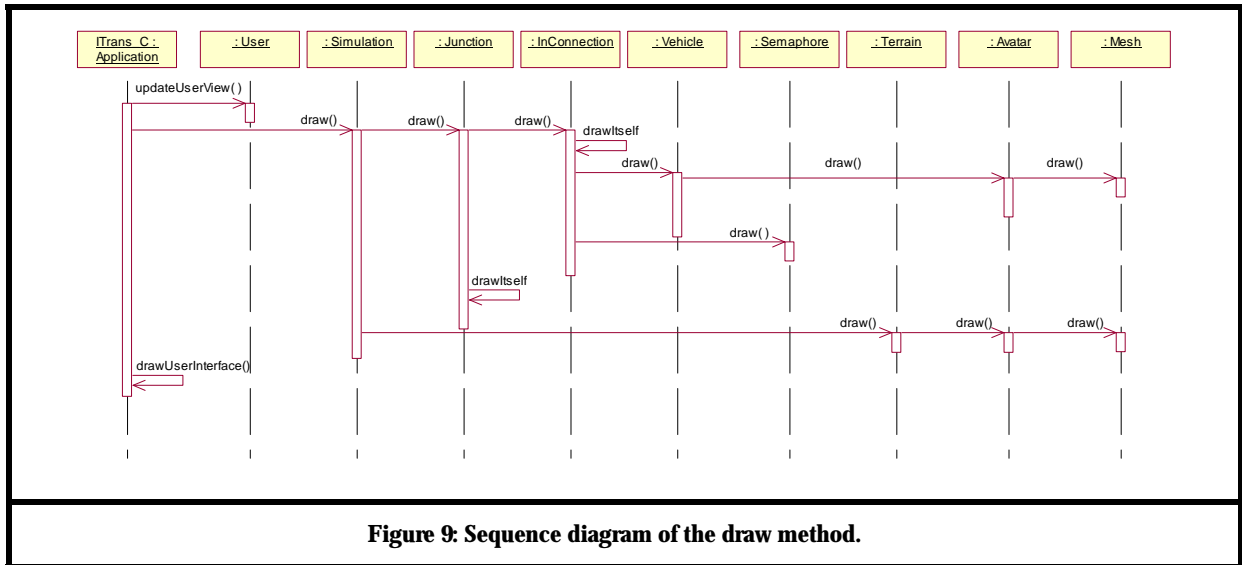


Figure 9: Sequence diagram of the draw method.

Finally, the method `deinitialize` assures that all structures are correctly destructed when the application finishes. The internal functioning of this method can be seen in Figure 10.

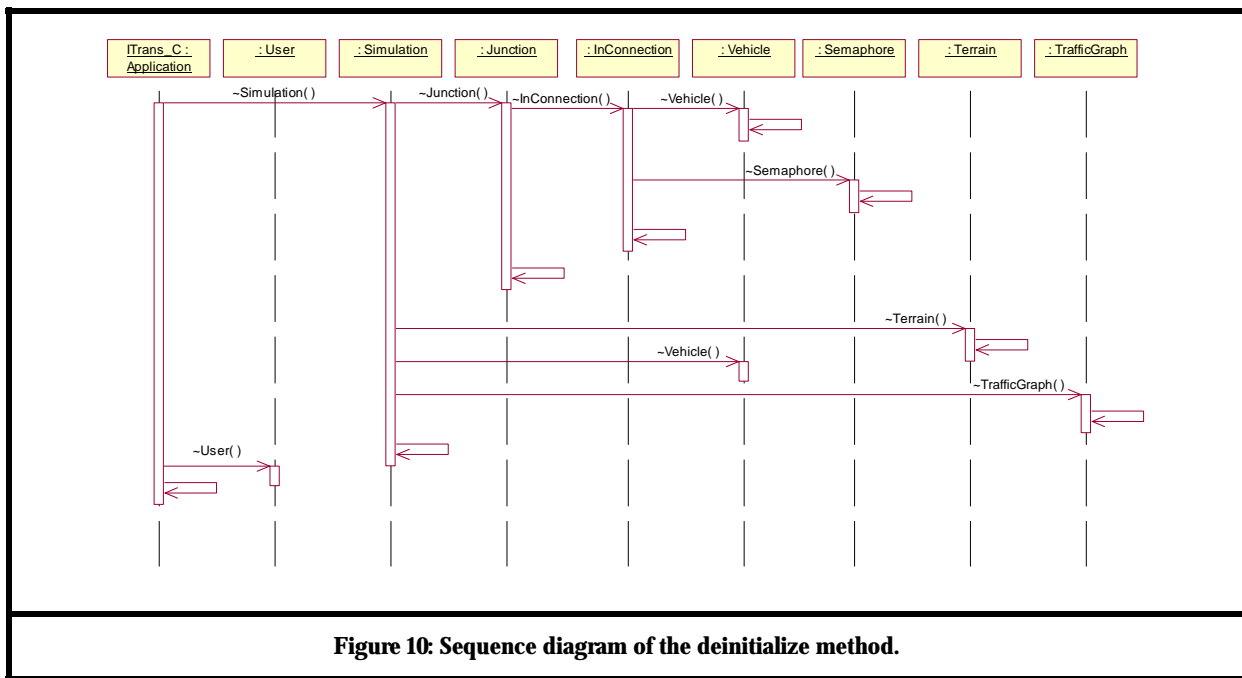
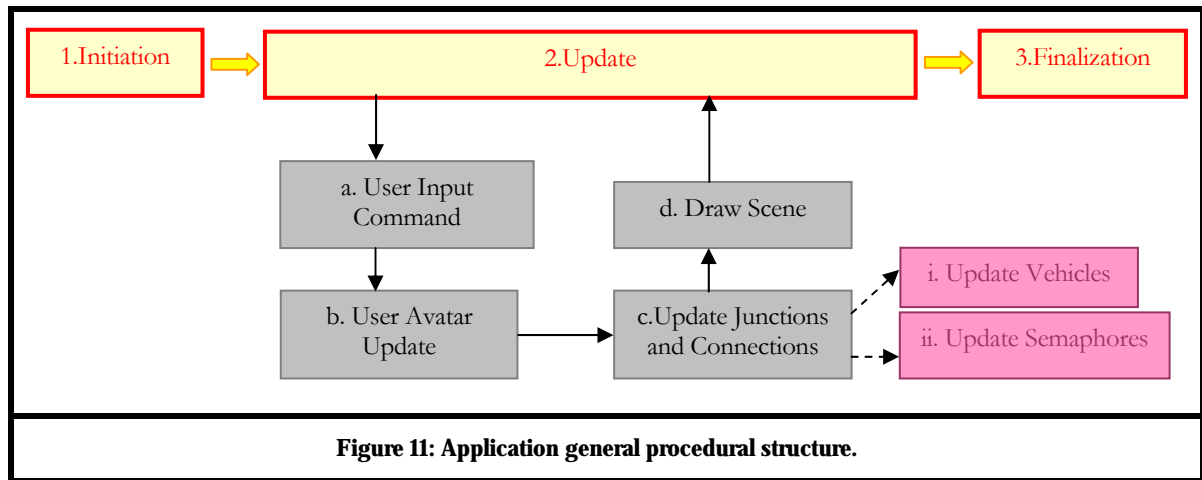


Figure 10: Sequence diagram of the deinitialize method.

This method calls the destructor method of the `Simulation` class which, first, destroys junctions. The junctions then destroy their incoming connections. These connections, in turn, destroy the vehicles contained in their lanes and which they control. Having destroyed all the junctions, the destructor method of the `Simulation` class destroys all terrains, the vehicles list and, finally, the traffic graph and other variables. Having the destructor method of the `Simulation` class concluded its execution, the destructor method of the `User` class is called by the method `deinitialize` and the application finishes.

The application can be better understood by the illustrated procedural structure in Figure 11.



As an avatar represents the terrain, the update of its levels of detail happens when the scene is graphically built. The same is done to the vehicle avatars. The complete classes' description can be seen in appendix A.1.

3.2. Vehicles' Paths

The path of a vehicle consists of a list of connections names contained in the traffic network, by which the vehicle must sequentially pass by. These names are, in fact, only identification numbers distributed to each connection during their initial configuration.

The generation of the vehicle path happens by the sequential choice of linked connections in adjacent junctions. The first connection contained in a vehicle path is always an entrance connection in which this vehicle is activated. From there on, an execution cycle is followed until an exit connection is reached, with the steps described in the order below.

1. First, a random value V_A between 0 and 1 is chosen. This value will represent the path choice of the user.
2. After that, the entrance probability P_{ei} is calculated for each one of the exit connections of a junction. The probability of choice of a connection is calculated by dividing its flow by the total outflow of a junction. An example of this calculation is shown below.

Suppose a connection C is connected to two other connections $C1$ and $C2$ by a junction J . If $C1$ has an average traffic flow of A vehicles/minute, while $C2$ has a flow of average

traffic of B vehicles/minute, then the probability of a vehicle to leave C and go to the $C1$ connection is of $A/(A+B)$, whereas the probability of this same vehicle to leave C and go to the $C2$ connection is of $B/(A+B)$.

The example of Figure 13 presents a junction with three exit connections. Assume that C_i is a connection i , $P(C_i)$ is the probability of a connection i , F_{C_i} is the flow of a connection i , F_{total} is the total exit flow of the junction, and I_{C_i} is the probability interval of a connection i , the calculation of the total flow and probabilities for each connection is done as follows:

$$F_{total} = F_{C1} + F_{C2} + F_{C3}$$

and

$$P(C_1) = F_{C1}/F_{total} \quad P(C_2) = F_{C2}/F_{total} \quad P(C_3) = F_{C3}/F_{total}$$

Thus, the probabilities for each of the three junction outgoing connections are correctly defined.

3. Once calculated the probabilities, these values are organized in an interval from 0 to 1, so that interval values are defined and attributed to each connections, as shown in Figure 12.

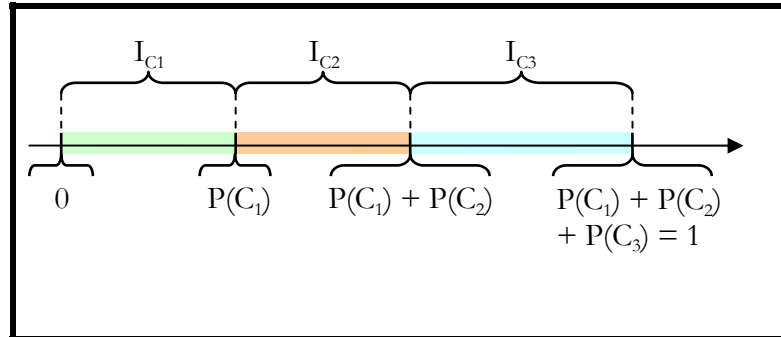


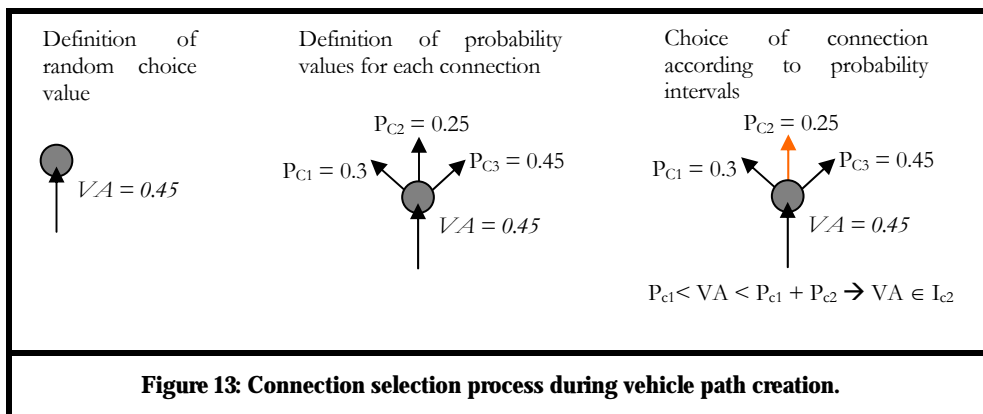
Figure 12: Probability interval calculation for outgoing connections located in a junction during vehicle path generation.

4. Finally, the chosen connection will be the one whose probability interval spans the previously chosen vehicle random value. In case this connection already belongs to the vehicle path and if there are other outgoing connection options in the junction, another connection from this junction, and which does not belong to the vehicle path, is chosen. If there are no other connection options in the junction or if the existing options also belong to the vehicle path, the initially chosen connection is reinserted in the path, though for the second time.

Once the connection has been chosen, the lane to be followed will be, similarly to the entrance connection, chosen randomly. Nevertheless, the lanes are chosen during the path traversal, in real time. That means the lanes are chosen only after vehicle path has been completely defined. For bi-directional streets, the chosen lanes are, for obvious reasons, restricted to the direction currently being considered. Moreover, to accomplish the change from one road link to the other, no transition to a lane closer to the next road link happens. The vehicle changes to the next road link independently of on which lane it is.

5. If the connection is an exit one, then, after stored in the path, the vehicle path creation will be finished and no other activity is needed. Otherwise, the path creation continues in the junction where the connection the vehicle should follow is and the above described cycle is restarted.

This basic cycle is seen in Figure 13. A more in depth approach to the functioning of this and other algorithms can be found in the appendix A.3.



3.2.1. Initial Configuration for Vehicles with a New Path

The initial configurations of the vehicles are done as follows: the vehicles are inserted in the simulation only by entrance connections, situated in the borders of the simulated traffic graph. Dead end streets will also work as entrance points as well as exit points. Therefore, these links will also be classified as exit or entrance connections.

Each entrance connection has a flow value of FC_i vehicles per minute. At each minute, the entrance connection will generate an internal list with FC_i random values ranging from 0 to 60. These values will represent the new entrance times for FC_i vehicles. The connection counts the time elapsed during each simulation minute. When a random time value is equal to the value

presented in the connection minute counter, this value is removed from the list and a new vehicle is requested by the entrance connection for insertion in the simulation.

As previously mentioned, the vehicle is initially created without having its path, being partially configured and placed in a list of available vehicles inside of the Simulation class for use by the connections. Only when a new vehicle is requested by an entrance connection is that a path is generated for this vehicle. Moreover, in this moment is also when the vehicle is activated and re-inserted in the simulation in a random lane inside of the connection that activated it.

As the lanes were randomly chosen, the traffic flow in a connection is homogeneously distributed amongst them.

It is important to highlight that the initial and maximum speeds, as well as the mass of each vehicle, are defined randomly, based in the average value of these features in domestic vehicles currently available in the marketplace. These values can be seen in Table 8.

Table 8: Velocity and mass value variation for vehicles in the simulation.

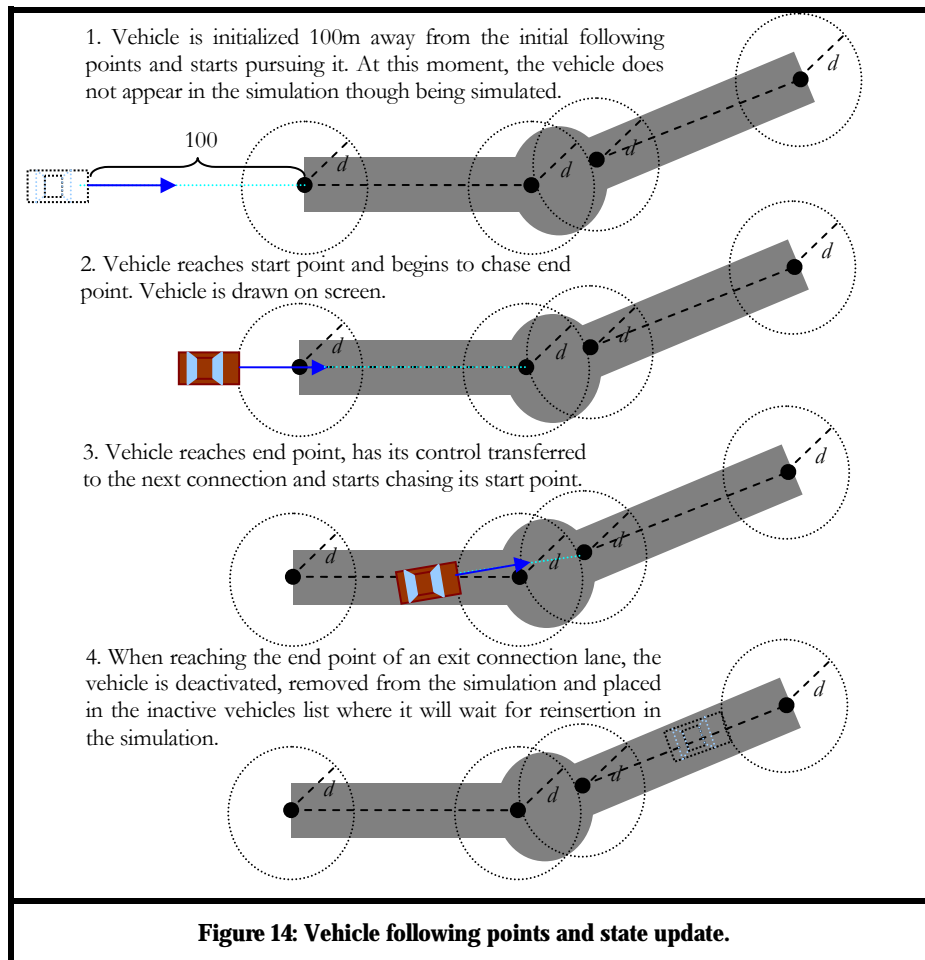
<i>Attributes</i>	<i>Values</i>
Initial speed	0 m/s
Maximum speed	35m/s – 50m/s
Mass	800kg – 1300kg

3.2.2. Updating the Vehicle to its Next Destination Point

The points sequentially followed by the vehicle are the start and end points of the randomly chosen lanes in each connection in the vehicle path. Initially, the vehicle starts 100 meters away from the initial point of the lane of the first connection contained in its path, so that the vehicle is located outside traffic graph defining the virtual environment. This deflection is necessary to stabilize the movement of vehicles before they are able to enter the traffic lanes. At this moment, the vehicle does not appear in the simulation yet.

The first point to be followed is exactly the initial point above described. When reaching a minimum distance d_m , defined when the application is initiated, from this point, it is assumed that the vehicle has arrived in it. Then the vehicle starts to follow the end point of this lane and only then the vehicle is graphically represented.

When the minimum distance is also reached for this point, the vehicle is deactivated and waits for the transference of its control to the current connection which composes its path and which is contained in another junction. The vehicle leaves the control list of the current connection and is inserted in the vehicles control list of the next connection. At this moment, the vehicle is reactivated by the new connection, has a randomly selected lane chosen and, thus, the pursuing cycle to the initial and final points of the lane is restarted. This process, illustrated in Figure 14, continues until an end point of a lane of an exit connection is reached.



3.2.3. Vehicle Final Destination Arrival Verification and Finalization

If the current point is the end point of a lane in the last connection contained in its path, the vehicle has its configurations restarted, is deactivated and placed in the list of inactive vehicles. It is then available for other connections calls, waiting its turn to re-enter the simulation.

3.2.4. Possible Vehicle States

The vehicles could be active and moving in the road links or inactive and occult, waiting for insertion in the simulation or transference between connections of adjacent junctions. The scene will be composed by a limited number of vehicles, which are activated and inserted in the simulation according to the traffic flow information of each connection. The inactive vehicles are organized in a row so that a fair activation order guarantees the participation of all the vehicles in the simulation.

3.3. Traffic Lights Simulation

The traffic lights are entities contained in each connection and exist to control its traffic flow. Each traffic light controls only the flow of the connection in which it is contained and can be understood as state machine. Four are the states existent in the traffic light during its functioning. They are presented below according to their order in the functioning cycle. However, the first of them only occurs in the first cycle.

Initial wait state: the traffic light is set to this state before starting its infinite cycle of three basic states: green light, yellow light and red light. When in this state, the traffic light waits a time interval T_w , whose objective is to synchronize traffic lights in connections whose flows are directed to the same junction or in adjacent connections composing a street or avenue. As a consequence of this synchronization, it is possible to prevent collisions of perpendicular flows in a junction and to generate the effect "green wave", when blocks of vehicles pass uninterruptedly from start to end of a street or avenue;

Green light: state indicating the traffic light is open, allowing vehicles to pass from one connection to the next;

Yellow light: intermediate state indicating traffic flow is going to be stopped by the red light. In order to reduce complexity due to the available time, it has the same effect as the green light, not affecting the flow of traffic or the behaviour of the vehicles. It was used merely to give realism to the simulation;

Red light: in this state, vehicles in the connection are obliged to stop next to the end point of the connection and wait for the traffic light to turn green again.

Besides these four, a fifth state also exists. However, this state does not belong to the basic cycle of the traffic light. On contraire, the so-called "inactive state" indicates that the traffic light is

not working or is deactivated. This type of state is used when it is intended to simulate internal semaphore malfunction or a blackout in the region where the traffic light is located.

3.3.1. Semaphore Configuration for Manually Generated Graph

For the traffic light to be correctly initiated when a traffic graph is manually configured, four time values are specified to indicate the change between its states and to define its total time cycle. These values are obtained from reading the traffic graph edges, as previously mentioned in Chapter 2 and whose line format can be seen in Table 2. These values are of the floating-point type and they are defined in seconds. The function of each of them is described below:

- T_g : final time for the green light. This value indicates the time in seconds, inside of the time cycle, when the traffic light must pass from green to the yellow light state;
- T_y : final time of the yellow light. This value indicates the time in seconds, inside of the time cycle, when the traffic light must pass from yellow to the red light state;
- T_r : final time of the red light. This value indicates the total time in seconds of the time cycle of the traffic light. It indicates, consequently, the time where the traffic light state must pass from red back to the green light state again and have its time cycle counter restarted;
- H_r : initial wait phase of the traffic light. This value indicates for how many seconds the traffic light must remain halted in the red light state before it initiates the time cycle counter and its normal time cycle.

With base in these values, four steps sequentially compose traffic lights' functioning:

1. An initial cycle of H_r seconds is initiated. During this period the traffic light state is red. As already mentioned, this step is used to synchronize adjacent traffic lights, and it happens only once, when the traffic light is initiated. It can be understood as a first particular and initial cycle. Passed H_r , the normal cycle of traffic light states starts;
2. The traffic light time cycle counter is activated and initiated, if necessary, and the state of the traffic light is set to green. The traffic light remains in the green light state until the counter reaches time T_g ;

3. The traffic light state is set to yellow. The traffic light remains in the yellow light state until the counter reaches time T_y ;
4. The traffic light state is set to red. The traffic light remains in the red light state until the counter reaches time T_r . When this time is reached, the cycle counter is restarted and the cycle is repeated from step 2.

The vehicles have direct access to the traffic light state of the connection where they are and by which they are being controlled, thus, being able to vary their behaviour according to this information.

If a traffic light does not have its values passed in the configuration line of the edge containing it, it will not be created. In case one of the four values passed to create the traffic light is negative, the traffic light is configured with the standard values presented in Table 9.

Table 9: Standard traffic light time values.

T_v	T_a	T_r	H_{ii}
26s	29s	60s	30s

If the values passed to T_v , T_y or T_r is zero, the traffic light will be created, but it will not be activated. A traffic light in this situation is set to the inactive state.

3.3.2. Semaphore Configuration for Automatically Generated Graph

If a traffic graph is generated automatically, the calculation of the traffic light times is done as follows. Given an initial standard wait time t for the traffic lights, a traffic light of a connection located in the iih position of one of the junction grid rows has an initial wait time $t_i = i \times t$. The same logic is applied for traffic lights in the opposite direction. Thus, the traffic light times are approximately synchronized to guarantee the flow of continuous traffic, generating the colloquially called "green waves".

For this effect to work in any traffic graph, t must have a value equal to the average time which a vehicle takes to dislocate between adjacent junctions. Being V_m the average speed of a vehicle in the simulation and D_j the distance between two adjacent junctions, the value of t is:

$$t = D_j / V_m$$

The time values T_g , T_y and T_r for the green, yellow and red states respectively are also calculated based on t as shown below.

$$T_g = t - 4;$$

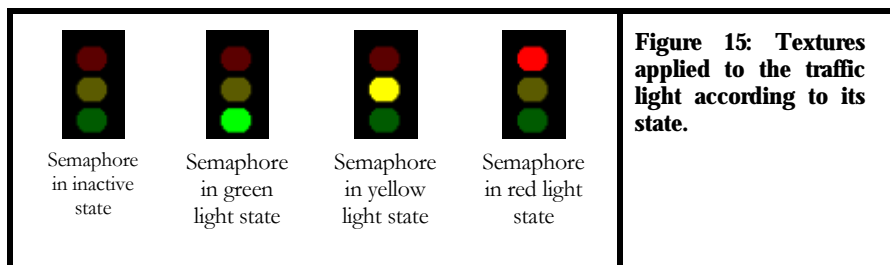
$$T_y = t - 1;$$

$$T_r = 2 \times t;$$

That way, during $t-1$ seconds of the time cycle, traffic will flow freely in the connection whereas in the rest of the time, $t+1$ seconds, vehicle flow will be interrupted. From the $t-1$ seconds of traffic flow, 3 seconds are spent on the yellow light traffic state and $t-4$ seconds on the green light state. These values had been subjectively defined, based on personal experience with transit lights cycle times. Moreover, in the $t+1$ seconds of flow interruption, the first second is called the state of "full red". In this second, the traffic lights controlling the transversal flow do not enter in green light state, remaining in state of red light too. The function of this full red state is to provide a complete stop of flow, during which all the semaphores of the junction are red and no traffic is flowing, thus giving an extra time for vehicles still located in the middle of the junction to leave there before the traffic light that controls the transversal flow turns to green and liberates more vehicles in the junction. In the automatic terrain generation, the minimum values for T_g , T_y and T_r are 3, 6 and 14 respectively. These values had been defined to prevent that the traffic light deactivates when the junctions are close to each other or when $t = Dj/Vm < 4$, which would imply in $T_g = 0$ and would consequently deactivate the traffic light.

3.3.3. Graphical representation

The traffic lights are graphically represented by a rectangle located 4 meters above the higher speed lane of a connection, that is, above the lane located more to the left and closer to the centre of the street. Different textures are applied for the green, yellow, red and inactive light states, as seen in Figure 15.



3.4. Vehicle Behaviour

A vehicle has diverse behaviours. They are classified in three types: the ones that modify the vehicle acceleration, the ones that modify its speed and the ones that modify its position.

The separation and pathFollowing behaviours apply a force to the mass of the vehicle. This generates acceleration and, as a consequence, modifies its speed. The addition of the forces applied to the vehicle by these behaviours generates a resultant acceleration that affects the automobile speed.

The parking behaviour, however, directly affects speed by gradually diminishing its value until the vehicle arrives in its parking point, independently of the forces applied to the vehicle by other behaviours. This behaviour is used when the vehicle must stop in a red traffic light.

Finally, the collision behaviour directly modifies the vehicle position, placing it in its previous position if any of its collision points intercepts the collision area of another vehicle.

Each one of these behaviours is presented in detail below.

3.4.1. Vehicle Position Update Based in its Current Target Point

Being P the position of a vehicle V_i , V its current speed, V_{max} its maximum speed, p_i the initial point of its current lane, p_f the end point of its current lane, $D = p_f - p_i$ the direction vector of the lane and D_{norm} the D vector normalized, the ideal speed V_i for a vehicle is:

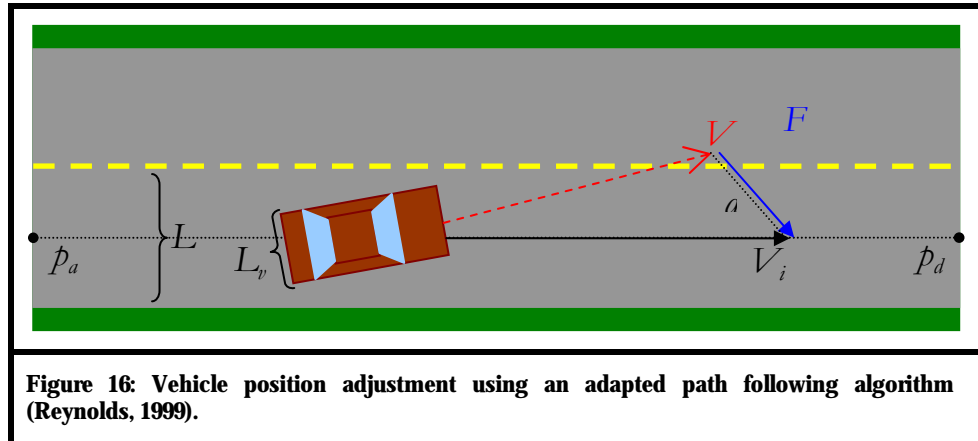
$$V_i = D_{norm} \times V_{max}.$$

Being d_p the maximum distance allowed between V_i and V , W_v the vehicle width and W_f the lane width, the value of d_p is:

$$d_p = W_v/2 - W_f/2 = (W_f - W_v)/2.$$

Considering the position of the vehicle as being the centre of its avatar, if the distance d between V_i and V is greater than d_p , this indicates that the speed V of the vehicle is distant from the edge of the lane by more than half of its width. As a consequence, in the next second, when the vehicle will be dislocated from a distance V , the vehicle will have the extremity of its body distant half of its width, being located outside the lane. To avoid this, a correction force F is calculated. Its orientation is the same of the vector $V_i - V$. This force generates acceleration in the vehicle and

makes it return to the lane. The value of this acceleration will be greater if the vehicle mass is smaller and if the distance d is greater too. Figure 16 presents an example of the functioning of this mechanism.



3.4.2. Red Light Vehicle Parking

A vehicle does not have its parking behaviour activated if the traffic light of the connection controlling it is either in the green, yellow or inactive states. In the red light state, however, the vehicle changes its normal path following behaviour to park as next as possible to the end of the its lane and, consequently, also as next as possible to the traffic light. For the implementation of such behaviour, connections and vehicles were added with extra data structures, similar to the ones presented in (Cameron, 1994).

The connection controls the entrance order of the vehicles in each lane by a ticket system. This connection stores a value of ticket for each lane, which indicates the number of the ticket to be delivered to the next vehicle entering the lane. When a vehicle enters a lane, it receives the number of that lane current ticket and the connection increases in one the value of the ticket to be delivered to the next vehicle entering that lane. Observing the ticket number of each vehicle, it is possible to control the entrance order of vehicles in each lane and, therefore, control their parking positions close to the traffic light.

If a vehicle V_c is in a lane l of a connection C in a position p and the traffic light S of C enters the red state, its point of destination p_b which might be set to the initial point of the lane p_i or its end point p_f could be substituted by a parking point p_o according to the conditions below.

Being D_{p_i} the distance between the position p of V_c and the initial point of the lane p_i and being D_{p_e} the distance between this same position p and the parking point p_e , p_d values varies according to the following rules:

1. If $p_d = p_j$, then $p_d = p_e$;
2. If $p_d = p_i$ and $D_{p_i} \geq D_{p_e}$, then $p_d = p_e$;
3. If $p_d = p_i$ and $D_{p_i} < D_{p_e}$, then nothing happens.

If the traffic light is red and if the vehicle V_c reaches the minimum distance required to assume that it arrived at its point of destination p_d and if this point is equal to p_e , the vehicle enters a wait state and it is not put into motion until traffic light is green. When the traffic light is green again, V_c 's destination point p_d receives the value previously being chased, had it been either p_i or p_j , and V_c continues to follow its path normally.

The parking point p_e is calculated based on the number of vehicles in a lane and on the ticket number of each vehicle. If n vehicles exist in the same lane of a vehicle V_{c1} with smaller ticket numbers than V_{c1} , the parking point of this vehicle will be:

$$p_e = p_j - n \times (L_v + 1) \times V_{norm},$$

where V_{norm} is the normal indicating the opposite direction of the lane, $V_{norm} = p_i - p_j$, and L_v is the vehicle length. The value of $1 m$ (meter) added to Cv represents the distance the vehicle must keep from the other vehicle in front of it.

3.4.3. Vehicle Collision Calculation

Each vehicle has four collision points, located in the most external vertices of its avatar and forming a rectangle that delimits its collision area. Four points instead of only two are created to prevent its calculation during collision verification.

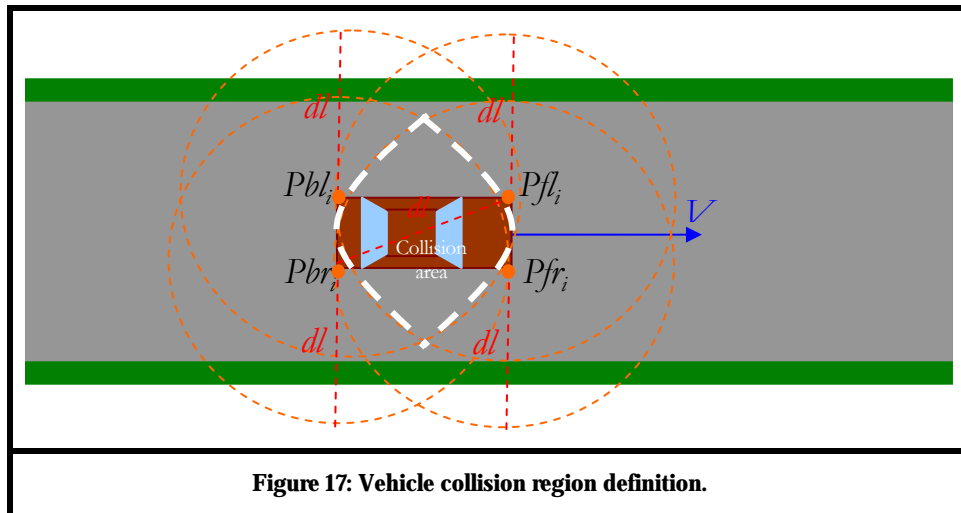
The collision algorithm is only applied for vehicles in the same lane. Moreover, the algorithm uses the number of tickets given to each vehicle during its entrance in a lane to restrict even more the number of vehicles to which it is applied. The collision check mechanism works as follows.

Given two vehicles $Vc1$ and $Vc2$ located in the same lane, with ticket numbers $b1$ and $b2$ respectively, their respective collision points are the front right Pfr_i , front left Pfl_i , back right Pbr_i and back left Pbl_i , for $1 \leq i \leq 2$. And being dl the standard diagonal length of a vehicle:

If $b1$ is smaller than $b2$ and being $Px \in \{Pfr_2, Pfl_2, Pbr_2, Pbl_2\}$ of $Vc2$, then if:

- The distance between Px and Pfl_i is smaller than dl and
- The distance between Px and Pfr_i is smaller than dl and
- The distance between Px and Pbl_i is smaller than dl and
- The distance between Px and Pbr_i is smaller than dl .

Then, $Vc2$ collided with $Vc1$ in the Px point. The collision region defined under these conditions can be seen in Figure 17. It consists of the intersection of the areas of the four circles of ray dl , each one centred in one of the collision points. If a point of collision of another vehicle enters this region, it is assumed that a collision happened.



The previous position of a vehicle is saved at each update. If, during the update of its position, a vehicle collides with another one, the vehicle goes back to the previous position stored in the last update. Thus, the vehicle that collided has its movement interrupted, for it does not move forward, and is simultaneously prohibited from occupying the same space of the vehicle in which it collided.

3.4.4. Calculation of the Separation Force between Vehicles

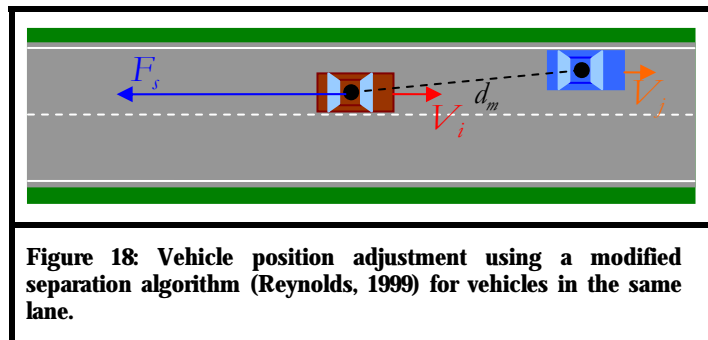
As an attempt to prevent collision, a repulsion force is applied to a vehicle when a minimum distance d_m between it and another vehicle is reached. Being d the current distance between the vehicles and being $D = d_m/d$, for each vehicle position update, if d is smaller than d_m , a repulsive force F_r is added to the total force F applied to the vehicle, where $F_r \sim D$. The minimum distance d_m and the orientation of F may vary between two values, depending on the situation where the vehicles are:

- If vehicles are in the same lane, the distance minimum is:

$$d_m = LVc_1/2 + LVc_2/2 + 1,$$

where LVc_1 and LVc_2 are the lengths of each one of the vehicles.

The orientation of F is opposed to the one of the connection flow, whose orientation vector goes from its initial point to its end point. Figure 18 presents an example of the functioning of this mechanism.



- If vehicles are in adjacent lanes, the minimum distance is calculated and the separation algorithm is only applied if the condition below is satisfied:

Being $DLVc_1$ and $DLVc_2$ the diagonal lengths of the vehicles, the minimum distance from which the separation algorithm must be applied is:

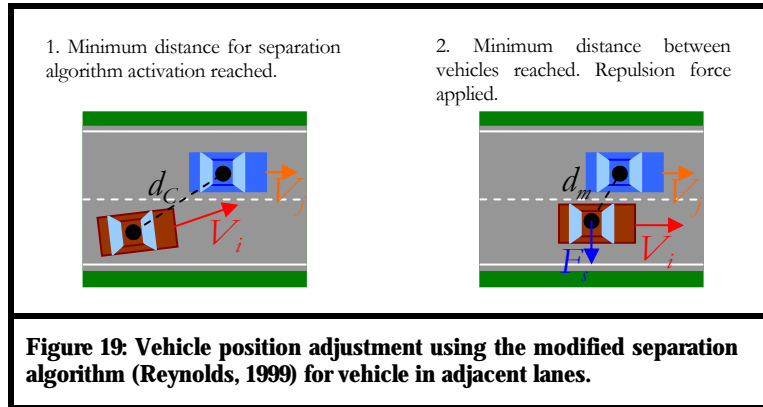
$$d_c = DLVc_1/2 + DLVc_2/2 + 1$$

The separation distance check is then effectuated and the minimum distance d_m is calculated and checked only if d is smaller than d_c . The value of d_m is:

$$d_m = W_{w1}/2 + W_{w2}/2 + 1/8,$$

where W_{v1} and W_{v2} are the widths of each one of the two vehicles.

The orientation of F is the same of the vector that goes from the end point of the lane of the current vehicle to the end point of the lane of the vehicle that entered the repulsion area. Figure 19 presents an example of this separation mechanism.



3.5. Optimizations

In order to improve performance, the simulation was optimized in many of its features. Some of them were implemented due to problems found during the development and realization of the simulation experiments; others were implemented to reduce its computational load. There were also alterations to boost processing for this version with centred architecture.

3.5.1. Relative Time Unit Alteration

The relative simulation time unit had to be changed according to the simulation frame rate, due to the fact that the calculation of positions, speeds and accelerations during vehicles position update was done using the Euler model (Mathworld, 2004). In this model, the values of acceleration and speed are multiplied by a time interval Δt equal to the elapsed time between simulation updates. Considering the frame rate as FR , it is given by $FR = 1/\Delta t$. The advantage of the Euler model is that, even if the frame rate decays, the vehicles will still move in the same speed, synchronizing simulation with the real time, independently of the simulator frame rates.

However, this model presents some problems for transit simulation. When the update rates of the model are very low, Δt increases its value and, consequently vehicle position leaps become larger. The problem is that vehicle curves become more open during the application of the path following algorithm (Reynolds, 1999). As this algorithm requires the vehicle to approach a minimum distance from its target point so that it can chase another point, when Δt is very large,

reaching certain target points simply does not become possible, mainly when the angle between adjacent connections in the vehicle path is inferior to 90° .

As a consequence, vehicles walk in circles around their destination point, without ever reaching them. Moreover, the vehicles have a greater difficulty in following lanes. In summary, the Euler model does not function well for the simulation when the frame rates are low.

To cope with this problem without the implementation of other methods, such as the fourth order Runge-Kutta (Mathworld, 2004), it was decided to modify the time interval Δt passed for the simulation between an update and another, depending on its current value. If $\Delta t \geq 0.04$, this means $FR \geq 25$ fps, and so the elapsed time value passed to the simulation is Δt itself and the model is stable. However, if $\Delta t > 0.04$, the elapsed time value passed to the simulation will be always 0.04 instead of the value of Δt , so that, for $FR < 25$, the model will tend to diminish its simulation speed in terms of vehicle displacement and traffic lights states.

By doing this, the simulation time will not always be equal to real time. Despite this disadvantage, the model becomes stable and works well independently of the frame rate with which the simulation is running.

3.5.2. Graphical Scene

In order to reduce rendering complexity, no illumination was used. As no light was added to the scene, the default OpenGL illumination was applied. All the objects appearing in the simulation had textures applied in their surfaces to guarantee colouring. The final appearance of the graphical scene can be seen in section 4.2.

3.5.3. LOD

In order to increase performance, levels of detail were not only applied to the terrain, as seen in section 2.2, but also to vehicles. LOD generation was done as follows.







First, a reasonably detailed model of a domestic vehicle was developed. This model, representing a vehicle with headlights, wheels, hubcaps, plates and windshields, was used as a representation of the higher detail level of the vehicle. Other models for LOD representation were created by polygon and vertices reduction of this first model until a model with minimum level of detail was reached. This minimum model represented the vehicle by only three plans parallel to each other much as the three coordinate plans. During this process, six distinct levels of detail were

created. For the correct reduction of the details without perceptible deformations between models, the modelling process involved a high degree of subjectivity and creativity. Although the six levels seemed to be enough to represent the vehicle in diverse visualization situations, other extra intermediate LOD could have been created.

Similarly to terrain avatars, textures with different colours were applied to each vehicle avatar LOD. This was useful for noticing the graphical visualization change between different vehicle avatar LODs during tests, since the mere change of the three-dimensional objects representing vehicle was not perceptible enough. And this partially confirmed the effectiveness of the technique levels of detail.

The models used in the vehicle avatar were not rigorously optimized with respect to their number of polygons. They have only been used to test the application performance in situations with different numbers of vehicles and levels of detail. The number of faces and vertices, the activation distance and the textures for each LOD can be seen in Table 10. The textures were obtained from reference (Molofee, 2004-2). The vehicle LOD functioning is presented in figures 24 and 25.

Table 10: Different vehicle LODs information.

<i>Level of detail</i>	LOD 0	LOD 1	LOD 2	LOD 3	LOD 4	LOD 5
<i>Number of vertices</i>	793	328	51	33	17	25
<i>Number of faces</i>	717	552	84	51	30	9
<i>Activation distance</i>	0	10	100	200	300	600
<i>Texture</i>						

3.5.4. Data Structure Optimization

Due to the fact that the current model is centred, two optimizations in the data structure were implemented with the purpose of increasing system performance. Although these optimizations would not be feasible in a distributed system, they were considered here. They would have to be re-evaluated if the simulation is to be distributed.

The first optimization consisted in centralizing the vehicles list to be used in the simulation. Instead of each junction containing entrance connections possess its own list with a percentage of the maximum number of vehicles allowed for insertion in the simulation, only one list with all the vehicles was created in the Simulation class. This measure not only diminished the amount of

variables in each junction, but it also helped to more efficiently control the total number of vehicles simultaneously allowed in the simulation.

The same technique was applied to the traffic graph, which should have been copied to each junction. However, a single version was also left in the Simulation class and shared by all junctions.

3.5.5. Algorithmic Improvements

Algorithms that guarantee the vehicle behaviours, as path following and collision, were optimized to become less onerous.

The path following algorithm calculates if a vehicle is leaving or not its trajectory, based not on the distance d_c between its next position and the centre of the lane in which it is located, but on the distance between the desired speed and the current speed, whose values are faster to calculate than d_c .

The collision algorithm was significantly simplified, being applied only for vehicles in the same lane which have ticket numbers of lane inferior to the vehicle's to which collision is being calculated. Moreover, the algorithm which detects the intersection between vehicles collision points does not use distance calculation between points and straight lines. Thus, the definition of vehicles collision areas is less complex, though more inexact, as it was seen in section 3.4.3.

Simplifications in the separation algorithm have also contributed for performance improvement. However, this algorithm was simplified not by performance improvement but by stabilization needs in terms of behaviour inside the lane and value variation according to distance between vehicles.

Chapter 4 :

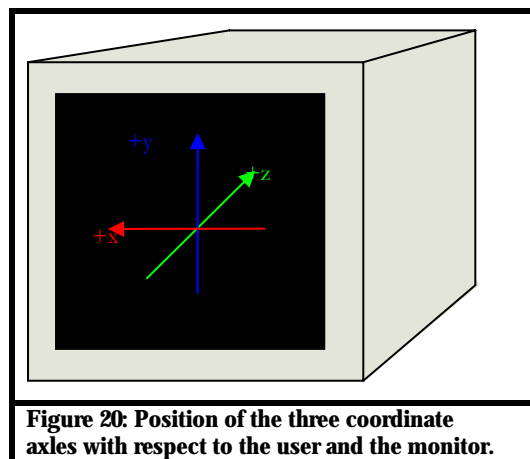
Interface

This chapter encloses the third and last great part composing the application, which is the interface. It was created using NeHe graphical library (Molofee, the 2004) which allows text impression in the OpenGL scene, guaranteeing information presentation on the simulation. This interface provides visualization of both the three-dimensional environment and simulation information.

Orientation axes were added to guide the user. The interface was also configured to always follow user movements, placing itself in his front. By the use of keyboard commands, the user can, besides navigate, save viewpoints for posterior visualization.

4.1. Axles System

Before understanding how user movements work, the explanation of to which direction coordinate axes point is needed. Figure 20 clarifies this concisely. This is the standard axes system of the graphical library used (Molofee, 2004).



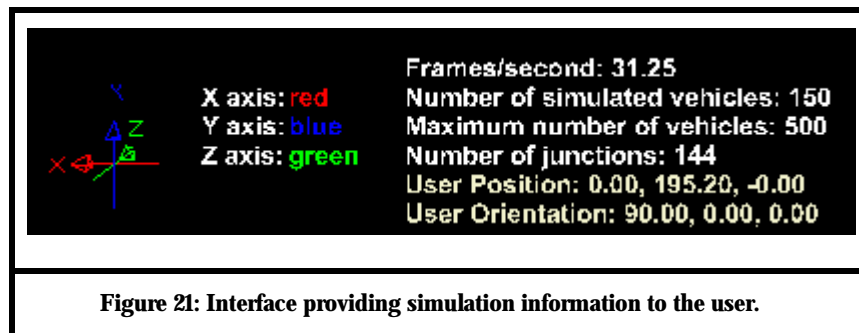
4.2. Interface Components

The user interface is sufficiently simplified. It is composed by a three-dimensional interface presenting the world and an input interface by keyboard commands.

The interface introduces the user in the world by a first person point of view. Thus, the user avatar does not possess any visible graphical representation.

Real-time update information is presented in the bottom region of the simulation screen. This information consists of: user position and orientation, simulation frame rate, total number of junctions and maximum number of vehicles simultaneously allowed in the simulation.

A small three-dimensional axes system, similar to the one shown in Figure 20, is also present to indicate positive and negative directions of the three coordinate axes. These axes serve in user guidance, indicating the correct direction of the axes independently of the direction to which the user is pointing. The interface is illustrated in Figure 21.



The input interface consists of a set of navigation and viewpoint creation, selection and removal keyboard commands. The list of all these commands may be seen in Appendix 1 - A.2.

The simulator interface may be seen in use below. Figure 22 and Figure 23 present user views for a manually generated traffic graph, where the higher levels of detail of the vehicle avatar are perceptible, as well as traffic light functioning. Figure 24 and Figure 25 present the user views for an automatically generated traffic graph. For these two, the variation of levels of detail in the terrain and in vehicles is perceptible.

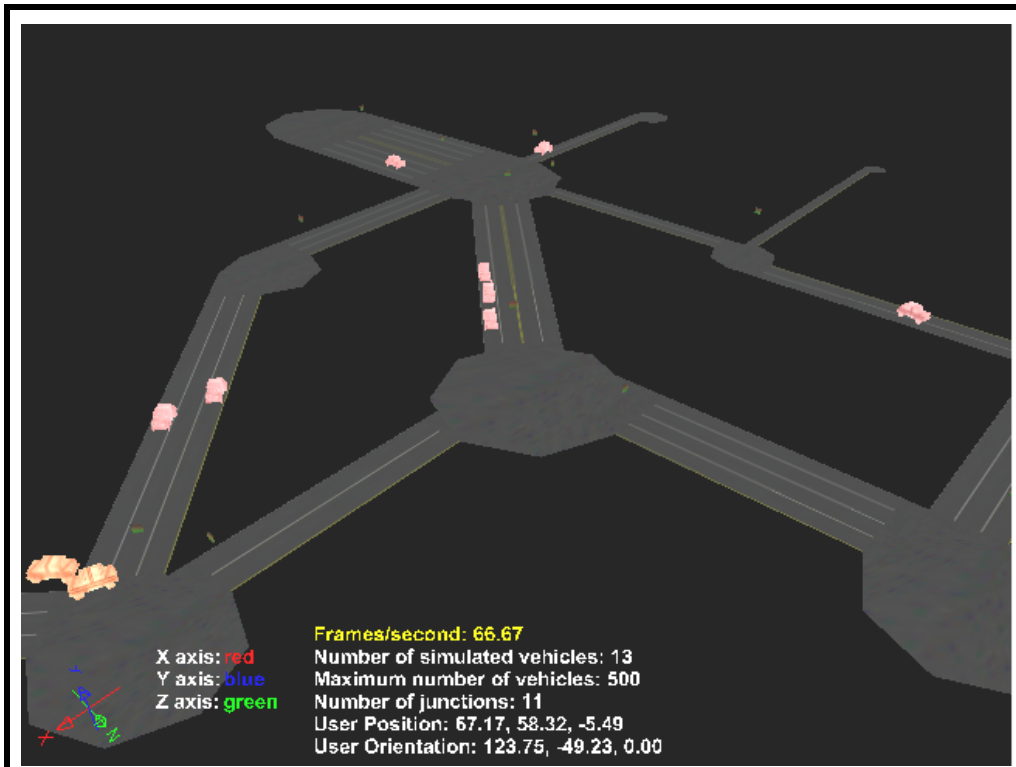


Figure 22: Traffic ways visualization for a manually generated traffic graph.



Figure 23: Visualization of semaphore functioning and higher LODs of vehicle avatars for a manually generated traffic graph.



Figure 24: Visualization of terrain and vehicle LODs for an automatically generated traffic graph.

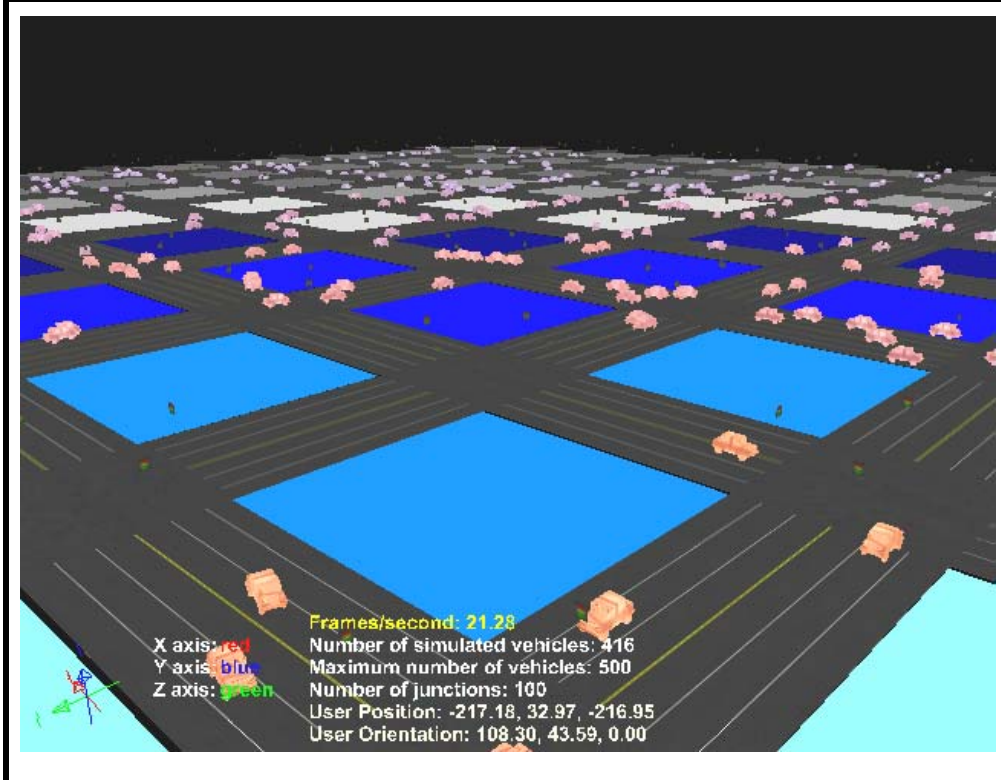
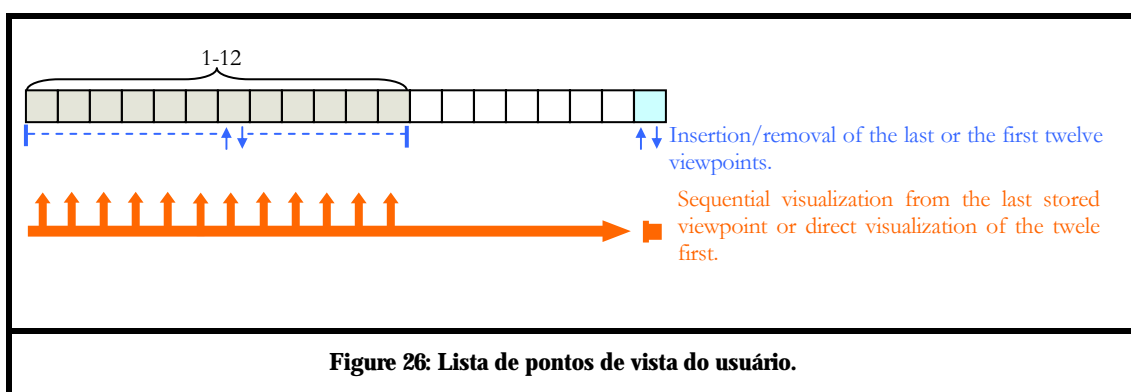


Figure 25: Visualization of terrain and vehicle LODs for an automatically generated traffic graph.

4.3. Viewpoints

The user has the power to store viewpoints for future consultation during traffic analysis. These viewpoints consist of specific orientations and positions of the user avatar and are stored in a list inside the user avatar. Every time the user activates the viewpoint saving mechanism, its orientation and position are stored in the end of the viewpoint list.

Besides the sequential navigation of the stored viewpoints, the user is able to remove or view the last visited viewpoint. He can directly remove or view the twelve first stored viewpoints in the list by the use of hot keys. This list of these commands are illustrated in Figure 26.

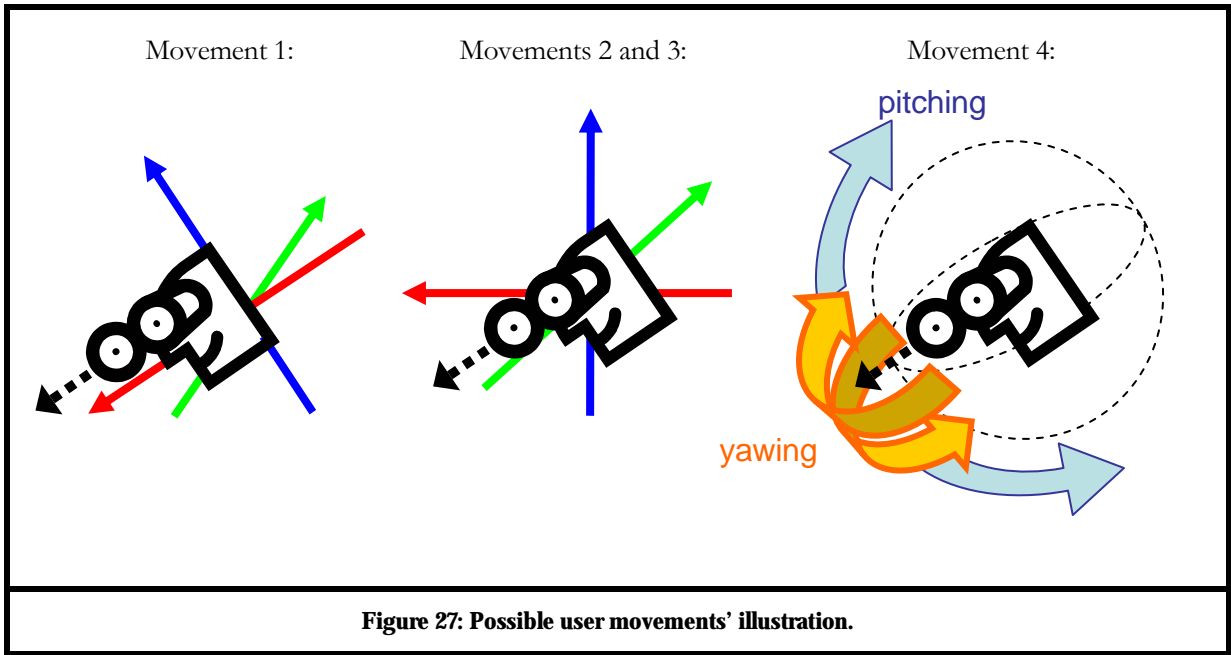


4.4. Simulation Control Commands

Besides the viewpoints list control commands, the avatar movement control commands also exist in the simulation. These movements are divided in four groups, presented below:

1. Translation in the direction s to where his avatar points, in the opposite direction of s and in any direction of the plan whose normal vector is s .
2. Translation in a parallel straight line to y axle. It is equivalent to the modification of the user height relative to the roads surface;
3. Translation in the direction s to where its avatar points, but remaining in a plane parallel to the XZ plane. It consists of a translation parallel to the surface of the tracks.
4. Rotation around its relative x axle, called pitching, or around its relative z axle, called yawing.

The illustration of these four types of movement can be seen in the Figure 27.



Chapter 5 :

Performance Analysis

5.1. Experiments

In order to test the influence in efficiency of diverse simulation aspects and the performance of the implemented algorithms, some experiments were carried. These experiments aim to prove the increase in performance due to the application of LODs to the vehicles, besides testing performance the degradation with the variation of the number of junctions and the shape of the traffic graph.

The machine where the experiments were carried was a PC with a 2.2 GHz processor, 1GB of RAM memory, a GeForce FX[®] 5200 128MB graphical board and with Windows XP[®] operational system. The performance measurement parameters consisted of memory consumption and the average frame rate per seconds of the simulation.

Only one variable of the simulation had its value changed at a time in each experiment, in order to guarantee the cause→effect relation between the value of the variable being analyzed and the application performance. All the carried experiments used automatically generated traffic graphs and terrains. This helped to maintain data consistency and to develop a diverse battery of tests faster.

Before initiating the average frame rate measuring process in the simulation, an interval of approximately 5 minutes was given. This was done so that the measure of the frame rate values could be obtained when the simulation model was in a stable situation. This situation occurred when all traffic lights had already entered its cycle of normal functioning, the flow of vehicles was already uniformly distributed in the entire rectangular traffic mesh and the entrance and exit flow of vehicles has practically become constant. In some experiments, this initial wait interval had to be 10 minutes. For all experiments, the maximum number of vehicles in the simulation was set to five hundred.

After the five minutes interval, five more minutes were dedicated to measure the average frame rate. The variation of this time between 5 or more minutes is consequence of the frame rate sampling to have been done at each 0,5 s passed in the simulation time, not in the real elapsed time.

Thus, the number of samples is a total of 600, but it could have taken more than five minutes to obtain them, if the frame rates were below 25 fps. The frame rate for each test of the experiments was, then, calculated from the average of these 600 measures.

To facilitate experiments reproduction, a table with the configurations used for the traffic graph is presented in the start of each experiment description. Their specifications are presented in sections 5.1.1 and 5.1.2. Their results and analysis are described in section 5.2.

5.1.1. Vehicle LOD Performance Experiment

The first experiment evaluates the performance in the use of levels of detail in vehicle avatars. The evaluation parameter for this experiment was only the simulation frame rate. The graph configurations of the experiment are presented in Table 11.

Table 11: Automatically generated graph configuration for the vehicle LOD performance test experiment.

N	M	d	f	n_l	w
3	4	200	60	5	3

The fields of Table 8 are the following: N – number of lines containing nodes in the traffic graph, M - number of nodes per line in the traffic graph, d - distance between nodes in the traffic graph, f - flow of vehicles per minute for all edges, n_l - number of lanes for all edges and w - width of the lanes in meters for all edges.

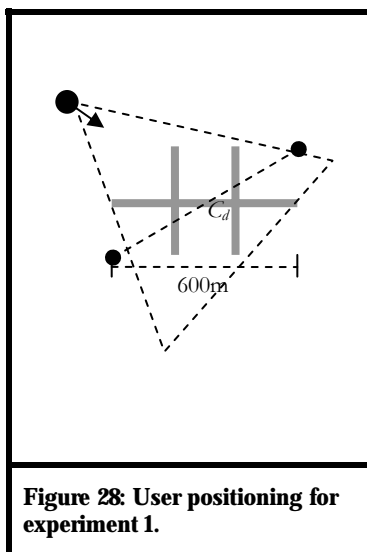


Figure 28: User positioning for experiment 1.

Initially, six tests were carried. In each of them, vehicle avatars had a single different level of detail. Thus, in each test vehicles were represented by only one of the six available models of vehicles. In all these tests the user was located away from the graph so that he could visualize its entirety, as shown in Figure 28. The frame rate was measured for each of the tests.

After that, a seventh test was carried, where vehicle avatars could assume all the six LODs according to distance from user. Due to this real-time vehicle LOD change, it was perceived during the experiments that, depending on the position of the user, the simulation could present frame rates with values quite varied. To calculate the average value of

these rates, three possible basic user positioning situations were identified, during its navigation, where vehicles would affect frame rate differently due to their different distances and LODs. This seventh test consisted of a set of tests, one for each of these situations. Its frame rate resulted from the average of the three test frame rates. The three detected situations are described below and are seen in Figure 29, together with the traffic graph shape chosen for the experiment.

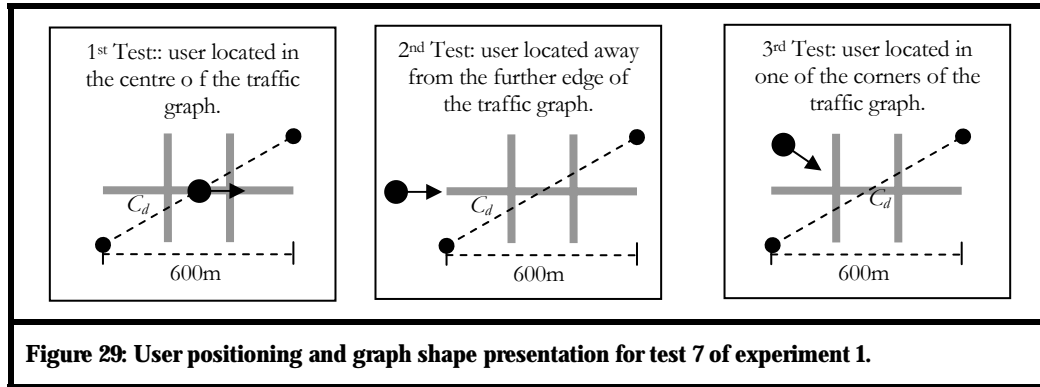


Figure 29: User positioning and graph shape presentation for test 7 of experiment 1.

In the first one, the user is placed in the centre of the traffic graph to a height of 1,5 m from the road surface. In this situation, the levels of detail of the vehicle avatars vary concentrically around the user. The levels with smaller distance of activation are the ones activated in this position configuration.

In the second test, the user is moved to one of the further entrance connections of the traffic grid to a height of 1,5 m from the road surface. In this case, levels of detail of vehicle avatars also vary concentrically to the user. However, in this position it is possible to view vehicles with avatars in all possible levels of detail. This is because the activation distance for the last level of detail of the vehicle is equal to 600m, which is also equal to length L of the graph. The calculation of L is shown below:

$$L = d \times (M-1) = 200 \times 3 = 600m.$$

The value $M-1$ represents the number of edges between m junctions. Multiplying it by the distance d between adjacent junctions results in the total length of one of the graph rows. As the number of columns M is greater than the line number N in the specified graph, this is the greater distance between parallel entrance connections in the graph. Being the user moved away less than 10m from the extremity, which is the maximum distance given for the presentation of the higher level of detail, as seen in the section 3.5.3, all the LODs will have the chance to appear in the simulation.

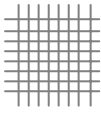
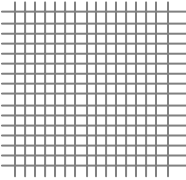
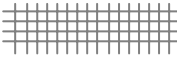
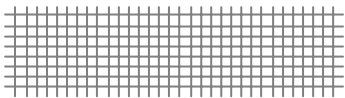

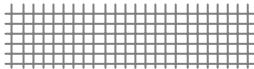
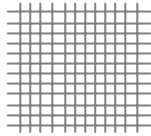
The third scene locates the user in one of the corners of the rectangular graph and at a height of 50 m from the road surface. If $L = 600m$, then the diagonal length of the traffic graph, C_{cb} is greater than 600m and this guarantees the view of the levels of less detail of the vehicles avatar. However, as the user is moved away from any lane, this situation benefits levels of detail not only with lower detail, but also with further activation distances.

5.1.2. Performance Experiment according to the Number of Junction

This second experiment tries to analyze the influence of the amount and positioning of the junctions on simulation performance. The objective is to test the performance of the vehicle path creation algorithm and the performance with regard to the number of entrance connections. The evaluation parameters of this experiment were the simulation frame rate and simulation consumption.

To carry this experiment, seven tests, each one with a traffic graph with different shape and number of junctions was realized. Three of them were quadrangular while the others four were rectangular. In order distinguish them during the analysis an identification number was given to each one of them. Moreover, due to the great amount of junctions, an initial pre-measuring wait interval of 10 minutes was given to graphs 1 and 3. The configurations for each of these graphs are shown in Table 12.

Table 12: Automatically generated traffic graph configurations for performance test according to the number of junctions.

<i>Id</i>	<i>N</i>	<i>M</i>	<i>Graph Representation</i>	<i>Wait time</i>	<i>d</i>	<i>F</i>	<i>n_l</i>	<i>w</i>
0	8	8		5	100	5	3	3
1	16	16		10	100	5	3	3
2	4	16		5	100	5	3	3
3	8	32		10	100	5	3	3
4	3	48		5	100	5	3	3
5	6	24		5	100	5	3	3
6	12	12		5	100	5	3	3

The fields of Table 12 are the following: *Id* - identification number of the traffic graph, *N* - number of lines containing nodes in the traffic graph, *M* - number of nodes per line in the traffic graph, *Graph representation* - spatial configuration of the traffic mesh that represents the graph, *Wait time* - initial wait time previous to the frame rate measurement, *d* - distance between nodes in the traffic graph, *f* - flow of vehicles per minute for all edges, *n_l* - number of lanes for all edges and *w* - width of the lanes in meters for all edges.

Vehicles avatars have only a single level of detail, the LOD3, whose polygon configurations are shown in Table 10 of section 3.5.3. This alteration was done to guarantee that the simulation was executed with reasonable efficiency for different complexity levels. A lower level of detail was not chosen to guarantee that variations in the amount of vehicles would impact in the frame rate measures in each test. During these tests, the user was located close to the centre of the traffic graph. No other variable had its value changed as it is seen in Table 12.

5.2. Results and Analysis

Below, the results of the two tests considered previously are presented. An analysis of the results is also presented.

5.2.1. Vehicle LOD Performance Experiment

In Table 13 and Figure 30, the frame rates are presented for the seven tests of the first experiment.

Table 13: Frame rates results for each test in experiment 1.

<i>Test number</i>	<i>Vehicle model level of detail</i>	<i>Average frame rate</i>	<i>Standard deviation</i>
0	LOD 0	1.84	0.14
1	LOD 1	4.41	0.37
2	LOD 2	2.96	3.18
3	LOD 3	29.39	5.13
4	LOD 4	40.70	14.35
5	LOD 5	51.55	15.81
6	All LODs simultaneously	$(12.00+23.42+35.46)/3 = 23.63$	$(1.62+5.73+10.89)/3 = 6.08$

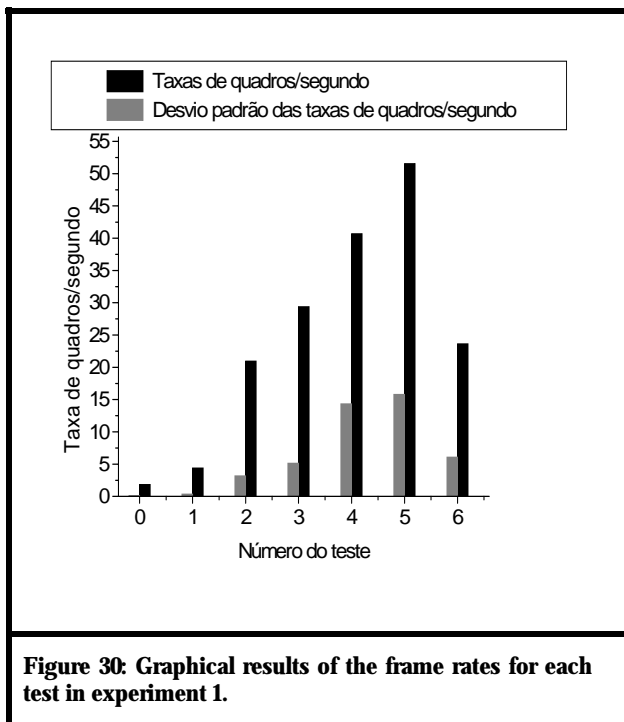


Figure 30: Graphical results of the frame rates for each test in experiment 1.

The reduction of frame rates in the simulation is perceptible when levels of higher detail are applied. It is also noticeable that these rates had a smaller increase as levels of detail decayed, considering the increase of the standard deviation for each of the average frame rates.

The levels of detail technique to the simulation revealed to be efficient. Besides model quality change being practically imperceptible, the performance of the simulation with all vehicle LODs was close to the average with respect to the values measured

for the other tests, where only one level will detail per time was used. The high polygon variation between models with more detail pushed this result below the average, as can be seen in Figure 30.

From Figure 31 to Figure 36, graphics presenting the frame rates measured and the average value for the first six tests are presented, in which only one level of detail was considered at a time.

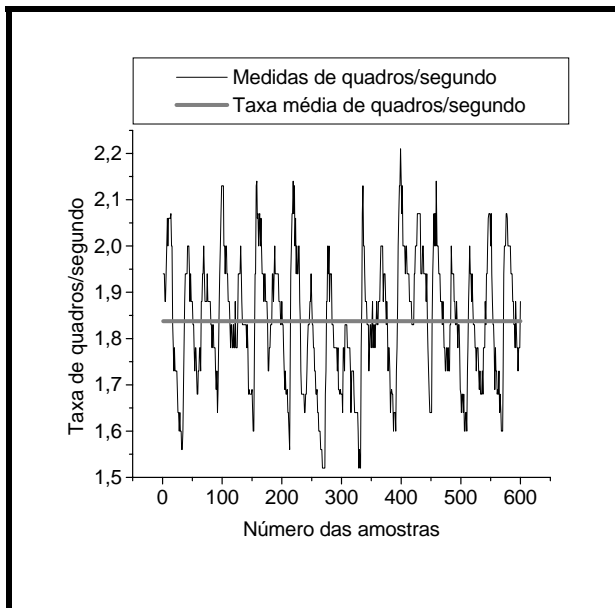


Figure 31: Frame rates for experiment 1 with LOD 0 applied to vehicle avatars.

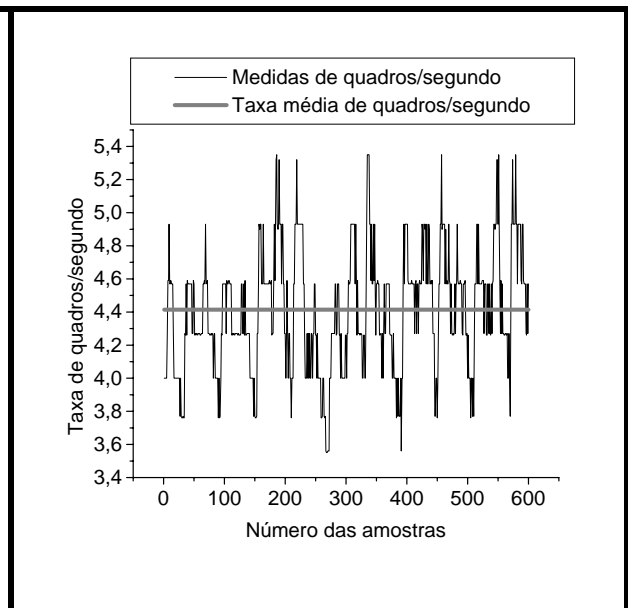


Figure 32: Frame rates for experiment 1 with LOD 1 applied to vehicle avatars.

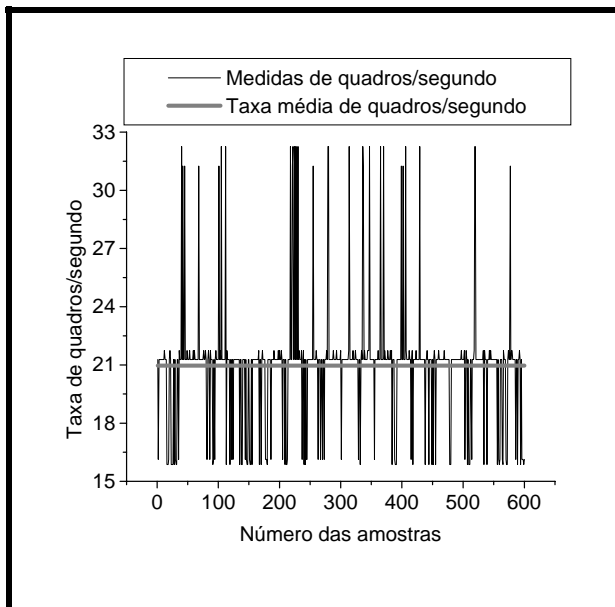


Figure 33: Frame rates for experiment 1 with LOD 2 applied to vehicle avatars.

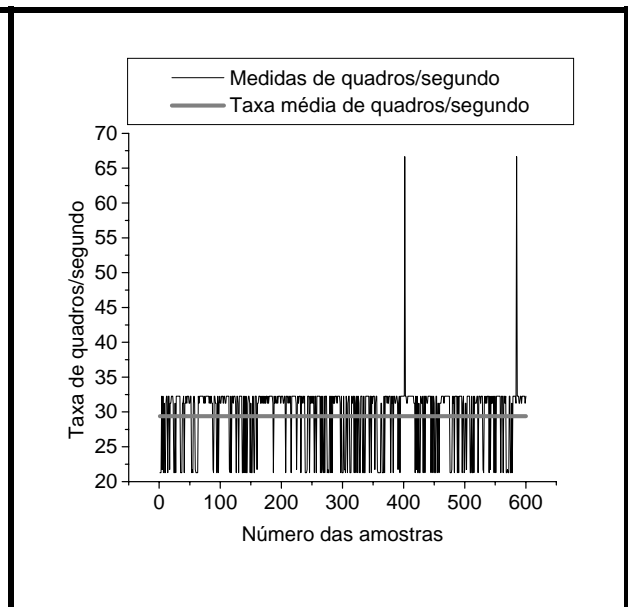


Figure 34: Frame rates for experiment 1 with LOD 3 applied to vehicle avatars.

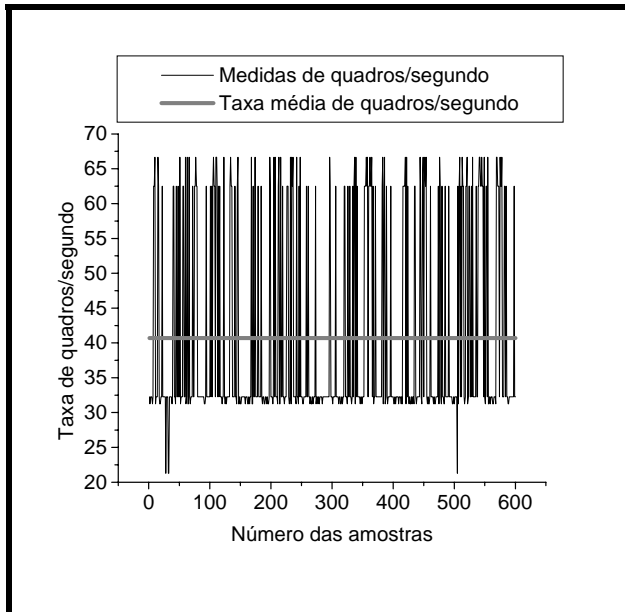


Figure 35: Frame rates for experiment 1 with LOD 4 applied to vehicle avatars.

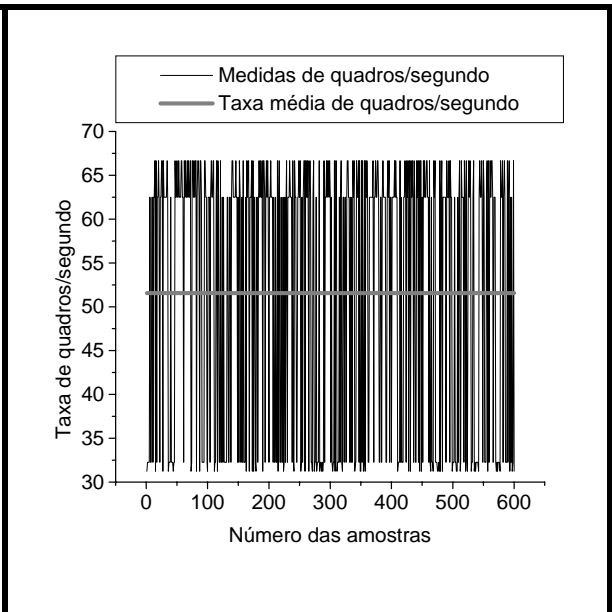


Figure 36: Frame rates for experiment 1 with LOD 5 applied to vehicle avatars.

From Figure 37 to Figure 39, graphics present the average frame rate measured for the seventh test with the user in the three different positions. Figure 40 presents the average between the samples in these three tests.

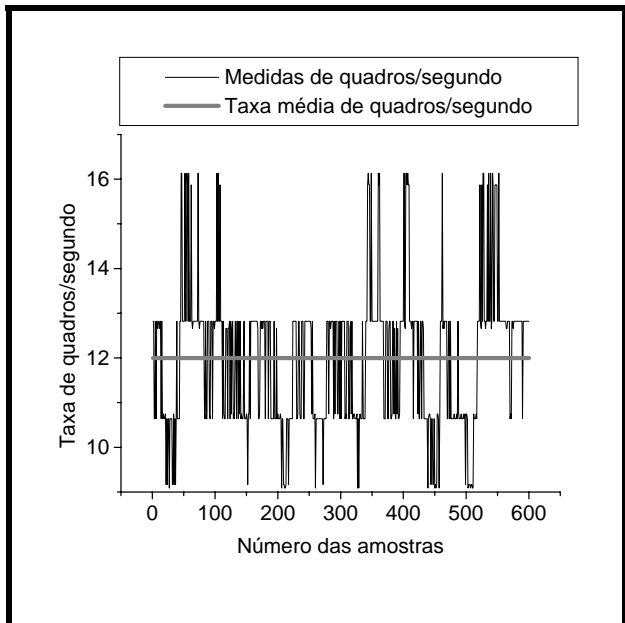


Figure 37: Frame rates for experiment 1 with all LODs applied to vehicles and the user located in the centre of the graph.

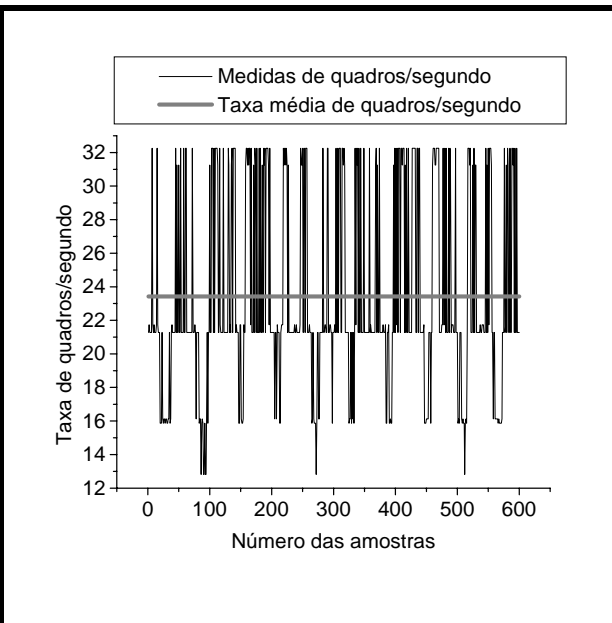


Figure 38: Frame rates for experiment 1 with all LODs applied to vehicles and the user located in the further extremity of the graph.

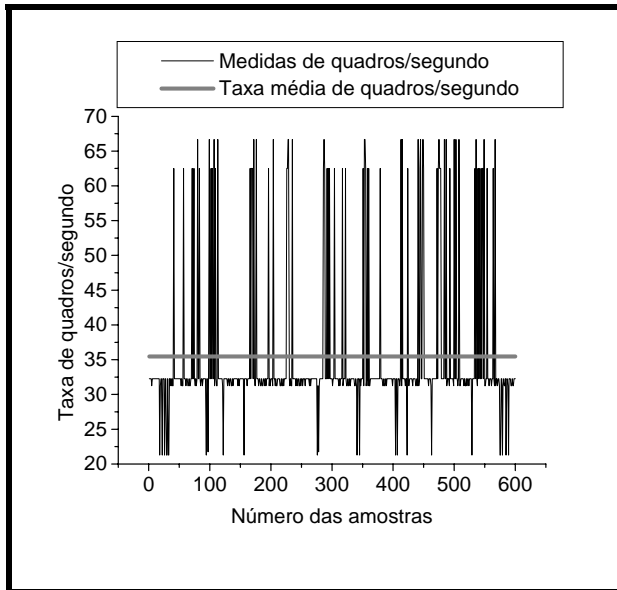


Figure 39: Frame rates for experiment 1 with all LODs applied to vehicles and the user located in one of the corners bounding rectangle of the graph.

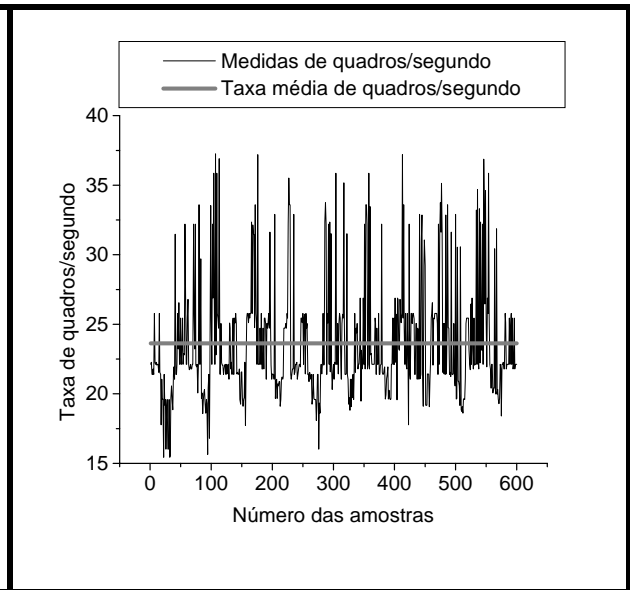


Figure 40: Average frame rates based on the samples of the three measurements of the seventh test of experiment 1.

Figure 41 plots a histogram defined with the data of experiment 1 for the test with vehicles being only represented by LOD3, which was previously presented in Figure 34. The reason for choosing this specific test for histogram plotting was because this test had the least amount of different values amongst all the tests carried for experiment 1.

It is apparent that data follows a multimodal distribution, with the existence of separate peaks. The higher of these peaks is the one located between the values of 30 and 35 fps, indicating that the majority of the sampled values are found within this interval. The possible reason for such a distribution is the little randomness of the entrance times for the vehicles in the simulation, since this lack of

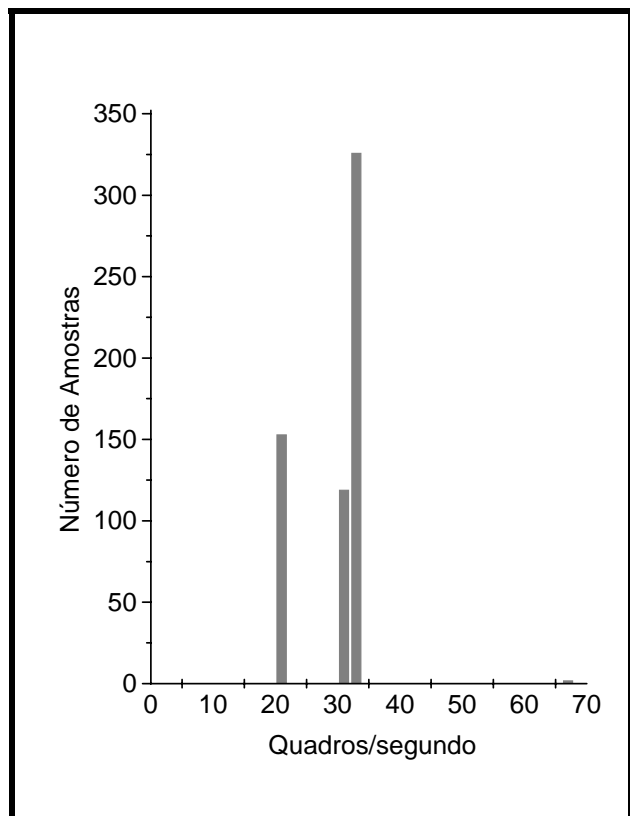


Figure 41: Histogram with data obtained from experiment 1 and vehicles represented only by LOD 3.

variation of values did not appear so evidently in the results of other tests. However, no experimental evidence was yet discovered to prove this affirmation.

5.2.2. Performance Experiment according to the Number of Junctions

In Table 14 and Figure 42, the frame rates per second and the levels of memory consumption for the seven graphs of experiment 2 are presented.

Table 14: Frame rates and levels of memory consumption for each tested graph in experiment 2.

<i>Graph Id</i>	<i>Frame rates per second</i>	<i>Standard deviation</i>	<i>Memory consumption (MB)</i>
0	54.98	14.50	45.552
1	15.03	1.49	195.908
2	61.22	9.28	40.344
3	15.24	1.37	185.528
4	20.97	3.11	71.496
5	18.26	2.65	98.692
6	17.38	2.37	106.472

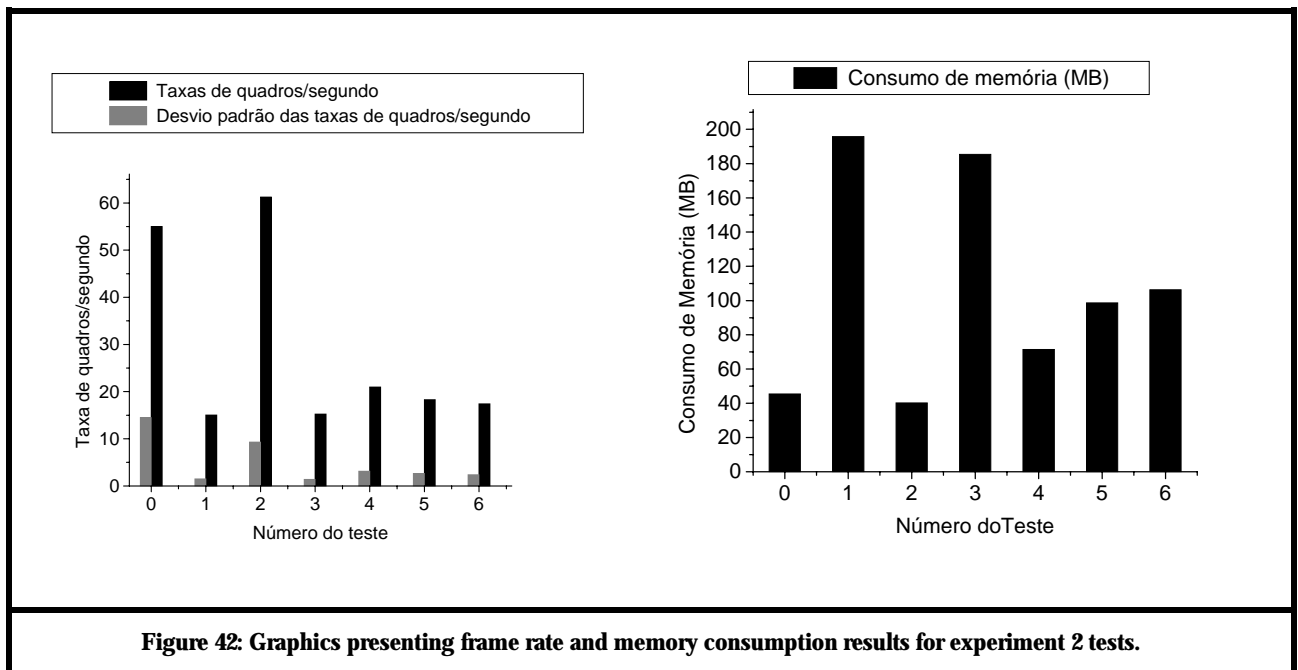


Figure 42: Graphics presenting frame rate and memory consumption results for experiment 2 tests.

From Figure 43 to Figure 50, graphics presenting frame rates samples and their average for the seven tests of experiment 2 are shown, for each of which different traffic graphs were considered, as previously explained in section 5.1.2.

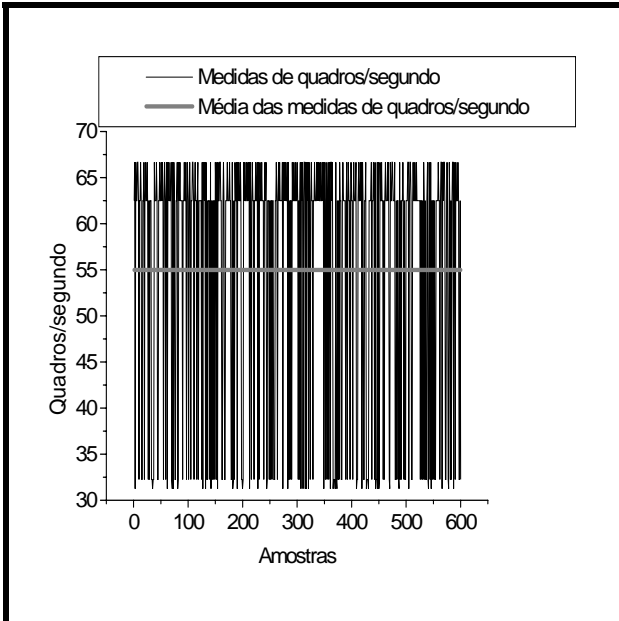


Figure 43: Frames rates of experiment 2 using graph 0.

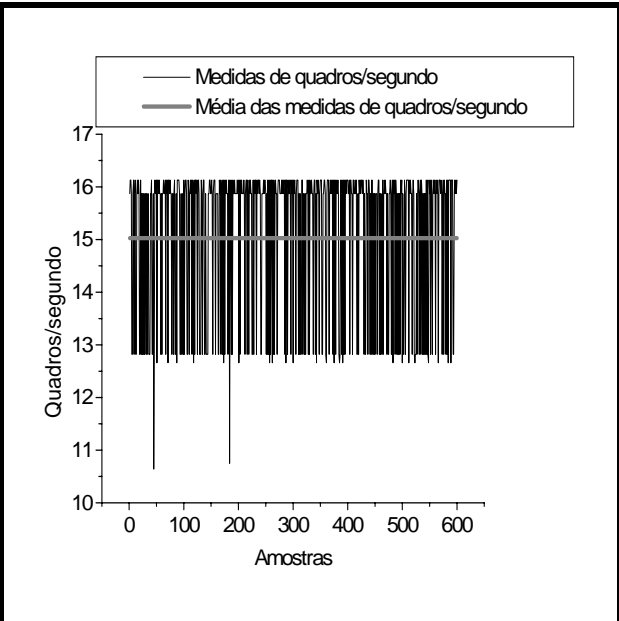


Figure 44: Frames rates of experiment 2 using graph 1.

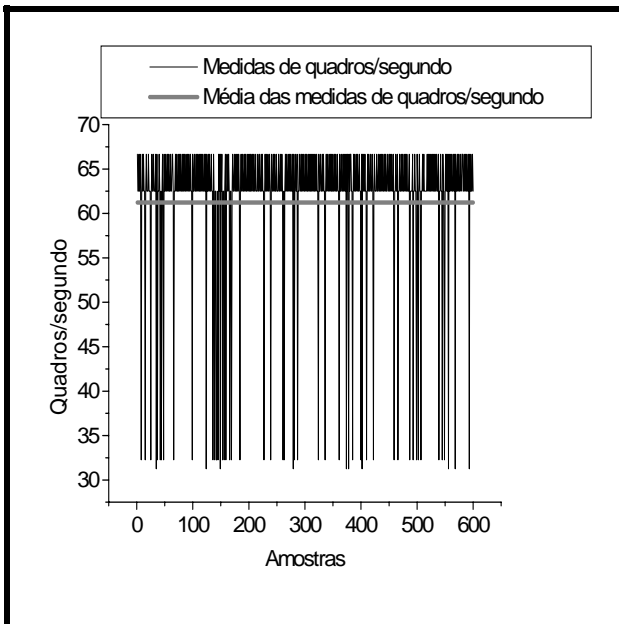


Figure 45: Frames rates of experiment 2 using graph 2.

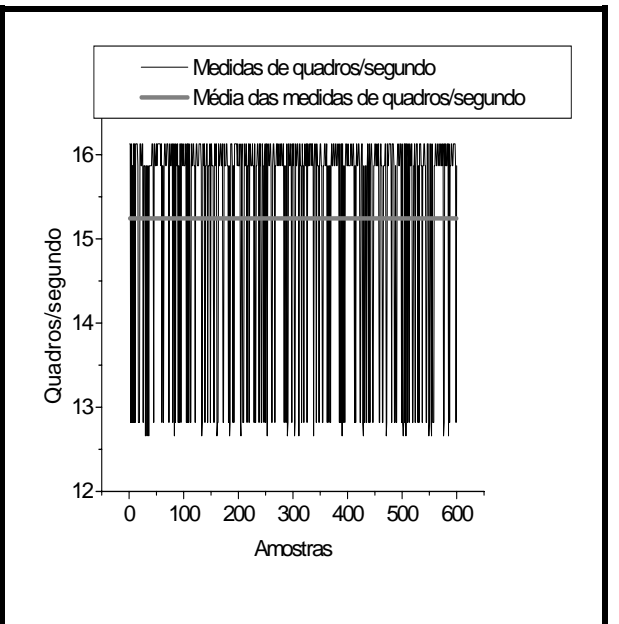


Figure 46: Frames rates of experiment 2 using graph 3.

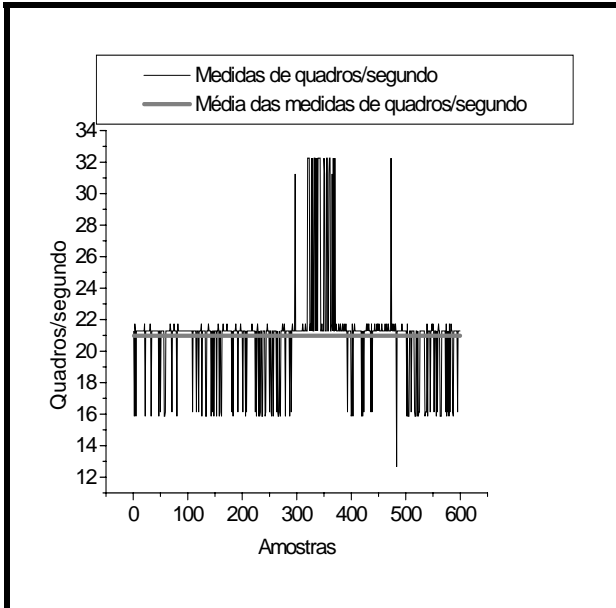


Figure 47: Frames rates of experiment 2 using graph 4.

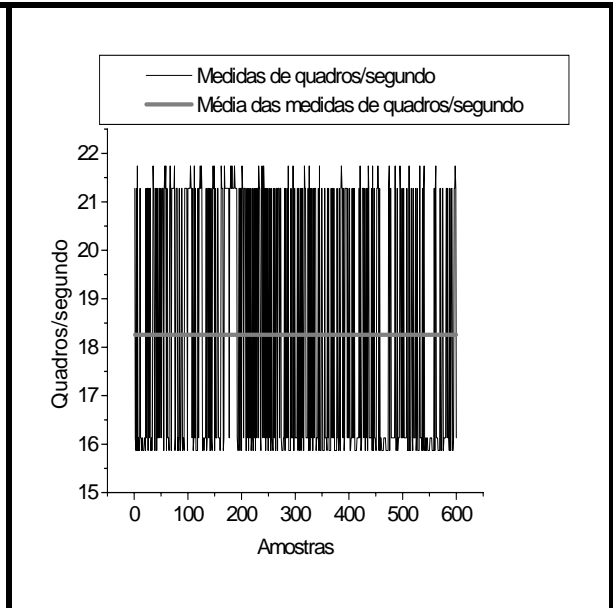


Figure 48: Frames rates of experiment 2 using graph 5.

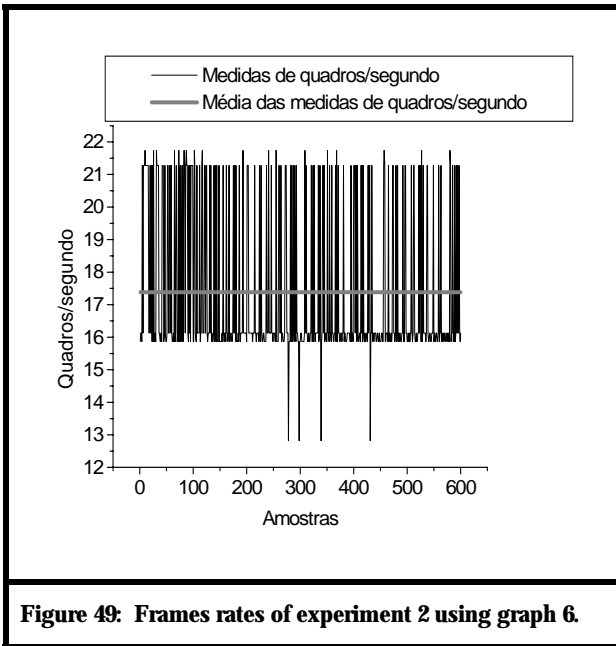


Figure 49: Frames rates of experiment 2 using graph 6.

When comparing the graph pairs: 0 and 2, 1 and 3 – where both the graphs within each pair have same number of junctions - little variation between their frame rates is noticed. This indicates that the graph format used in the simulation has little influence in its performance. This result can also be concluded from the analysis of graphs 4, 5 and 6, which also have the same number of junctions and whose frame rates are practically the same.

Nevertheless, a growth in memory consumption can be seen between the pairs of graphs 4 and 5, 5 and 6, 2 and 0, 3 and 1, where, for each pair, the first graph consumes less memory than he second. This is justified by the fact that the first graph of each pair has less internal edges than the second, as seen in Table 15. Thus, the number of connections between the junctions is smaller in the first than in the second graph and, as a consequence, the simulation of the first consumes less memory than of the second.

Table 15: Number of entrance connection for each graph used in experiment 2.

<i>Graph</i>	<i>Dimension in number of junctions</i>	<i>Number of entrance connections</i>
0	8×8	$2 \times (8-2) + 2 \times (8-2) = 6+6 = \mathbf{24}$

1	16×16	$2 \times (16-2) + 2 \times (16-2) = 28+28 = \mathbf{56}$
2	4×16	$2 \times (4-2) + 2 \times (16-2) = 4+28 = \mathbf{32}$
3	8×32	$2 \times (8-2) + 2 \times (32-2) = 12+60 = \mathbf{72}$
4	3×48	$2 \times (3-2) + 2 \times (48-2) = 2+92 = \mathbf{94}$
5	6×24	$2 \times (6-2) + 2 \times (24-2) = 8+44 = \mathbf{52}$
6	12×12	$2 \times (12-2) + 2 \times (12-2) = 20+20 = \mathbf{40}$

However, still considering these graph pairs, all the first graphs of each pair have more entrance edges than the second. It might be expected that the higher the number of entrance edges the higher the volume of vehicles passing and staying in the model, which would consequently cause a processing increase and a frame rate reduction. This was not perceived in these simulation results, though. The justification for this is the fact that, in rectangular graphs, the probability of a vehicle to cross the graph from one side to another is greater than in quadrangular graphs, which leads to a reduction of the average size of the vehicles paths. Therefore, the increase of flow in rectangular maps, with more entrance connections, is compensated by the reduction of the average path traversal times of vehicles in the traffic graph, not considerably affecting the simulation frame rate.

Figure 50 presents a histogram with the data of the experiment 2 with graph 4, whose data was previously presented in Figure 47. This test was chosen because its graph has the highest amount of different values among all the tests of experiment 2.

It is observed that data also follow a multimodal distribution, with the existence of some separate peaks. The higher of these is located between the 22 and 20 frame rate values, indicating that the majority of the values are found in this interval. The theory that justifies little entrance time randomness for vehicles in the simulation explained in experiment 1 may also be applied here.

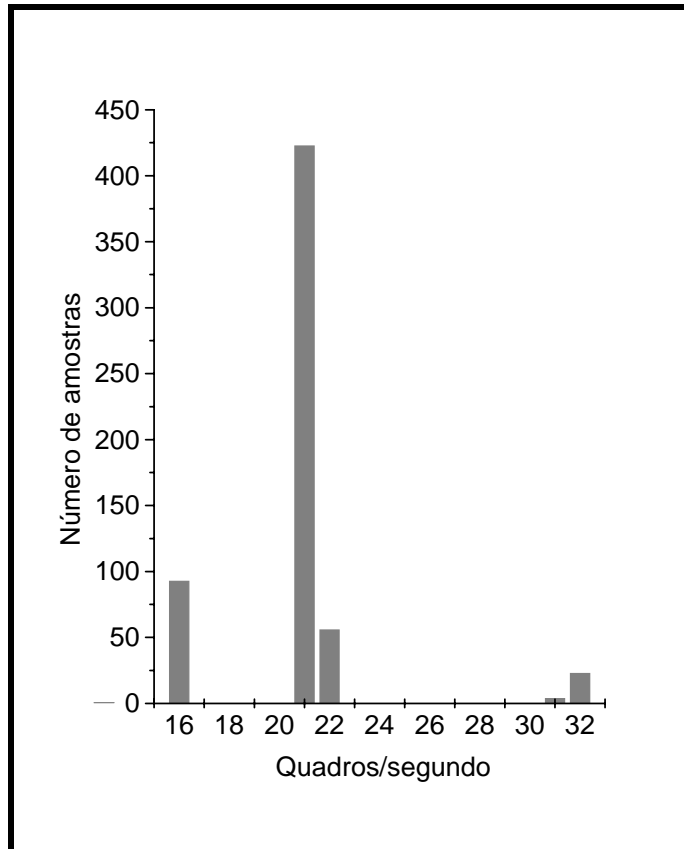


Figure 50: Histogram with test data of experiment 2 with graph 4.

Chapter 6 :

Conclusion

Based on the experiments carried and the analysis of their results, it is evident that efficiency is attainable when simulating urban traffic in three-dimensional environments. The presence of a third dimension, although its effectiveness has only been partially justified in this initial version of the research work, immensely contributes to applications of this kind. It simultaneously allows more information overlapping, more efficient navigation, more realistic environment creation, besides more precise analysis based on data of difficult visualization in bi-dimensional interfaces such as terrain and height models.

Although 2D interfaces take care of the larger amount of transit simulators data control, navigation and manipulation necessities, they are not complete. The visualization and navigation between distant points presented in perspective are only possible with the use of a three-dimensional interface. Only with this third dimension is that the full interaction implementation between terrain, underground and road construction information becomes possible. This interaction allows cost estimation based on the construction type according to terrain surface, such as choosing between a viaduct, a bridge or a tunnel.

However, it is important to refrain that caution must be taken during the construction of three-dimensional interfaces. Traffic simulation environments using them involve a larger number of agents and amount of data to be processed in real time than in bi-dimensional interfaces. The use of avatars with levels of detail, besides the optimization of vehicle behaviour algorithms, is fundamental to improve performance in monolithic processing.

Equally important is the distribution of this simulation to extend the scope of the traffic-simulated region. The presented structure showed efficiency and the idea of division in junctions seems to be the best solution to homogenize processing among machines. A possibility, which might increase the potential use of each of these distributed processing units, is to create junctions groups whose traffic flow total sum is similar. The agglomeration degree would be regulated according to the average flow intensity in each junction and with the processing capacity of each processing unit. These values would be optimized so that each unit had its processing capacity used to its full potential. The same would apply for scene terrain and object controlling units.

It is believed that the expansion of this simulation model to contain more types of vehicles, with more complex behaviours than the ones presented here, is possible. The insertion of pollution control and fuel consumption elements, cyclists and pedestrians flow, bus lines and car crashes also seems feasible. However, the larger the complexity of the model, the larger is the volume of data to be processed and, as a consequence, the higher the necessity for model distribution.

With the insertion of a network communication protocol and the reformulation of some data structures, the presented architecture seems to be capable of being easily adapted to a distributed model. With this adaptation, the construction of a transit scene involving more variables and each time more realistic becomes a project whose results will certainly and significantly contribute to the area of urban traffic analysis and planning.

6.1. Difficulties

Despite the good results with respect to simulation functioning, it is important to mention the problems found in its development. The most relevant one concerns the regulation of the vehicle behaviour algorithms. The influence of a certain number of forces in the vehicle would often produce abrupt movements which would erratically modify its trajectory along the lane.

During some of the simulation experiments, sparse problems with some of the simulated vehicles had occurred. They would stagnate when arriving to entry points of certain connections. The experiments were simulated again with vehicles with lower LODs and the problem seemed to have disappeared. The cause for the problem was not yet identified, but there are suspects that it occurs during the execution of the separation algorithm.

The separation algorithm is still a little unstable. When applied in situations of intense traffic, it apparently generates senoidal movements in the vehicles, making them leave lanes where they would have to remain. In order to minimize this problem, the degree of influence of this force behaviour over vehicles was reduced. Moreover, the vehicle also passed to suffer a reduction in its speed if located in an intermediate distance inside the current traversed lane, with the separation force between vehicles in the same activated lane activated.

Another problem was the excessive lane change of vehicles, which turns the transit a little more chaotic than normal. Moreover, in double-way roads, interaction between flows of opposing directions did not occur. This adds a level of higher complexity to the vehicles behaviour.

The automatic creation of traffic graphs helped considerably in the fast accomplishment of experiments. However, the synchronization of traffic lights is still in experimental stage and is the cause of some collisions during simulation.

6.2. Contributions

This dissertation contributes in diverse areas of research, including 3D traffic simulation, electronic games and virtual reality. The main contributions of this work are described in the following paragraphs.

Despite the great amount of currently available transit simulation tools, none was found with a three-dimensional interface executed without the aid of supercomputers. Although the work here presented still finds itself in an initial stage, it must be noticed its potential for execution in domestic PCs computers.

The automatic traffic graph generator is useful in the initial elaboration of urban transit traffic meshes for electronic games and virtual environments in general. The relationship of users as pedestrians and of vehicles as automatic entities could be implemented using network communication mechanisms. Its adaptation to generate an archive containing the information of the automatically constructed graph and its posterior manipulation using a graph editor may not only serve for the creation of fictional cities, but also for the creation virtual models for real cities having a more organized traffic mesh.

Mapping a traffic graph, by the definition of points and edges in an archive, accelerates, though just in part, the process of mapping the traffic of one specific region. Nevertheless, the insertion of a graphical editor for construction of the graph is an important implementation stage.

Finally, the modification of the pathfollowing algorithm, which instead of considering the distance of the vehicle from the centre of the lane, uses the distance between speed vectors, and of the algorithm separation, whose scope was reduced to closer vehicles, contributed significantly for the improvement of simulation performance.

6.3. Future Work

The current simulation model already allows the forecast of simple traffic situations in a reasonable realistic way. However, there is still much to be done before the current model reaches

a sufficiently flexible level to simulate any traffic situation. In this section, the main steps to be followed in order to turn this into reality are presented.

6.3.1. Vehicle Behaviour Improvement

First, the vehicles behaviours need to be improved in certain aspects. The separation algorithm needs to be more stable, so that the exerted force is capable to prevent the majority of vehicle collisions.

The vehicle area of collision needs to benefit with the model representing it, so the collision events only occur when an intersection between vehicle avatars occurs. Moreover, this algorithm must more precisely detect the relative spatial position between the vehicle that collides and the vehicle collided, in order to dislocate the cars in different ways according to their disposition from each other. Collision detection between vehicles in the same junction, but in different connections, also needs to be implemented.

The path following behaviour needs to be optimized for the *Runge-Kutta* interpolation model (Mathworld, 2004), whose precision is better than the one of the Euler model currently being used. This will turn the movement of vehicles less susceptible to the frame rates variations during simulation.

The vehicles still needs other behaviours to allow them to cross lanes of opposing directions without colliding with the opposite flow. They must also become capable of parking in commercial points and streets where the parking is allowed. Furthermore, a passing mechanism must be implemented according to the psychological behaviour of drivers. As other characteristics are inserted in the model, such as pedestrians, climatic bus stops and climatic variables, the vehicle behaviour is to become more realistic.

6.3.2. Simulation Distribution Modification

Secondly, some modifications will be necessary to distribute the simulation in independent processing units. These changes are presented next.

The methods for transmission of vehicles between connections will have to be made using a network protocol. The same will apply for drawing methods of the graphical objects. The information of each model such as references to its levels of detail, avatar position and orientation will be transmitted from the machine controlling each junction to the machine where the user

viewing the scene is. The machine of the user will already have all possible avatars stored locally and will draw each object whose information is being received.

The list of inactive vehicles and traffic graph structures, currently located in the Simulation class, will be contained in each junction, guaranteeing to the latter the necessary autonomy to execute in a different machine.

During the distribution of the processing, the ITranS_C class will initiate, synchronize and monitor system functioning, and will be executed in a separate machine together with the Simulation class. User control will also be executed in a different machine. This will make possible for more than one user to enter a single simulation. A user log control will, then, need to be created. The model will also need to be capable of providing these users with the ability to follow the movement of a specific vehicle in transit.

On the one hand, the insertion of groups of junctions will reduce the network traffic load in cases when many junctions with little traffic flow exist. On the other hand, an algorithm to group them and thus homogeneously distribute processing between machines will have to be implemented. This algorithm will need to take in consideration the traffic flow of the junctions and the processing capacity of each machine during the creation and distribution of these junctions groups.

6.3.3. Topographic Information Improvement

Thirdly, the insertion of topographical elements to the model will have to be done to guarantee a better orientation to the user in the analyzed region. Algorithms for positioning and automatic scale adjusting the terrain surface according to traffic graph must be implemented. This will reduce manual work during the construction of the simulation environment. Still, a graph editor must be created and be able to receive as input a map of the region whose traffic mesh is to be analyzed, so that the points and edges of the traffic graph are drawn on the map, instead of having their coordinates manually defined in an archive.

Another possibility is the standardization of the data configuration interface, which would receive real data, extracted from maps with different geographic information. This will automate all the process of data acquisition.

6.3.4. Auto-adjustable LOD System

Finally, the activation distance and the use of different LODs in the scene objects could vary according to application performance. For a simulation with many vehicles or including a vast geographic area, levels of detail with inferior graphical quality would be used to represent objects and vehicles. When there were a little amount of vehicles being simulated or if the simulation involved a small geographic area, the objects levels of detail would be higher and would provide the user with a better graphical quality of the scene.

6.3.5. Final Considerations

With these changes, the generation of more complex and realistic simulations will happen more efficiently and less costly. The development of most of them, however, is given non-trivially and requires quite an effort for its conclusion. The contribution between research groups in other areas, such as Distributed Systems, Computer Networks and Theory of Computation is important to accelerate the architecture change, the algorithmic and the network transference performance increase for the simulation. Furthermore, the contact with companies responsible for the control of urban traffic must be considered so that the interface satisfies the final user. The interface will then present only the necessary functionalities, eliminating the efforts in developing functionalities of doubtful utility and, thus, increasing the research and development team potential.

Bibliography

The bibliography is organized in two sections. The first one includes references cited in the dissertation and used in the research. The second contains references which, despite not mentioned in the dissertation, had also been used in the development of the research.

References

- [1] Adzima, J., "AI Madness: Using AI to Bring Open-City Racing to Life". Gamasutra, published in January 24th, 2001. Available at: www.gamasutra.com/features/20010124/adzima_01.htm. Accessed in: June, 2004.
- [2] Al-Shihabi, T and Mourant, R.R. "A Framework for Modeling Human-like Driving Behaviors for autonomous Vehicles in Driving Simulations". Proceedings of the fifth international conference on Autonomous agents, International Conference on Autonomous Agents, 2001, Montreal, Quebec, Canada, pgs.: 286 - 291, ISBN: 1-58113-326-X.
- [3] Barros, P. G., Kelner, J., "Simulação de Tráfego: uma Experiência com Realidade Virtual". Proceedings of SVR 2003 - VI Symposium on Virtual Reality, COC editor, 2003. v.1. p.140 - 151, Ribeirão Preto, Brazil, October, 2003.
- [4] Bayarri, S. et al. "Virtual Reality for Driving Simulation". Communications of the ACM, May, 1996, pgs.: 72-76, vol. 39, n. 5.
- [5] Blue, M. and Bush, B. "Information Content in the Nagel-Schreckenberg Cellular Automaton Traffic Model". Physical Review E **67** (2003), p. 047103.
- [6] Brutzman, D. P.; Macedonia, M. R. and Zyda, M. J. "Internetwork Infrastructure Requirements for Virtual Environments". Proceedings of the Virtual Reality Modeling Language (VRML) Symposium, San Diego Supercomputer Center (SDSC), San Diego, CA, December 13th to 15th, 1995.
- [7] Cameron, G.; Wylie, B.J.N and McArthur D. "PARAMICS : Moving Vehicles on the Connection Machine". Proceedings of the 1994 ACM/IEEE conference on Supercomputing, Conference on High Performance Networking and Computing, 1994, Washington, D.C., pgs.: 291 – 300, ISBN ~ ISSN: 1063-9535, 0-8186-6605-6.
- [8] Clark, J. and Daigle, G. "Importance of Simulation Techniques in Its Research and Analysis". Proceedings of the 29th Conference on Winter Simulation Conference, Winter Simulation Conference, Atlanta, Georgia, United States, 1997, pgs.: 1236 - 1243, ISBN:0-7803-4278-X.
- [9] De Floriani, L. and Magillo P., *Regular and Irregular Multi-resolution Terrain Models: a Comparison*, Proceedings - 10th ACM International Symposium on Advances in Geographic Information Systems, pgs.: 143 - 148 , 2002 , ISBN:1-58113-591-2. *World-up R5 User's Guide*, 1997-2000 by Engineering Animation, Inc.;
- [10] DeLeon, V.; Berry, R.; *Bringing VR to the Desktop: Are You Game?*, IEEE Multimedia, April-June, 2000 edition (Vol. 7, No. 2), pgs.: 68-72.
- [11] Díaz, A.; Vázquez, V. and G. Wainer "Vehicle routing in Cell-DEVS models of urban traffic". Proceedings of European Simulation Symposium. Marseille, France. 2001.

- [12] Discreet. Discreet website. Available at: <http://www.discreet.com/>. Accessed in: December, 2004.
- [13] Fullford, D. "Distributed Interactive Simulation: It's Past, Present, and Future". Winter Simulation Conference Proceedings, 1996, pgs.: 179-185.
- [14] Harris, L.R.; Jenkin, M.; Zikovitz, D.; Redlick, F.; Jaekl, P.; Jasiobedzka, U.; Jenkin, H. and Allison, R. "Simulating self motion I: cues for the perception of motion". Virtual Reality, 2002, volume 6, number 2, pgs.: 75 – 85.
- [15] Hsin, V.J.K. and Wang, P.T.R "Modeling Concepts for Intelligent Vehicle Highway Systems (IVHS) Applications". Proceedings of the 24th conference on Winter simulation, Winter Simulation Conference, 1992, Arlington, Virginia, United States, pgs.: 1201 – 1209, ISBN: 0-7803-0798-4.
- [16] Jayakrishnan, R. and Mahmassani, H.S. "Dynamic Simulation-Assignment Methodology to Evaluate In-Vehicle Information Strategies in Urban Traffic Networks". Winter Simulation Conference Proceedings, Balci, O. Sadowski, R. P., and Nance, R. (eds.), pgs.: 763 - 769, 1990.
- [17] Klein, U.; Schulze, Th.; Straßburger, S. and Menzler, H.P. "Traffic Simulation Based on the High Level Architecture". Proceedings of the 1998 Winter Simulation Conference, eds. Medeiros, D.J. and Ed Watson, SCS, Washington.
- [18] Lemessi, M., "An SLX-based Microsimulation Model for A Two-lane Road Section" Proceedings of the 2001 Winter Simulation Conference. WSC 2001, Arlington, VA, USA, December 9th to 12th , 2001, pgs.: 1064-1071, ISBN:0-7803-7309-X .
- [19] Li, H. and Lim, A. "Local Search with Annealing-like restarts to Solve the vehicle routing Problem with Time Windows". Proceedings of the 2002 ACM symposium on Applied computing, Symposium on Applied Computing, 2002, Madrid, Spain, pgs.: 560 - 565 , ISBN:1-58113-445-2.
- [20] Liu, R; Clark, S.D.; Montgomery, F.O. and Tate, J. (2000). *The Microscopic Modelling of Kerb Guided Bus Schemes*. Presented in Transport Research Board Annual Conference, Washington, 2000.
- [21] Liu, R. and Tate, J. (2000). *MicroSimulation Modelling of Intelligent Speed Adaptation System*. Paper presented in European Transport Conference, Cambridge, September 2000.
- [22] Lo Tártaro, M.; Torres, C. and Wainer, G. "Defining models of urban traffic using the TSC tool". Proceedings of the Winter Simulation Conference, Washington, DC. U.S.A. 2001.
- [23] Macedonia, M. R. and Zyda, M. J. "A taxonomy for networked virtual environments". Proceedings of the 1995 Workshop on Networked Realities, 1995.
- [24] Macredie, R.; J. E. Taylor, S.; Yu, X. and Keeble R. "Virtual Reality and Simulation: An Overview". Proceedings of the 28th Conference on Winter Simulation, pgs.: 669 - 674, Coronado, California, United States, December 8th to 11th, 1996.
- [25] Manouselis, N.; Karampiperis, P. and Kosmatopoulos, E. "A multi-agent, microscopic traffic simulation architecture incorporating entities with adaptive behaviors". Proceedings of the 1st Human Centered Transportation Simulation Conference, Iowa, Novembro de 2001.
- [26] Marson, F., Jung, C. and Musse, S. "Modelagem Procedural de Cidades Virtuais". Simpósio Brasileiro de Realidade Virtual, SVR2003, Ribeirão Preto, Brazil, October, 2003.

- [27] Wolfram Research. Mathworld – The Web’s Most Extensive Mathematics Resource. Available at: <http://mathworld.wolfram.com/>. Accessed in: October, 2004.
- [28] Molofee, J. NeHe Productions. Available at: <http://nehe.gamedev.net/>. Accessed in: October 5th, 2004.
- [29] Molofee, J. NeHe OpenGL Tutorials. Available at: <http://nehe.gamedev.net/lesson.asp?index=01>. Accessed in: August, 2004.
- [30] Morar, S. S.; Macredie, R. and Cribbin, T. “An Investigation of Visual Cues used to Create and Support Frames of Reference and Visual Search Tasks in Desktop Virtual Environments”, 2002, Virtual Reality, volume 6, fascicule 3, pgs.: 140 -150.
- [31] OpenGL.org. OpenGL - The Industry's Foundation for High Performance Graphics. Available at: <http://www.opengl.org/>. Accessed in: March 2005.
- [32] Owen, L.E.; Zhang, Y.; Rao, L. and McHale, G. “Traffic Flow Simulation Using Corsim”. Proceedings of the 2000 Winter Simulation Conference, 2000. Available at: <http://www.informs-cs.org/wsc00papers/152.PDF>.
- [33] Pantel, L. and Wolf, C. L. “On the Impact of Delay on Real-Time Multiplayer Games”. International Workshop on Network and Operating System Support for Digital Audio and Video: Network Issues for Video and Games, New York, NY, USA, ACM Press, pgs.: 23-29, 2002.
- [34] Paruchuri, P.; Pullalarevu, A.R. and Karlapalem, K. “Multi Agent Simulation of Unorganized traffic”. Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1, International Conference on Autonomous Agents, 2002 Bologna, Italy, pgs.: 176 - 183 , ISBN:1-58113-480-0.
- [35] Trolltech. Qt Overview. Available at: <http://www.trolltech.com/products/qt/>. Accessed in: March 2005.
- [36] Raja, D; Bowman, D.A.; Lucas, J. and North, C. “Exploring the Benefits of Immersion in Abstract Information Visualization”, *8th International Immersive Projection Technology Workshop*, 8 pages, Iowa State, May 2004.
- [37] Raja, D. and Bowman, D.A. “A Method for Quantifying the Benefits of Immersion Using the CAVE”, Presence Connect, June 2004.
- [38] Reisman, R. and Ellis, S. “Augmented Reality for Air Traffic Control Towers”. Proceedings of the SIGGRAPH 2003 Conference on Sketches & Applications: in conjunction with the 30th Annual Conference on Computer Graphics and Interactive Techniques, International Conference on Computer Graphics and Interactive Techniques, San Diego, California, 2003, pg.: 1.
- [39] Reynolds, C. W. “Steering Behaviors For Autonomous Characters”. Proceedings of Game Developers Conference 1999, Miller Freeman Game Group, San Francisco, California, pgs.: 763-782.
- [40] Roberson, G.; Czerwinski, M. and V. Dantzich “Immersion in Desktop Virtual Reality”, Proceedings of the 10th annual ACM Symposium on User Interface Software and Technology, 1997, UIST'97, pgs.:11-19.
- [41] Roehl, B “Distributed Virtual Reality – An Overview”. Proceedings of the first symposium on Virtual reality modeling language, Virtual Reality Modeling Language Symposium, 1995, San Diego, California, United States, pgs.: 39 – 43, ISBN:0-89791-818-5 .

- [42] Schulze, T.; Lemessi, M. and Filippi, F. “Simulation of a night taxi-bus service for the historical center of Rome”. Proceedings of the 2001 Winter Simulation Conference, WSC 2001, Arlington, VA, USA, December 9th to 12th, 2001, pgs.: 1072-1078.
- [43] Schulze, T. and Fliess, T. “Urban traffic Simulation with Psycho-physical Vehicle-following Models”. Proceedings of the 29th conference on Winter simulation, Winter Simulation Conference, 1997, Atlanta, Georgia, United States, pgs.: 1222 - 1229, ISBN: 0-7803-4278-X.
- [44] Stappers, P.J.; Gaver, W. and Overbeeke, K. “Beyond the limits of real-time realism: Moving from stimulation”. L. Hettinger & M. Haas, Psychological Issues in the Design and Use of Virtual and Adaptive Environments. Lawrence Erlbaum Associates, Inc., Mahwah, NJ, 2000.
- [45] Schmitz, M. “Sistema de controle de tráfego urbano utilizando sistemas multi – agentes”. Blumenau, 2002. Final graduation work (BSC in Computer Science) Universidade Regional de Blumenau.
- [46] Schmitz, M. and Hübner, J. F. “Uso de SMA para avaliar estratégias de decisão no controle de tráfego urbano”. Seminário de Computação, 2002, Blumenau. Anais do XI Seminário de Computação. Blumenau: FURB, 2002. pgs.: 243-254. Available at: <http://www.inf.furb.br/~jomi/pubs/2002/Schmitz-seminco2002.pdf>.
- [47] Seneviratne, P., Access *Traffic Simulation Model (ACTSIM)*, final report for Project 78 of, Intelligent Transportation Systems Program, Utah State University, Logan, UT, November 2001.
- [48] *SMARTTEST - Final Report for Publication*, ITS, University of Leeds (GB), Project funded by the European Commission about the Transport RTD Programme Project of the 4th Framework Programme, January 13th, 2000, Contract Number: RO-97-SC.1059.
- [49] Smith, P., “GDC 2002: Polygon Soup Programmer’s Soul: 3D Pathfinding”. Gamasutra, published in April 4th, 2002. Available at: www.gamasutra.com/features/200220405/simth_01.htm.
- [50] Sun, J.; Yu, X.; Baciú, G. and Green, M., “Template-based Generation of Road Networks for Virtual City Modeling”. Proceedings of the ACM symposium on Virtual reality software and technology, Virtual Reality Software and Technology, 2002, Hong Kong, China, pgs.: 33 – 40, ISBN: 1-58113-530-0.
- [51] Tavares, J.; Pereira, F. B.; Machado, P. and Costa, E. “On the Influence of GVR in Vehicle Routing”, 2003.
- [52] Thangiah, S. R.; Shmygelska, O. and Mennel, W. “An Agent Architecture for Vehicle Routing Problems”. Computer Science Department, Slippery Rock University. SAC’2001, ACM, 2001.
- [53] Wagner, C. “Developing Your Own replay System”. Gamasutra, published in February, 4th, 2004. Available at: www.gamasutra.com/features/20040204/wagner_01.shtml.
- [54] *World-up R5 User’s Guide*, 1997-2000 by Engineering Animation, Inc.;

Recommended Bibliography

Boehm-Davis, D.A.; Marcus, A.; Green, P.A., Hada, H. and Wheatley, D. “The Next Revolution: Vehicle Use-Interfaces and the Global Rider/Driver Experience”, CHI '03 extended abstracts on Human factors in computing systems, Conference on Human Factors in Computing Systems, Ft. Lauderdale, Florida, USA, 2003, pgs.: 708 – 709, ISBN: 1-58113-637-4.

Ben-Moshe, B.; Katz, M.; Mitchell, J. and Nir, Y., *Visibility Preserving Terrain Simplification – An Experimental Study*, Proceedings - 18th Annual ACM Symposium on Computational Geometry, pgs.: 303-311, June 2002.

Companhia de Trânsito e Transporte Urbano do Recife. CTTU on-line, website. Available at: <http://www.recife.pe.gov.br/pr/servicospublicos/cttu/>. Accessed in: January 19th, 2005.

Morar, S.S. and Macredie, R. D. “Special Issue on “Interacting with Desktops Virtual Environments: Perception and Navigation”, *Virtual Reality*, May 19th, 2004, volume 7, pgs.:129 - 130.

Mamber, U. “Introduction to algorithms – A Creative Approach”. Addison- Wesley, 1989, ISBN: 0-201-12037-3.

Microsoft Corporation. Microsoft Developer’s Network Site. Available at: <http://msdn.microsoft.com/>. Accessed in: December 2004.

Kohl, N. Nate Kohl’s Cpp Reference Site. Available at: <http://www.cppreference.com>. Accessed in: January 2004.

Schildt, H. “C Completo e Total – 3^a edição revista e atualizada”. 1995 McGraw-Hill, 1997 Makron Books do Brasil Editora Ltda., ISBN: 85-346-0595-5.

Sheridan, T.B. “Interaction, Imagination and Immersion Some Research Needs”. Proceedings of the ACM symposium on Virtual reality software and technology, *Virtual Reality Software and Technology*, Seoul, Korea, 2000, pgs.: 1 – 7, ISBN: 1-58113-316-2.

A.2. User Command Keys

Below is presented a table containing user keyboard commands. Table 16 lists viewpoints control commands while Table 17 lists the user navigation commands. In both tables, the left column presents the keys to be pressed for each control command while the right column describes their actions.

A.2.1. Viewpoint Control Commands

Besides navigation commands, the user also has viewpoints control commands, as already mentioned in section 4.3. The keyboard activation keys for these commands are listed in Table 16 below.

Table 16: User viewpoint control commands.

<i>Command key</i>	<i>Command function description</i>
INSERT	Saves to the viewpoints list the current user view.
DELETE	Erases the last viewpoints visited by the user from the list of viewpoints.
ENTER	Views the last viewpoint visited. If this key is pressed more than once without the user moving, the user sequentially and cyclically visits the stored viewpoints in the viewpoints list.
Teclas de função F1 a F12	Visualization shortcuts from the first to the twelfth viewpoints stored.
SHIFT + teclas de função F1 a F12	Erase from the first to the twelfth viewpoints stored.

A.2.2. Navigation Commands

As mentioned in section 4.4, the user movement is divided in basically three groups. The activation keyboard keys for the control commands of the three types of movement are shown in Table 17.

Table 17: User navigation commands.

<i>Command key</i>	<i>Command function description</i>
Up arrow (↑)	Moves the user avatar in the direction to which it is pointing.
Down arrow (↓)	Moves the user avatar in the opposite direction to which it is pointing.
Left arrow (←)	Moves the user avatar to the left and perpendicularly to the direction to which it is pointing.
Right arrow (→)	Moves the user avatar to the right and perpendicularly to the direction to which it is pointing.
Home	Moves the user avatar up in the direction of its normal vector.
End	Moves the user avatar down in the opposite direction of its normal vector.
Space	Resets the pitching angle, responsible turning the avatar up or down, by setting its value to zero. By doing this, the user points to a direction parallel to the XZ plane.
CTRL + ↑	Turns the user up around its X axle.
CTRL + ↓	Turns the user down around its X axle.
CTRL + ←	Turns the user left around the Y axle.
CTRL + →	Turns the user right around the Y axle.
SHIFT + ↑	Moves the user avatar in the direction to which it is pointing, but keeps him parallel to the XZ plane in his current height.
SHIFT + ↓	Moves the user avatar in the opposite direction to which it is pointing, but keeps him parallel to the XZ plane in his current height.
SHIFT + ←	Moves the user avatar to the left and perpendicularly to the direction to which it is pointing, but keeps him parallel to the XZ plane in his current height.
SHIFT + →	Moves the user avatar to the right and perpendicularly to the direction to which it is pointing, but keeps him parallel to the XZ plane in his current height.
SHIFT + space	Resets the yawing angle, responsible for turning the user to the left or to the right, by setting its value to zero. By doing this, the user points to a direction parallel to the YZ plane.

A.3. Algorithms

Below, the main algorithms used in this dissertation are described. They are written in pseudo-code without need of previous knowledge in a specific programming language.

A.3.1. Vehicle Path Following Algorithm

- The vehicle speed desired vector is calculated. It is equivalent to the normal of the vector that goes from the vehicle position to its current destination point.
- If the distance between the vector of the current speed and the desired speed vector is greater than half the width of a lane plus half the width of the vehicle, then: {
 - An attractive force to the lane is calculated. Its normal is the vector that goes from the current speed to the desired speed (see Figure 16).
 - This force is applied to the vehicle.

A.3.2. Vehicle Collision Detection Algorithm

- For all vehicles in the connection: {
 - If the vehicle is in the same lane as the current vehicle, then: {
 - The diagonal length of the vehicle is calculated.
 - If the distance between one of the collision points of the vehicle and all the collision points of the current vehicle is smaller than the diagonal length of the vehicle, then, a collision occurred (see Figure 17).

A.3.3.

Vehicle Separation Algorithm


- For all vehicles in the connection of the current vehicle: {
 - If the vehicle is in the same lane as the current vehicle and its ticket number is smaller than the one of the current vehicle, then: {
 - The distance between the two vehicles is calculated.
 - If the distance is smaller than the minimum separation distance allowed for vehicles in the same lane, then: {
 - If the vehicle is pursuing the end point of the lane and if it is slightly deviated (10m or more) from the lanes end and start points, then, the vehicle is decelerated proportionally to the reason between its current distance and the minimum distance allowed from the other vehicle.
 - A repulsion force in the opposite direction of the current connection is calculated. Its value increases proportionally to the reason between its distance and the minimum distance allowed from the other vehicle (see Figure 18).
 - This force is applied to the vehicle.

If the vehicle is not in the same lane as the current vehicle, then: {

- The distance between the two vehicles is calculated.
- If the vehicles are almost parallel to each other, only with a difference of 1m or less between their extremities, then: {
 - If the distance between them is smaller than 1/8 of the width of the lane, then: {
 - A repulsion force is calculated, whose normal is the vector that goes from the end point of the lane of the vehicle being approached to the end point of the lane of the vehicle approaching (see Figure 19).
 - This force is applied to the vehicle.

A.3.4.

Dissertação de Mestrado apresentada por **Paulo Gonçalves de Barros** à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título “**Itrans – Simulador de Trânsito 3D**”, orientada pela **Profa. Judith Kelner** e aprovada pela Banca Examinadora formada pelos professores:



Prof. Fernando da Fonseca de Souza
Centro de Informática / UFPE


Prof. Claudio Kirner
Faculdade de Ciências e Tecnologia da Informação / UNIMEP


Prof. Verônica Teichrieb
Escola Politécnica / UPE


Profa. Judith Kelner
Centro de Informática / UFPE

Visto e permitida a impressão.
Recife, 1 de março de 2005.


Prof. JAELSON FREIRE BRELAZ DE CASTRO
Coordenador da Pós-Graduação em Ciência da Computação do
Centro de Informática da Universidade Federal de Pernambuco.