Microscopic traffic simulation with reactive driving agents

Patrick A.M.Ehlert and Leon J.M.Rothkrantz, Delft University of Technology

Abstract -- **Computer traffic simulation is important for making new traffic-control strategies. Microscopic traffic simulators can model traffic flow in a realistic manner and are ideal for agent-based vehicle control. In this paper we describe a model of a reactive agent that is used to control a simulated vehicle. The agent is capable of tactical-level driving and has different driving styles. To ensure fast reaction times, the agent's driving task is divided in several competing and reactive behavior rules. The agent is implemented in and tested with a prototype traffic simulator program. The simulator consists of an urban environment with multi-lane roads, intersections, traffic lights, and vehicles. Every vehicle is controlled by a driving agent and all agents have individual behavior settings. Preliminary experiments have shown that the agents exhibit human-like behavior ranging from slow and careful to fast and aggressive driving behavior.**

Index terms – **artificial intelligence, traffic simulation, multi-agent systems, driving behavior**

I. INTRODUCTION

In the last two decades, traffic congestion has been a problem in many countries. To reduce congestion, most governments have invested in improving their infrastructure and are exploring new traffic-control strategies. A problem is that infrastructure improvements are very costly and each modification must be carefully evaluated for its impact on the traffic flow. Computer traffic simulations form a cost-effective method for making those evaluations. In addition, traffic simulations can evaluate the improvements not only under normal circumstances, but also in hypothetical situations that would be difficult to create in the real world. Obviously, the used simulation model needs to be accurate in modeling the circumstances and in predicting the results.

Intelligent agents, which are smart autonomous computer programs, can be used to simulate the driving behavior of individual drivers. The adaptability and flexibility of an intelligent agent make it possible to control various types of vehicles with different driving styles. Each agent can be equipped with its own behavior settings to simulate personalized driving behavior. This way, the simulated vehicles will behave realistically and the interaction between multiple drivers can be studied.

This paper describes a model of a reactive agent that can perform tactical-level driving. Tactical-level driving consists of all driving manoeuvres that are selected to achieve short-term objectives. Based on the current situation and certain pre-determined goals, the agent continuously makes control decisions in order to keep its vehicle on the road and reach its desired destination safely.

II. MICROSCOPIC TRAFFIC SIMULATORS

Many traffic simulators that are used today are macroscopic simulators. Macroscopic simulators use mathematical models that describe the flow of all vehicles. These models are often derived from fluid dynamics and treat every vehicle the same. Only the more advanced models can differentiate between vehicle types (e.g. cars, trucks, and busses) and even then all vehicles are treated equally within one vehicle type.

In real life many different types of vehicles are driven by different kind of people, each with their own driving style, thus making traffic flow rather unpredictable. In microscopic simulations, also called micro-simulations, each element is modeled separately, allowing it to interact locally with other elements. For example, every simulated vehicle can be seen as an individual with the resulting traffic flow being the emergent behavior of the simulation.

A Multi-Agent System (MAS) [1] can be used to form the basis of a microscopic traffic simulator. The main components (agents) of a MAS traffic simulator will be the vehicles. Every vehicle can be controlled by an individual agent. Other important elements of the simulator can also be modeled as agents, for example a traffic-light agent that controls a group of traffic lights. In 1992, Frank Bomarius published a report on such a MAS [2]. His idea was simply to model all the used objects as agents that could communicate the relevant data. Four years later two MSc. students at the University of Edinburgh implemented this idea for their final MSc. project [3],[4]. Their nameless text-based simulator uses Shoham's AGENT-0 architecture [5] to create multiple agents that function as vehicles or traffic lights, but also as roads and intersections. As the emphasis of their project was on creating a MAS-simulation and not necessarily creating realistic driving behavior, all their vehicle agents use very simple rules based on gap acceptance and speed. More advanced behaviors like overtaking cannot be modeled due to the simplicity of both their agent and simulation environment.

A more advanced simulation environment is the SHIVA simulator, which stands for Simulated Highways for Intelligent Vehicle Algorithms [6]. The SHIVA simulator was especially designed to test tactical-level driving algorithms and allows fast creation of different test scenarios. In his PhD thesis Rahul Sukthankar described a reasoning system for tactical driving called

Both authors are with the Mediamatics / Knowledge Based Systems Group, Department of Information Technology and Systems, Delft University of Technology, 2628 CD Delft, The Netherlands
Email: P.A.M.Ehlert@its.tudelft.nl or L.J.M.Rothkrantz@cs.tudelft.nl

0-7803-7194-1/01/$10.00 2001 IEEE

POLYSAPIENT and his use of SHIVA to test his system [7]. A drawback of the SHIVA simulator is that it needs a special SGI machine to run and is not publicly available.
At first glance, the approach we used with our driving agent resembles the POLYSAPIENT reasoning system used by Sukthankar, but its implementation is quite different. First of all, our simulator implements an urban environment. SHIVA, and most other traffic simulators, models highway or freeway traffic. Second, with our agent multiple behavior parameters can be set to produce the desired driving behavior. Most other simulators only use one or two driving-behavior parameters (usually aggression or gap acceptance and preferred speed) or none at all. Third, by using relatively independent behavior rules our agent's functionality can be expanded or altered easily and the agent can be used in completely different environments.

III. TRADITIONAL VERSUS REACTIVE AGENTS

An intelligent agent is an autonomous computerized entity that is capable of sensing its environment and act intelligently based on its perception. Traditional agent architectures applied in artificial intelligence use sensor information to create a world model [8],[9]. The world model is processed by standard search-based techniques, and a plan is constructed for the agent to achieve its goal. This plan is then executed as a series of actions. The traditional approach has several drawbacks. Sensor constraints and uncertainties cause the world model to be incomplete or possibly even incorrect, and most traditional planning methods cannot function under noisy and uncertain conditions. Furthermore, in complex domains like tactical driving it is infeasible to plan a complete path from the initial state to the goal state, due to the large amount of searchable states and the inability to perfectly predict the outcome of all possible actions. As a result a real-time response cannot be guaranteed, making the traditional planning methods unsuitable for tactical driving.
Reactive agents, also called reflex or behavior-based agents, are inspired by the research done in robotic control. Their primary inspiration sources are Rodney Brooks' subsumption architecture [10] and behavior-based robotics [11]. Reactive agents use stimulus-response rules to react to the current state of the environment that is perceived through their sensors. Pure reactive agents have no representation or symbolic model of their environment and are incapable of foreseeing what is going to happen. The main advantage of reactive agents is that they are robust and have a fast response time, but the fact that pure reactive agents do not have any memory is a severe limitation. This is the reason that most reactive agents use non-reactive enhancements.

IV. DRIVING AGENT MODEL

We have designed a model of a reactive driving agent that can control a simulated vehicle. The agent is designed to perform tactical-level driving and needs to decide in real-time what manoeuvres to perform in every situation. These decisions are based on the received input from the agent's sensors. After the agent reaches a decision, the instructions are translated into control operations that are sent to the vehicle.
The driving agent is modular in design. Every part can be adapted, replaced, or otherwise improved without directly affecting other modules. The used parts are: several sensors to perceive the agent's environment, a communication module, a memory and controller for storing data and regulating access to the memory, a short-term planner, multiple behavior rules and behavior parameters, and an arbiter for selecting the best action proposed by the behavior rules. A picture of the agent's layout is shown in Figure 1.

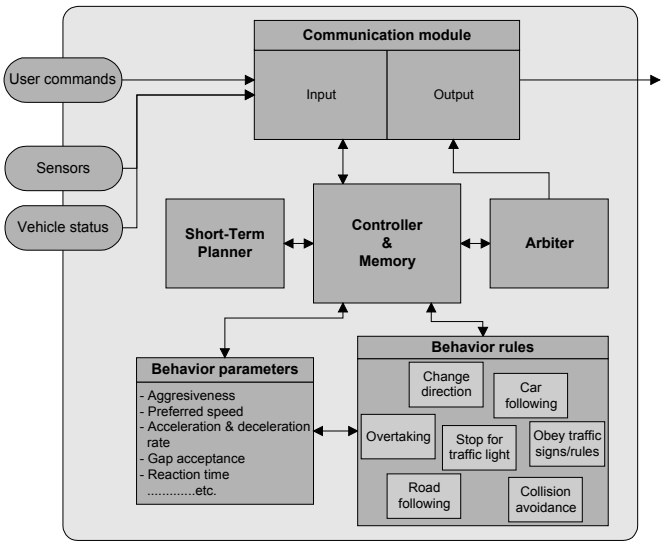

Figure 1: Driving agent layout

Our agent uses both traditional and reactive methods to perform its task, but the emphasis is on the latter since fast response times are important. Sensor information is stored in the memory and forms a temporary world model. Reactive procedures called behavior rules or behaviors use the available information in the memory to quickly generate multiple proposals to perform a particular action. Planning in the traditional sense is not applied. The short-term planner only uses simple linear extrapolation to calculate the expected positions of moving objects and the arbiter determines the best action based on the priority ratings of the action proposals included by the behavior rules.

A. Reasoning

The complete loop from receiving sensor messages to sending an output message to the vehicle can be seen as one reasoning cycle. The timing of a reasoning cycle and the activation of the agent's parts are done by the controller that also regulates the access to the memory. Since we want the driving agent to react in at least real-time, the agent is able to complete several reasoning cycles per second. The activation of the agent's parts is shown in Figure 2.

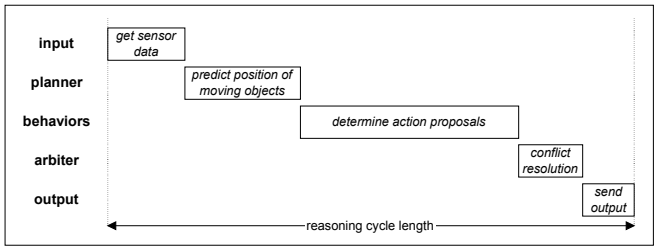

Figure 2: The reasoning cycle regulated by the agent's controller

The agent uses two types of sensor information. The first type gives information about the agent's environment, for example the distance and angle to objects, or the width of the agent's current lane. The second sensor type returns information about the vehicle that is controlled by the agent. This includes speed, acceleration, heading, wheel angle, and fuel level. In addition, the agent can receive orders from the user.

All information that is sent to the agent is received by its communication module that contains knowledge of the used communication protocols. When a message is received, the communication module tries to recognize the message format, the sender, and its content. When the message is ok, the input section of the communication module temporarily stores it until all received messages can be written to the agent's memory. Temporary storage is necessary since one does not want data in the memory to be read and written at the same time. Outgoing messages can be sent immediately since no conflicts can arise there.

Next, all incoming messages are transferred to the agent's memory and the short-term planner makes a fast prediction of the position of all moving objects perceived in the environment. The actual reasoning of the agent is performed by the behavior rules, also called behaviors. They specify what to do in different situations. Based on the available data in the agent's memory, every behavior can propose an action. All action proposals have a tally or priority rating. The arbiter selects the best proposal based on the priority ratings and sends it to the communication module. Finally, the communication module translates the proposal to control instructions that can be understood by the vehicle.

B. Behavior rules

The agent's driving task is divided into several subtasks that are automated by independent behavior rules. This way the agent's functionality can be expanded easily without any modifications to the existing behaviors. The used behavior rules are very much dependent of the agent's environment. We have chosen to let the agent drive in an urban environment. The reason for this is that an urban environment is one of the most difficult and complex traffic scenarios. In a city, many unexpected events can happen and the agent has to deal with many different situations. This way we can show the potential of our driving agent concept. Note that the design of our agent does allow driving in other environments. Only the agent's behavior rules might need to be adapted or expanded. For the city environment we designed the following behaviors:

1) Road following

The road-following behavior is responsible for keeping the agent driving on the road. Besides controlling the lateral position of the agent's vehicle, based on the distance to the road and lane edges, the road-following behavior also influences the agent's speed. It makes sure that it slows down for curves and on straight roads it will accelerate until the desired speed set in the behavior parameters is reached.

2) Intersection / changing directions

If the agent is approaching an intersection, its speed is reduced, precedence rules are applied, and the agent will choose one of the side roads. The changing-directions behavior can be split up into several sub-behaviors, one for each type of intersection. This is consistent with the fact that humans use different strategies to handle different types of intersections.

3) Traffic lights

The traffic-lights behavior makes sure that the agent stops for red or yellow traffic lights if possible. The behavior checks if the sensed traffic light regulates the agent's current lane and slows down the vehicle. The agent's braking start-point depends on its preferred braking pressure (deceleration rate) and is set in the behavior parameters.

4) Car following

The car-following behavior ensures that the agent does not bump into any other vehicle. If another car is driving in front of the agent, speed is reduced to match that car's speed. The precise braking pressure depends on the speed difference between the agent's vehicle and the other vehicle, the distance between them, and the set gap acceptance of the agent.

5) Switching lanes and overtaking

Related to the car-following behavior is the switching-lanes-and-overtaking behavior. If a slower vehicle is in front of the agent, it may decide to overtake this vehicle. This decision depends on the velocity difference between the two vehicles and the available space to overtake the vehicle, both in front and to the left of the other vehicle.

6) Applying other traffic rules

Besides traffic lights and precedence rules at junctions, other traffic rules need to be followed. Examples are, not driving at speeds above the local maximum, driving on the right side of the road as much as possible (in the Netherlands), and no turning in one-way streets. Only aggressive drivers have a tendency to break some of those rules.

For this behavior, it is necessary to keep track of the traffic signs and restrictions encountered by the agent. Because the memory of the agent will clear data on a regular basis to save space, the traffic-rules behavior needs to keep track of these signs itself, in its own private memory space. This memory space is embedded within the behavior. Note that the behavior also needs to keep track when the signs and rules apply. Usually, turning onto a new road will reset most of the current restrictions.

7) Collision detection and emergency braking

The collision-detection and emergency-braking behavior is a special kind of safety measure that is activated when the agent is on a collision course with an object. At all times the behavior needs to ensure that the vehicle can be halted before it hits the object. Actions from the emergency-braking behavior have the highest priority and always overrule all other behaviors.

C. Behavior parameters

In order to create different driving styles all behavior rules are influenced by behavior parameters. One of the most important (visible) factors is the driver's choice of speed. This choice has a large effect on the specified behaviors. Drivers that prefer high speeds are more likely to overtake other vehicles than slower drivers. Another factor is the distance the driver keeps to other cars, also called gap acceptance. Aggressive drivers keep smaller gaps than less aggressive drivers. A third parameter is the driver's preferred rate of acceleration or deceleration. Again, aggressive drivers tend to accelerate faster than less aggressive drivers.

Besides the above-mentioned behavior factors, other aspects can influence an agent's driving behavior, for example the reaction time of an agent and the range of its sensors. An agent's reaction time can be altered by changing the length of its reasoning cycle. The sensor range determines the visibility of the agent and can be used to simulate fog or bad weather conditions.

V. IMPLEMENTATION

We have constructed a prototype traffic simulator program to test our driving agent design. The programming language we used to build the simulator is Borland Delphi 5 Professional for NT. We have chosen this language in part since we were already familiar with it, but mainly because Delphi is an easy language, very suitable for quick prototyping.

Our simulator uses a kinematic motion model that deals with all aspects of motion apart from considerations of mass and force. The model implements smooth motion of vehicles, even during lane changes. Furthermore, the vehicles can move along realistic trajectories, but since forces are not modeled, the vehicles will perform manoeuvres without slipping.

A. The prototype simulator

The simulator program roughly consists of four elements: a user interface to provide visual feedback, a simulation controller, an environment containing simulated objects, and the driving agent model. The task of the simulation controller is to start, pause or stop a simulation run and keep track of the elapsed time. The simulation controller also initializes, starts, and stops the driving agents. During simulation, the controller regularly sends an 'update' order to the environment. The environment then calculates new values for all its objects and sends relevant visual feedback to the screen. This 'simulation update' loop is shown in the left part of Figure 3. By default the update frequency is about 20 times per second, but this rate can be adjusted so that the program can run on slower computers.

The environment is formed by all the simulated objects together. Different environments can be loaded via Map Data Files. These files contain a description of a road network and traffic-control systems. Our current simulator implementation contains multi-lane roads, intersections, traffic lights, traffic-light controllers, and vehicles.

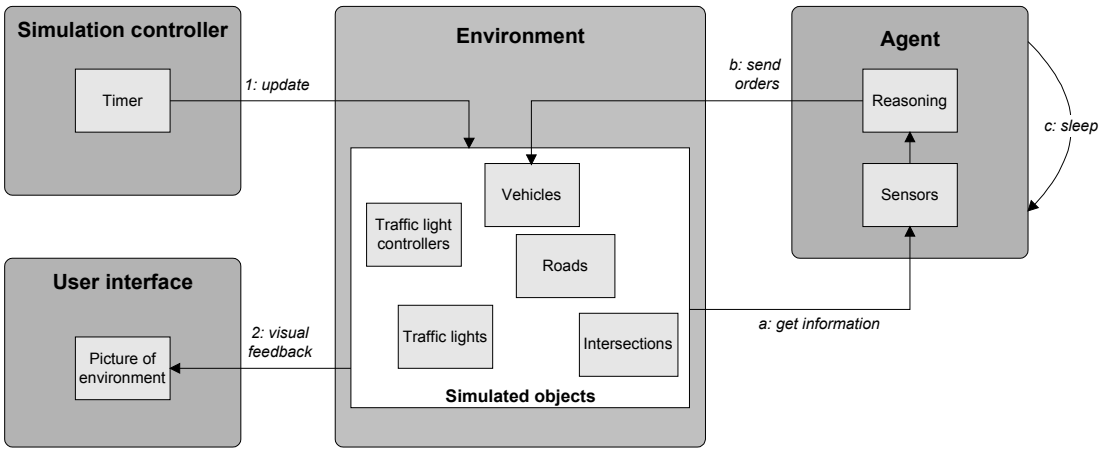

Figure 3: Simulation and agent update loop

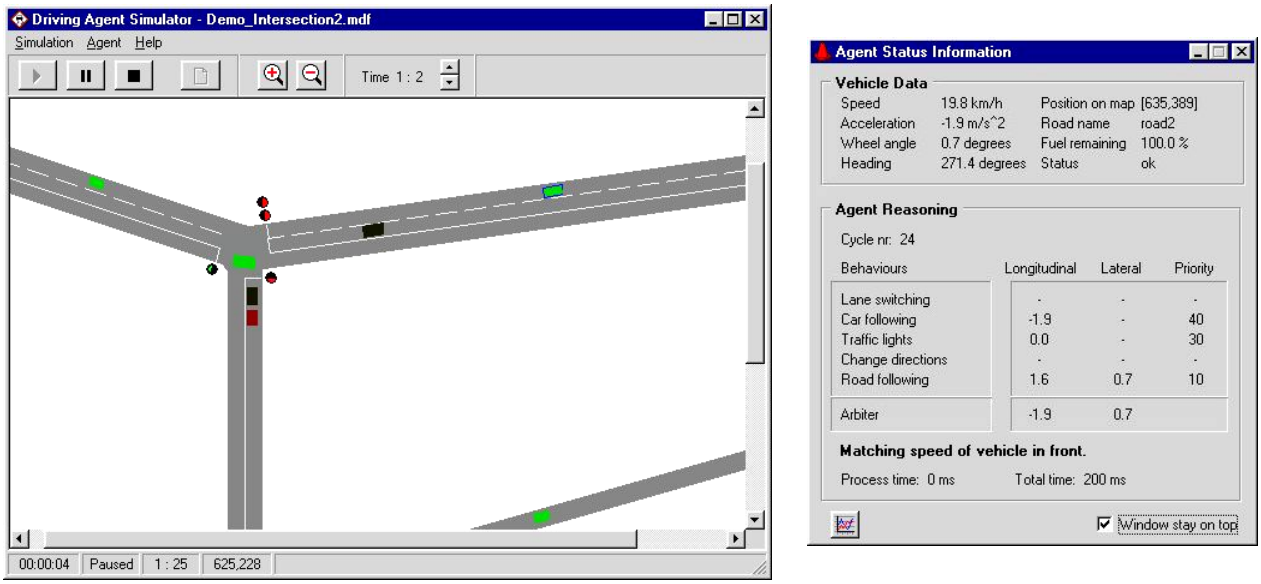

Figure 4: Screen shot of the prototype simulator used to test the driving agent model

A. The driving agent

Every vehicle in the environment has its own driving agent, but there is one agent that has the focus of attention and can be 'controlled' by the user. This means that the user can change the settings of this agent's behavior parameters and can follow its reasoning process in the Agent Status Information window shown in Figure 4.

All agents are implemented as threads, which are lightweight processes, and are started by the simulation program. The advantage of using threads is that the simulation can be faster, running threads in parallel (if the operating system allows it), and that the agents can run independent of the simulation program. The disadvantage is that there is a limit to the number of threads one can use, because the overhead in managing multiple threads can impact the program's performance. The execution loop of an agent is shown in the right part of Figure 3. If the agent finishes a reasoning cycle its thread is put asleep for a while. This is done to prevent agents from using all available CPU time. By default an agent's cycle time is 200 ms, so the agents will perform 5 reasoning cycles per second.

The agent's behavior rules are implemented as if-then rules. All behaviors are divided into several tasks. Tasks are executed in a serial manner, the least important task first and the most important task last. This way the important tasks 'override' the action proposals of less important tasks. The execution of the behavior rules is also done consecutively, but in this case the execution order does not matter since the arbiter will wait until all behaviors are finished determining their action proposal.

VI. RESULTS AND DISCUSSION

We have presented a model of a reactive driving agent that can be used to control vehicles in a microscopic traffic simulator. A prototype simulation program was constructed to test our agent design. Although we have not validated the used parameters yet, preliminary experiments have shown that the implemented agent exhibits human-like driving behavior ranging from slow and careful to fast and aggressive driving behavior. The experiments were done using the first five behavior rules discussed earlier in the "behavior rules" section. Here, we briefly discus the results of one of our experiments. The experiment consists of two different drivers approaching an intersection and stopping in front of a red traffic light. Both drivers perform this task without any other traffic present. The first driver is a careful driver with a low preferred speed, reasonably large gap acceptance, and a low preferred rate of deceleration. We call this driver the 'grandpa' driver. The second driver is a young and aggressive driver, with a high preferred speed, small gap acceptance, and a high preferred rate of deceleration. The drivers start at the same distance from the intersection. The speed of both vehicles during the experiment is shown in Figure 5.

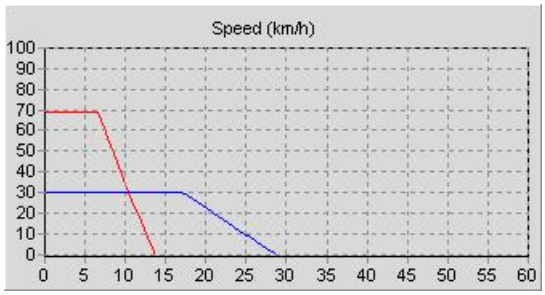

Figure 5: Speed of the grandpa driver (blue) and young aggressive driver (red) during the experiment

Since the grandpa driver is driving at a lower speed, it takes a while before he starts braking, but his braking start-point (50m) is closer to the intersection than that of the young aggressive driver (65m), due to his lower speed. The difference between the used braking pressures is clearly visible. Both drivers brake with a relatively stable deceleration (approximately 0.7 m/s^2 and 2.7 m/s^2), which is consistent with human braking behavior.

The experiment was done several times, but in almost all cases the shown graphs were roughly the same. In addition, the precise stopping positions of both vehicles were approximately the same in all experiments. The young aggressive driver had a tendency to brake relatively late and often came to a stop just in front or on the stopping line. The grandpa driver on the other hand always came to a full stop well ahead of the stopping line. The stopping positions of both vehicles during one of the experiments are compared in Figure 6.

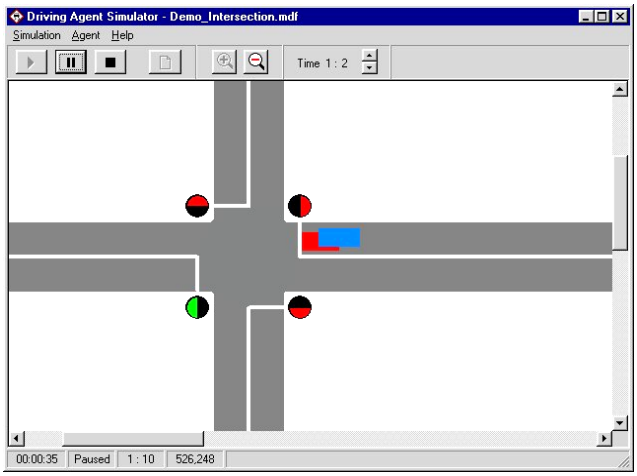

Figure 6: Compared stopping positions of the young aggressive driver (red) and grandpa driver (blue)

The aim of our simulation program was to test the design and functionality of our driving agent, but as a result its current implementation is rather inefficient since we did not optimize it for speed. Our focus was on the correctness of the agent's driving behavior and reasoning process. The computer used to implement and test the program is an Intel Pentium III, 450 MHz with 64 MB of RAM, running Microsoft NT 4.00. On this computer we were able to run experiments with up to 40 vehicles (agents). Experiments with more vehicles are possible, but result in a slow-running simulation. Currently, we are working on improving the simulator's memory management and processing speed to be able to use more agents.

VII. CONCLUSIONS AND FUTURE WORK

The main advantage of agent-based microscopic traffic simulation over the more traditional macroscopic simulation is that it is more realistic. Instead of using general traffic-flow models, traffic becomes an emergent property of the interaction between agents. Another advantage is that agent-based simulation is more flexible. Changes to traffic scenarios can be made quickly by altering the position of individual vehicles and changing an agent's parameters. A disadvantage is the increase of computational resources and the higher number of parameters that need to be set and validated.

Preliminary experiments have shown that our driving agent exhibits human-like driving behavior and is capable of modeling different driving styles, ranging from slow and careful to fast and aggressive driving behavior.

At the moment we are experimenting with different types of agents in several scenarios. Our goal is to study the possibilities of the traffic simulator and agent in order to improve them further. The simulation environment can be made more realistic by adding new objects, such as busses, trucks, emergency vehicles, pedestrian crossings, traffic signs, trees and buildings. Once the simulator is improved with the new objects, the agent's functionality must be extended to deal with these objects. In addition, the simulation environment needs to be validated. Although we have tried to use realistic values for vehicle acceleration, turn radius, road size etc., the used settings might prove to be inaccurate. We also need to study human driving behavior more extensively in order to validate our driving style models.

The drawback of adding new functionality will be that both the simulation environment and the agent will need more computation time and will run more slowly. Therefore, we are considering using a distributed approach in the future so that the driving agents can run on different computers. The simulation controller and environment can act as a server and the agents can be the clients communicating to the server.

VIII. REFERENCES

[1] Ferber, J. (1999) *Multi-agent systems: an introduction to distributed artificial intelligence*. Addison Wesley Longman Inc., New York.

[2] Bomarius, F. (1992) *A Multi-Agent Approach towards Modeling Urban Traffic Scenarios*. Research Report RR-92-47, Deutches Forschungszentrum für Künstliche Intelligenz, September 1992.

[3] Chan, S. (1996*) Multi-agent Traffic Simulation – Vehicle.* MSc dissertation, Department of Artificial Intelligence, University of Edinburgh.

[4] Chong, K.W. (1996) *Multi-Agent Traffic Simulation - Street, Junction and Traffic light.* MSc dissertation, Department of Artificial Intelligence, University of Edinburgh.

[5] Shoham, Y. (1993) "Agent-oriented programming". In *Artificial Intelligence 60*, pages 51-92.

[6] Sukthankar, R., Hancock, J., Pomerleau, D. Thorpe, C. (1996) "A Simulation and Design System for Tactical Driving Algorithms". In *Proceedings of artificial intelligence, simulation and planning in high autonomy systems*.

[7] Sukthankar, R. (1997) *Situation awareness for tactical driving.* PhD Thesis, Technical report CMU-RI-TR-97-08, Robotics institute, Carnegie Mellon University.

[8] Wittig, T. (1992) *ARCHON: an architecture for multi-agent systems.* Ellis Horwood Limited, England.

[9] Rao, A. S., and Georgeff, M. P. (1995) "BDI Agents: From Theory to Practice". In *Proceedings of the 1st International Conference on Multi-Agent Systems* (ICMAS-95), San Francisco, USA, June, pages 312-319.

[10] Brooks, R.A. (1986) "A robust layered control system for a mobile robot". MIT AI Lab Memo 864, September 1985. Also, in *IEEE Journal of Robotics and Automation* Vol. 2, No. 1, March 1986, pages 14-23.

[11] Arkin, R. C. (1998) *Behavior-based robotics.* The MIT Press, Cambridge, Massachusetts.