Three-dimensional Urban Traffic Simulation with ITranS

Paulo G. de Barros, Judith Kelner, Rodrigo C. Farias, Dinaldo A. Pessoa

Centro de Informática
Universidade Federal de Pernambuco
{pgb, jk, rcf3, dap2}@cin.ufpe.br

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. Nowadays, different mathematical models exist to facilitate this activity. They are virtually built and computer simulated and are basically categorized in two groups: macro and micro-simulation models. The first presents traffic as flow values whereas the second specifies traffic by representing each vehicle. The primary goal of this work was to develop a singular traffic micro-simulation tool using a desktop Virtual Reality (VR) interface and improved steering behaviours. This tool will serve as a basis for the future development of a transit simulator with a distributed architecture.

1. Introduction

Intense and unorganised transit is one of the main problems of large cities worldwide. Its solution is still considered a great challenge nowadays.

A diversity of mathematical models has been created to study such problem and minimize its effects. They represent vehicle flow by edges and nodes in a graph. Edges represent roads and nodes represent junctions. Extra features are added to edges, such as vehicle flow, number of lanes, lane width, velocity limit and length. Features are also added to nodes. Based on these basic primitives the virtual road structures are built.

The abovementioned mathematical models, denominated macro-simulation models, are built and simulated in computers. They represent transit as the flow in road sections. Another adopted model is the micro-simulation, based on the simulation of the behaviour of each transit entity, that is, vehicles or pedestrians. These entities follow distinct paths according to the traffic flow in each street.

With computational simulation, it is possible to build better planned roads in an urban area, since the road plan, which consists of its number of lanes and its traffic lights and road signs position, may be re-evaluated and adjusted as much as necessary until the expectation, which motivated its design, is reached [9]. And it is with the purpose of improving the urban traffic planning process that ITranS is here proposed.

The use of VR interfaces increases the range of visualization and interaction possibilities within a transit application. The use of more immersive non-desktop devices grants more precision in the user interaction and perception of the virtual environment, thus increasing her productivity. Although in an initial stage, some of these advantages have already their results published and are well-known in the research community [3].
Furthermore, a third dimension is a very important feature for simulation tools, since it allows information superposition, flexible navigation, realistic environment creation and visualization of multiple data difficult to be seen in a bi-dimensional (2D) interface, such as terrains or other height dependent data. Perspective view enables the visualization of a broader area of analysis and interaction among three-dimensional (3D) objects, such as terrains and roads, bridges, tunnels or viaducts. Certainly, the use of new steering behaviours algorithms makes of ITranS a singular transit simulation tool, capable of simulating transit quite realistically and with little restriction to path patterns.

The remainder of this paper is structured as follows. Section 2 defines the project goals. Section 3 reviews prior work in distributed real-time applications and transit simulation, as well as a few virtual reality (VR) concepts. Section 4 presents the ITranS tool, with its 3D interface, flexible graph mapping and realistic vehicle behaviours. Section 5 details the experiments carried out to analyze the performance of the tool in different scenarios. Lastly, section 6 gives the conclusion.

2. Design Goals

The primary goal of this work was to develop a traffic simulation tool with desktop VR interface to provide an adaptable creation of virtual scenes based on real world traffic for realistic VR driving simulations, tourism, games, military training or strategic systems and entertainment in general. This type of interface was chosen due to its reduced cost and broad use, as well as its potential application to portable PCs.

The application, called ITranS (Interactive Transit Simulator) [7], is composed of a simulator that controls terrain surface, vehicles, levels of detail (LOD) of the objects and traffic lights; and by a 3D interface which allows users to navigate and explore different points of view and types of interaction.

The simulator is structured to serve as a basis for the development of a simulator with a distributed architecture, a 3D version of the ones developed by Cameron in 1994 [4] and Klein in 1998 [8]. A feature which must be highlighted in ITranS is the creation of improved versions of artificial life algorithms to control vehicles movement based on the work of Reynolds in 1999 [14]. ITranS has also a 3D interface based on OpenGL with LODs applied to objects. In addition, it enables the configuration of these objects such as terrain, vehicles, semaphores and the traffic graph itself.

3. Related Work

There are a number of research groups and companies working with transit simulation [1] [2] [4] [8]. A research carried by the Institute of Transport Studies in the University of Leeds, United Kingdom [6], lists 57 tools that deal with diverse traffic situations.

These tools have a number of features in common. Most of them have its update time based on discrete units, although few tools have already been implemented with real-time updates. Regarding vehicles route definition, once both origin and destination points are specified, intermediate points may be defined in real-time or at initialization. The operation of traffic lights is an common feature. Moreover, few are the tools that consider pedestrians, cyclists, public transports or parking behaviours.
Most traffic models include state output information, such as the average velocity of vehicles on a road section, their trip times or traffic jams lengths, formation and localization. Simulators also have configuration parameters. Integration with other tools seldom occurs. Furthermore, simulation time is usually faster than real lifetime.

These tools use macro-simulation or micro-simulation models or even both of them. Multi-agent systems [10] have also been considered using SACL and KQML or C++ and the Qt graphical library [13]. Solutions for the generation of roads based on 2D images have been used in the construction of urban models of cities and their streets. Nevertheless, the previous preparation of these images is still necessary in most cases.

In general, model calibration parameters are application specific. Statistics generated from the collected data are frequent calibration parameters. Validation seldom occurs and is done by comparing output to field research data. Lane change is well mapped. Vehicles’ behaviours is based on psychophysical models [15] and vehicles may have different velocities, accelerations and dispositions for passing or running [1].

With regard to VR, models have evolved to become each time more realistic, either by the use of game engines for processing the 3D environment, or by the definition of improved LOD techniques. Distributed virtual environments use technologies such as IEEE DIS and the HLA architecture [5] [8]. Three-dimensional traffic visualization models are also under evolution [2].

Inside this context, ITranS may be classified as a combined desktop micro-simulator. It can adapt to simulate freeway or urban traffic situations and run in portable computers. Its performance is limited only by computer processing capabilities. ITranS distinguishes itself from other applications for the use of new artificial life algorithms to represent vehicle behaviour, providing movements without specific patterns. Another outstanding feature is the use of domestic hardware and 3D interface to provide full transit simulation functionality with little or no equipment investment. Although in an initial stage, ITranS has the potential to help in the solution of urban traffic problems.

4. ITranS Tool

ITranS is a tool with an optimized architecture to allow flexible parameterization and distribution of simulation processing. To accomplish this, it was built with an improved architecture, which divides the tool in three modules, detailed in Figure 1 below.

- **Data Configuration System**: Sets up the basic application structures;
- **Simulator**: Contains all logical system, including vehicles, traffic lights and user;
- **Interface**: Presents the graphical scene and processes input commands from the user.

![ITranS basic modules and their interaction.](image)
The interaction between these components is given as follows. First, the data configuration system initializes the world entities. The roads are created based on a graph, which will be mapped to junctions and road sections, also called road links. Each graph edge has specific values of vehicle flow per minute and traffic light times. Vehicles, semaphores and terrain are inserted according to the road flows and surface information. The compilation of this information is done during data configuration. Next, the simulator is activated, performing, in every processing cycle, the corresponding actions for each entity in the virtual environment. Also, at each frame, the user interface is updated and input commands are processed.

The vehicles routes are generated before they start moving and area, being inflexible during traversal. The maximum velocity of a vehicle is fixed and lane change happens randomly only during change of vehicle control between adjacent road sections.

4.1. Data Configuration

During data configuration, there are three archives that set up the simulator: the first of them describes the traffic graph, the second includes terrain configuration and the third contains vehicles configuration. The three of them are further detailed below.

4.1.1. Traffic graph

The graph configuration archive may specify graphs in two different modes: manual or automatic. The first mode configures the application by using a set of points and edges individually described by the user. The second automatically generates a homogeneous grid of junctions and bi-directional road links according to standard edge and point configuration parameters set by the user.

By providing two different approaches to graph generation, the tool intends to be helpful for a large number of applications, such as virtual city tours, urban games, simulation, training, architectonic models, etc. The automatic generation technique could be used for fast construction of virtual environments, whereas the manual generation would help in the precise and realistic creation of urban traffic networks.

4.1.2. Terrain

Terrain generation varies according to the traffic graph generation mode specified. For grid graph generation, terrains consist of simple squares that fill in the holes within blocks. For manually generated graphs, terrain is defined in a second archive describing a set of terrain meshes, which will represent different levels of detail, as well as their positions and their orientations. The terrain specification file may also be used to define other objects in the simulation scene, such as buildings and trees.

4.1.3. Vehicles

The third archive sets vehicle features, such as average weight, velocity and acceleration. It also sets the maximum number of vehicles simulated simultaneously.

4.2. Simulation

The core and most important part of ITranS is its simulator. This part controls the execution flow of the virtual world, updating vehicles and traffic lights, controlling their
interaction and regulating their LODs and of the terrain. With respect to the user, it changes her position, orientation and viewpoints according to interaction. The simulator, its entities and their behaviours are described in the following subsections.

4.2.1. Architecture

ITranS simulator has an architecture whose processing is done junction by junction, in which each junction autonomously updates its connections. And, since each connection updates the vehicles in its lanes, junction distribution in different machines is possible.

The traffic network of a region is mapped in a bidirectional graph, where graph nodes are present in every beginning and end of contiguous road sections. A contiguous road section does not intercept any node. Each node is mapped to a traffic junction.

A one-way road section is represented by one edge whereas a two-way road section is represented by two edges of opposite directions; contiguous road sections that are not rectilinear are represented by a set of edges. Each junction controls the traffic in its edges of incoming traffic, the in-connections. Incoming and outgoing edges in the borders of the map are responsible for inserting and removing vehicles in the simulation.

The 3D scene is a set of 3D objects representations. Its control is distributed among junctions, which contain these objects LODs and adjust them according to their distance from user. These adjustments are made when the scene is graphically built.

Figure 2 shows ITranS architecture and the relation among its main entities.

- **User:** Allows the user to navigate in the scene. Contains her avatar and point of view;
- **Simulation:** Controls the virtual environment and contains junctions and terrains;
- **Junction:** Represents road intersections and controls in-connections;
- **Terrain:** Represents the topographic surface of the simulated region;
- **In-connection:** Represents a road section and controls its vehicle flow;
- **Vehicle:** Represents a mobile entity that moves along the road sections in its path.

As computer processing is centralized in this version, an extra entity was created, the ITranS_C class. It contains the entire simulation environment, initiating and controlling the OpenGL scene [12], the terrain, avatars, user interaction and junctions. The latter have a list of in-connections with their vehicles and traffic lights. The NeHe graphical library [11] was used to simplify the OpenGL settings. They are supporting methods for the application. Therefore, ITranS_C manages all junctions and terrains. Each junction manages a set of in-connections. Each in-connection manages all the vehicles inside it, as well as its semaphore. Hence, the entire traffic flow is controlled.
4.2.2. **Vehicle Path**

The path of a vehicle consists of a list of in-connections names contained in the traffic network, by which the vehicle must sequentially pass. Vehicle path generation consists in sequentially choosing in-connections in adjacent junctions. The first in-connection is always an entrance connection. From there on, a cycle is followed until an exit connection is reached. The vehicle probability of path choice is equal to the flow on each in-connection divided by the total flow on its controlling junction.

4.2.3. **Vehicle Set-up**

The initial configuration of the vehicles is done as follows: the vehicles are inserted in the simulation inside entrance connections at the borders of the traffic graph. Dead end streets will also work as entrance or exit points for vehicles.

According to the traffic flow per minute in an entrance connection, vehicles are created at 60-seconds intervals. Vehicle distribution in this interval is homogeneous and random. During the insertion of a vehicle, its path is created and kept fixed during path traversal. Since lanes are randomly chosen, traffic flow in a connection is homogeneous.

Initial and maximum velocity and the mass of each vehicle are also defined randomly, based in average values for domestic vehicles available in the marketplace.

4.2.4. **Vehicle State Update**

The spatial 3D points sequentially chased by a vehicle are the start and end points of the randomly chosen lanes for each in-connection in the vehicle list. Vehicle control is passed from one in-connection to another following the order set by the vehicle path.

If a vehicle reaches the end point of the last lane to be followed, the vehicle is reset, deactivated and placed in a list with inactive vehicles. It is then available for being activated and re-entering the application by entrance in-connections requesting vehicles.

Therefore, vehicles may be either active, moving along in-connections, or inactive and occult, waiting for their insertion in the simulation or their transference between in-connections of adjacent junctions. The scene is composed of a limited number of vehicles, which are activated according to flow demand.

4.2.5. **Semaphores**

Each traffic light controls only the flow of the connection it belongs to, and can be understood as a state machine. Four are the states of a traffic light during its functioning.

The first state is activated only in the first cycle. In this state, the semaphore remains paused in the red light for a small period of time. It is used to synchronize adjacent semaphores or semaphores interfering in other semaphores traffic flow. The other three states are yellow, red, and green light. In the green light state, traffic is allowed to flow freely. In the red state, traffic is forbidden from flowing. Finally, in the yellow state, the flow behaves identically as in the green light state. This similarity was used to simplify implementation and, nevertheless, maintain a realistic visualization.
4.2.5.1. Manual and Automatic Configurations

Semaphores may be configured either automatically or manually depending on how the traffic graph was generated. If the graph was generated manually, the semaphore parameters are passed as four extra parameters for each in-connection.

If the graph is automatically generated, semaphores are automatically calculated. Given a grid graph, an initial halt time \( t \) is defined for its rows. Traffic lights located in the \( ith \) column have an initial halt time of \( i \times t \). The same logic is applied to semaphores controlling the opposite side of the road. \( t \) is the average time a vehicle takes to traverse between adjacent junctions. Its value is the division of the distance between junctions by the vehicles average velocity. The time settings for the three semaphore states: green, yellow and red are calculated as a set of supplementary intervals of \( t \).

4.2.6. Vehicle Behaviour

Vehicles have three different types of behaviour. Each of them can directly affect its acceleration, velocity or position. The separation and path-following behaviours apply a force to the vehicle that changes its acceleration, whereas the parking behaviour directly affects its velocity and the collision and road following behaviour affect its position. These behaviours are described below.

**Path-following:** an ideal velocity vector \( V_i \) is calculated by multiplying the normalization of the vector that goes from the vehicle position to its destination by its maximum velocity. If the distance \( d_e \) from this vector and its velocity \( V \) is greater than a distance \( d_p = (W_l - W_v)/2 \), where \( W_l \) is the lane width and \( W_v \) is the vehicle width, as in Figure 3, a force \( F \) is applied. \( F \) has the same orientation as the vector resultant from the difference \( V_i - V \) and its intensity is proportional to this difference, which is \( d_e \).

![Figure 3. Path following algorithm calculation.](image)

**Parking:** using a system of tickets for each connection [4], vehicles order is controlled during red light parking. If the traffic light of the connection where the vehicle is becomes red, three vehicle behaviours are possible. If a vehicle \( Vc \) 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 \( p_d \), which may be set to the initial point of the lane \( p_i \) or its end point \( p_f \) could also be substituted by a parking point \( p_e \), according to the conditions below.

- If \( p_d = p_e \), then \( p_d = p_e \);
- If \( p_d = p_i \) and \( D_{Pi} \geq D_{Pe} \), then \( p_d = p_e \);
- If \( p_d = p_i \) and \( D_{Pi} < D_{Pe} \), then nothing happens.

\( D_{Pi} \) and \( D_{Pe} \) are the distances from the vehicle to \( p_i \) or \( p_e \) respectively.

**Separation:** If two vehicles in the same street section reach a minimum distance from each other, a separation force is applied. If vehicles are in the same lane, a repulsion force with opposite direction to the lane flow decelerates the vehicle further.
behind. If vehicles are in adjacent lanes, a force perpendicular to the lanes with a direction opposed to the adjacent lane is applied to the vehicle further behind.

**Collision:** each vehicle has a set of four points located at their corners. When any of these points reaches a minimum distance $d_c$ from the same points of other vehicle located in the same road section and lane, it is assumed that they collided. The vehicle which collided is set back to the position previous to the collision.

**Terrain Following:** by calculating the lane direction vector and the percent of the road section extension traversed by the vehicle, the simulation updates the vehicle height and orientation to make it follow acclivities, viaducts or bridges.

### 4.2.7. Optimizations

In order to optimize the graphical processing of the 3D scene, LODs functionality was applied to vehicles and 3D objects. Six LODs were subjectively modelled for vehicles. During tests, each level was coloured differently so that changes between them were perceptible. No source of light was created as a means to reduce graphical processing.

The Euler model was used to adjust vehicle position and orientation. Due to the inaccuracy of the model according to the time interval passed, for intervals smaller than 0.04 seconds, which means a frame rate lower than 25-frames-per-second (fps), the value 0.04 was maintained and passed to the simulation model instead of the real time interval. On the one hand, this setting caused delay in the simulation when frame rates were below 25 fps. On the other, the vehicles movements became consistent independently of the frame rate value.

### 4.3. Interface

Two sub-modules compose ITranS interface: the first shows 2D simulation information and the second presents the 3D world in which simulation objects are drawn. All information is updated in real-time. The 2D interface is illustrated in Figure 4.

![Figure 4. ITranS 2D interface.](image)

The 3D world introduces the user in the first person, but the user does not possess any graphical representation. Two user views are presented in Figure 5 below.

![Figure 5. Two user views for an automatically and a manually generated traffic graph respectively.](image)
The first is a view of an automatically generated graph and the second of a manually generated one, whose model was based on the cartographic information and on-field transit flow measurements of the Complexo de Salgadinho, a coastal region located between Olinda and Recife, in the state of Pernambuco, Brazil. In the second view, 2D interface information was removed in order to improve the visibility quality of the image.

In addition, the user has the power to store viewpoints for future use during traffic analysis. These viewpoints consist of specific orientations and positions of the user. Besides the sequential navigation in the list of these stored viewpoints, the user is able to remove or view the last visited viewpoint. He can also directly remove or view the twelve first stored viewpoints in the list by the use of hot keys.

5. Performance Evaluation

Two experiments were carried out to evaluate simulation performance according to certain implementation aspects such as LODs and traffic graph structure. These experiments were run in a PC with a 2.2 GHz processor, 1GB of RAM, a graphical acceleration board GeForce FX 5200 © 128MB and the Windows XP © operational system. The measuring parameters were memory consumption and frame rate.

All tests only manipulated each variable at a time and were all based on automatically generated traffic graphs. Measurements were initiated only after the traffic flow was uniformly distributed in all road sections, which would take five to ten minutes after start. The maximum number of vehicles available was five hundred for all tests. Frame rates information was measured in 0.5s intervals for five simulation minutes. It implied in six hundred samples taken in approximately five or more minutes. Terrain following behaviour was not applied during tests.

5.1. Vehicle LODs Performance Test

Vehicle LODs influence on the simulation performance was the first aspect analysed in tests. Its measurement parameter was the average frame rates of a total of seven experiments and the value varied was the vehicles LOD used. For the first six experiments, vehicles had only one different LOD activated per experiment. User was always placed in the same position.

In the seventh experiment, vehicles had all LODs activated. To span the possible variations in LOD disposition in the traffic graph according to user position, this experiment was, in fact, composed of three tests with the user in three different views. The average of the frame rates of these three tests was calculated as a result. The traffic graph configurations for all the seven experiments were the same and can be seen in Table 1. N and M are the dimensions of the graph grid, d is the distance between adjacent junctions, f is the flow of vehicles per minute in each entrance connection and nl is the number of lanes per connection.

<table>
<thead>
<tr>
<th>N</th>
<th>M</th>
<th>D</th>
<th>f</th>
<th>nl</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4</td>
<td>200</td>
<td>60</td>
<td>5</td>
</tr>
</tbody>
</table>
5.2. Junction Number Performance Analysis

The second experiment consisted in measuring the performance effect caused by modifying either the number of nodes or their organization in the traffic graph. Seven tests were carried as specified in Table 2. Only LOD 3 was applied to all vehicles and the user was always placed in the centre of the traffic graph.

Table 2. Traffic graphs settings for junction number performance analysis.

<table>
<thead>
<tr>
<th>Test</th>
<th>N</th>
<th>M</th>
<th>Wait time</th>
<th>d</th>
<th>f</th>
<th>nl</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>8</td>
<td>8</td>
<td>5</td>
<td>100</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>16</td>
<td>16</td>
<td>10</td>
<td>100</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>16</td>
<td>5</td>
<td>100</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>32</td>
<td>10</td>
<td>100</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>48</td>
<td>5</td>
<td>100</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>24</td>
<td>5</td>
<td>100</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>12</td>
<td>12</td>
<td>5</td>
<td>100</td>
<td>5</td>
<td>3</td>
</tr>
</tbody>
</table>

6. Results

The results of the first experiment are presented in Figure 6a. It is perceptible the decrease in frame rate according to LOD increase and the gain due to the application of LODs. The use of LODs allows the visualization of vehicle model details according to distance and resulted in an average frame rate compared to the first six experiments.

For the second experiment, the results are shown in Figure 6b and Figure 6c. By comparing the results of the graphs in the pairs 0-2 and 1-3, little frame rate variation is noticed. This indicates the fact that traffic graph conformation has little influence in the simulation performance. This result is also confirmed for experiments 4, 5 and 6.

Nevertheless, memory consumption increase is noticed when comparing the graphs of the pairs of experiments 4-5, 5-6, 2-0 and 3-1. The first graph of each pair, by having less internal edges, consumes less memory than the second. The first graphs also have more entrance connections, which should imply in larger traffic volume per minute and in larger volume of vehicles simultaneously moving. However, due to the narrowing of the traffic graph, the vehicles time of permanence has decreased, which justifies the frame rate not having been affected by the traffic graph conformation change.

Figure 6. Frame rates results for the first (a) and second (b, c) experiment.
7. Conclusions and Future Work

Based on the experiment results and the visual results achieved with the modeling of the Complexo de Salgadinho, the possibility of efficiently simulating urban traffic in three-dimensional environments became evident. The new behaviours have worked successfully and resulted in rather realistic vehicle movements.

Despite the advantages in the addition of one extra dimension, care must be taken during implementation. Distributed processing, efficient algorithms and rendering techniques are fundamental to maintain high performance for large environments. By inserting a communication protocol and adjusting some of the simulation structures, the presented architecture seems to be capable of easily adapting to a distributed model. Once the simulation is distributed, its user interaction via portable computers, such as PDAs and cell-phones, is a foreseeable scenario.

7.1. Future work

Despite the achievements of this project, much still must be done in order to allow it to simulate a real urban traffic situation.

First of all, vehicle behaviours need to be improved in order not only to support more complex situations, but also to give more precision to drivers movements. Psychological features, together with public and heavy transport simulation, pedestrians and climatic parameters, will provide even more realistic transit behaviour. Lane changing must be less restricted and vehicles road signs must be considered.

Secondly, simulation distribution is necessary to enhance performance and extend the scope of the simulation. Only with such an improvement will large urban environments be simulated in real-time.

In third place, the insertion of realistic topographic models will guide the user in the traffic graph analysis and manipulation. For the roads to automatically follow the terrain surface, new algorithms will also need implementation.

Fourthly, the use of levels of detail could be auto-adjustable according to user system performance. Slow machines would apply only low levels of detail to their entities whereas powerful machines would render entities with more detailed LODs.

Lastly, ITranS must be further validated with the construction of models from other urban regions, based on a more extensive database of traffic measurements in different days and hours, or even months.

To summarize, it is believed that, with such changes implemented, the simulation of more complex and realistic traffic situation will be efficiently achieved. A demo video, as well as other pictures of ITranS project may be found at: http://www.gprt.ufpe.br/~grvm/itrans/.

8. Acknowledgements

We would like to thank CAPES for funding this research project. We would also like to address special thanks to Rodrigo C. Farias and Dinaldo A. Pessoa for their contribution to the article with the modeling of the Complexo de Salgadinho.
9. References


