Autonomous Agents Navigating in Virtual Road Network

Eun-Sang Cho, Kwang-Jin Choi, Hyeongseok Ko
Graphics and VR Research Lab
School of Electrical Engineering
Seoul National University, Seoul, 151-742, KOREA.
{choeuns,kjchoi,ko}@graphics.snu.ac.kr

Abstract

In a virtual environment, agents must demonstrate some degree of realism and interactivity. This paper discusses the algorithm that enables agents to navigate a virtual road network realistically and interactively. The road network is modeled by a language invented by CGVRLab. in Seoul National University. The road description files written in this language provide the information of road environments to the navigating agents and the scene visualizer. We call this navigating agent in the road an *ambient car*. The ambient cars must follow the traffic rules as human does. To do this, the ambient car should continuously check its circumstances, such as, the traffic lights, lanes, road signs, and other ambient cars. Because of the huge scale of road network and the large number of ambient cars, the algorithm considers only the area where the participant is currently located. By this locality, the performance of the whole system does not fluctuate much in different situations. The behavior of ambient cars according to the predefined rules may appear monotonous. We added probability distribution functions to introduce some randomness. We implemented the above idea on Silicon Graphics Indigo 2 workstation. The ambient car exhibited its awareness of lanes, traffic lights, and other cars. The participants could hardly distinguish between a human-controlled car and the computer-controlled ambient car generated by the algorithm.

1 Introduction

In VR applications, most realism stems from the interaction between computer-generated agents and human. Therefore the computer generation of believable behaviors of autonomous agents is emerging as an important technique in many VR applications.

This paper presents the algorithms for generating ambient traffic in driving simulation. The goal of our driving simulation is not just a fast walkthrough on the empty roads, but we like to produce interesting events along the way, so that the participants get valuable lessons which can be very costly on the actual roads. One of the elements leading to the realization of such interesting events is the *ambient traffic*. i.e., we need some mechanism that produce realistic behaviors of the background traffic that interacts with the *simulation car* [1] – the car driven by the participant.

We are not implementing the ambient car with the conventional AI approaches such as symbolic reasoning or path planning. Such approaches have been proven particularly poor in modeling autonomous agents in a dynamic environment. Instead, we implement the behavior of the ambient cars with emergent functionalities [2] and a minimum amount of global control. The resulting behavior is robust, reactive, and can adapt to highly dynamic situations.

2 The System Overview

In our ambient traffic generation, each ambient car is modeled as an autonomous agent. It collects the information about the world: the road and lane it is currently running on, the type of the junction and traffic light ahead, the neighboring cars, etc. Based on the collected information, it decides where to move at the next frame time.

The autonomous agents with various shapes

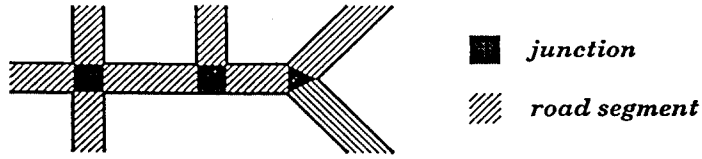

Figure 1: The Junctions and Road Segments

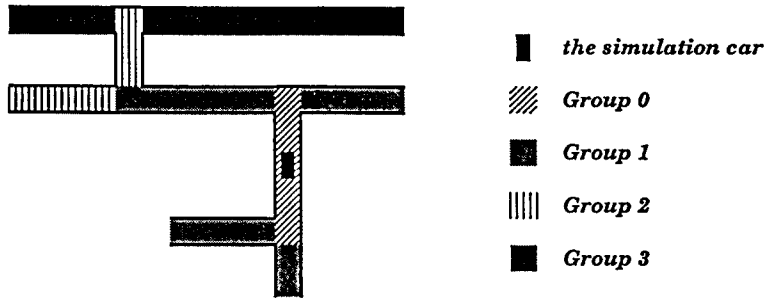

Figure 2: Classification of Road Segments

and behavioral characteristics are placed on the road. Instead of scattering an indefinite number of cars all over the roads, we maintain a fixed number of cars only around the simulation car. This set of cars are called the *active set*, and the cars in the active set are called the *active cars*. Of course, the active set should be maintained dynamically. To manage the active set, we construct *ambient traffic manager* which is the manager of the ambient car.

As shown in Figure 1, a *junction* is the region where three or more roads meet. A *road segment* (or simply a *road* or *segment* if it is clear from the context) is defined as the part of the road between two junctions.

We classify the road segments into four groups (Figure 2). The classification is based on how close the segment is to the current location of the simulation car. Thus, Group 0 consists of the single road segment which the simulation car is currently running on. Group 1 consists of the roads adjacent to the Group 0 road. Group 2, in turn, is the roads adjacent to the Group 1 roads, excluding the Group 0 road. Finally, Group 3 consists of the remaining roads. As shown in Figure 2, we also classify the junctions into the above Groups.

We call the collection of the roads in Groups 0, 1, and 2 the *active area*, and only the active area is considered in the ambient traffic simulation. Also, the visualization is done only for the active area and the active cars running in the active area. Group 2 roads are drawn in much less detail than Group 0 or 1 roads.

Each active car regularly checks if it has moved to a Group 3 road. If it has, the manager removes the car from the active set. The details are presented in the Section 3.1.

Once the active set of cars are determined, the next job is to move the cars along the roads. the decisions a car can make are changing lanes, left turn, going straight, right turn, or U-turn. Car does not have a specific destination, we can make the above decision based on some probability distribution. The details are presented in the next section.

3 The Algorithm

The temporal and spatial pattern of the ambient car navigation should look like that of a human-driven car. The ambient car should change the lanes to prepare for the left or right turns, or reduce its speed if a sharp curve or an intersection is ahead. Also, the diversity of cars and good performance of the whole system are needed.

3.1 Managing the Ambient Cars

The ambient traffic manager maintains the active set of ambient cars. The active set is the collection of the cars running in the active area.

Before drawing a new frame, the manager updates the contents of Groups 0, 1, 2, and 3, based on the new location of the simulation car.

Then, each ambient car finds out whether it is still in the active area according to the updated groups. If it is not, that car is deleted from the

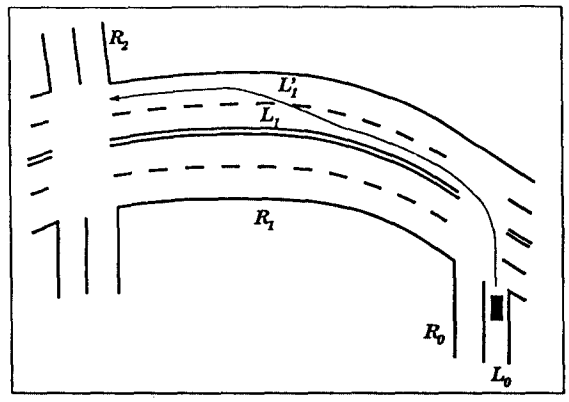

Figure 3: The Trajectory Computation

active set. Even within the active area, the car needs to know in which group of roads it is located for the visualizer to determine at which level of detail should the car be drawn.

Whenever an active car is removed, a new car should be placed into the road. To avoid a sudden pop-up, the new car is always placed on a road randomly chosen from Group 2.

3.2 The Trajectories of the Ambient Cars

When an ambient car approaches a junction, it should decide whether to proceed forward, left, right, or make a U-turn. Because the ambient car does not have any specific destination, the choice among the four directions can be made arbitrarily.

Referring to Figure 3, suppose a car is about to enter into a new road R_1 from the lane L_0 of the old road R_0. The decision for the new road segment is made one step ahead. Therefore, at this point, R_1 had been determined already when the car entered into R_0. Just before the car enters into R_1, we have to decide the lane L_1 of R_1 into which the car enters, the road R_2 to be taken at the end of R_1, and the lane L_1' which the car should be in to prepare for the next move into R_2.

Once all the above elements are determined, then a spline technique generates the actual path from L_0 to the end of L_1'.

The trajectory computation is a relatively expensive operation. But, the overhead of the trajectory computation is well distributed to the entire time duration.

3.3 Simulating the Diversity

Ambient cars can have different characteristics in two different aspects: the shape and behavior. The *carelessness factor* is used to control the degree of carelessness of an ambient car. If it has a large value, the ambient car tends to go faster, make more lane changes, and so forth. Whenever a car has to be made active, the above two parameters are randomly set.

3.4 Observing the Traffic Signals

An ambient car needs to obey one of the large number of traffic signal, i.e., the one nearest in the forward direction. We call such light the `current_traffic_light`. Usually there is a traffic light at the end of current road, and that is the one we are looking for. If there is no traffic light, however, `current_traffic_light` will have the null value and the junction is ignored.

In our implementation, each traffic light has its own timing diagram. Once the ambient car finds out the `current_traffic_light` and its information, the ambient car observes traffic signals by controlling its speed. If the `current_traffic_light` is red, the ambient car reduces its speed until it makes a complete stop in front of the light or at the back of the preceding car. If the light turns to green, it resumes to advance.

3.5 Interactions among the Cars

The collision avoidance is the most basic but important element that should be considered in simulating the interaction among the cars.

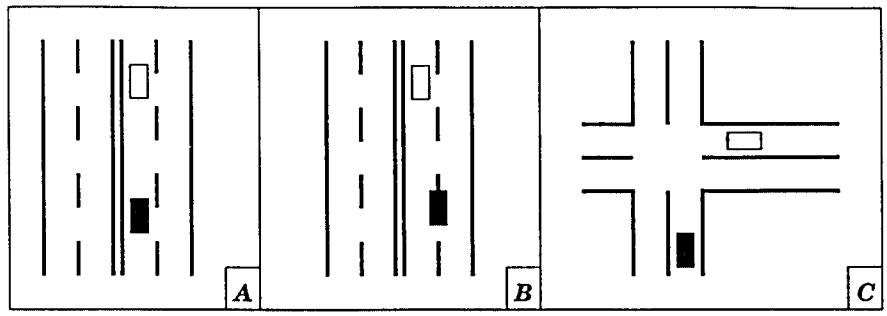

Figure 4: Computing the Influential Set. (The grey car represents C_i. All the white cars in the cases A, B, and C belong to the influential set of C_i.)

We implemented an simple algorithm which makes the ambient cars avoid collision and look realistic. It goes through each car C_i in the active set, and do the following two steps:

For each car C_j in the active set,[1] with $j \neq i$,

- Compute the distance between C_i and C_j. If the distance is greater than the threshold,[2] C_i is exempted from the collision avoidance procedure. If the distance is smaller than the threshold, do the following steps.
- We determine the *influential set* of C_i, which is the set of cars that can influence the movement of C_i. If C_j is in front of C_i in the same lane (Figure 4 (A)), C_j is put into the influential set. If C_i lies between two lanes, and C_j is in the front running in either one of the two lanes (Figure 4 (B)), then C_j is also put into the influential set. Suppose C_i and C_j are on different roads but approaching toward the same junction (Figure 4 (C)). If C_j will arrive sooner to the junction, and if there is no traffic light at the junction, then C_j is put into the influential set.

The ambient car checks its influential set and avoid collision by controlling its speed.

The above algorithm can involve $O(n^2)$ computation, where n is the number of active cars. However, most of the cars are far enough each other and thus returned from the beginning of the procedure as the trivial case, the amount of computation for the collision avoidance is normally much less than $O(n^2)$. Note that the above algorithm does not depend on the size of the road network.

4 Experiments

We implemented the above algorithms in C++ and OpenGL. The program runs on Silicon Graphics Indigo 2 workstation with an R4400 CPU and Maximum-Impact graphics board.

We ran our program at different conditions to measure the performance and realism of the algorithm.

4.1 Performance

We used a road network that consists of 152 road segments and 108 junctions. Figure 5 shows a snapshot from the simulation.[3] The total length of all the road segments was 32.4 kilometers (19.9 miles). About the half of the segments were curved and the other half were straight. The network included 324 traffic lights, 57 road signs. The total number of triangles for the entire network was about 750,000. Due to the localization, only about 35,000 triangles had to be drawn at each frame.

With the active set of 20 cars in the fully textured scene, the program could generate frames at 18~24Hz depending on the scene complexity.[4]

When the visualization of the cars are turned off, we can simulate the ambient cars more than 70 without overload of CPU. So, the graphics pipeline, not CPU, is the bottleneck of the simulation.

[1] C_j also ranges over the simulation car and the scenario cars.

[2] We use a scaled threshold. A constant threshold is scaled by the relative speed between C_i and C_j.

[3] The textures for the sky and trees are borrowed from the *Performer Town* [3].

[4] Each car consists of about 100 textured triangles. As the simulation car navigates, the number of cars and buildings appearing in the viewport fluctuated.

Figure 5: A Snapshot from the Simulation (with 20 Active Cars)

The complexity of the road network doesn't affect the whole performance of the simulation. If the number of cars remains constant, even when a huge road network is simulated, the uniform performance is guaranteed.

4.2 Realism

We organized a test to measure the realism of our algorithm. In the test, a person was asked to sit on the simulator and navigate the road without producing any critical events. In the test, we invited several subjects to watch the road for a few minutes and pick the car which they think is human-controlled.

The subjects could pick the human-controlled car after a while. the scene was dynamically changing, therefore minor details were overlooked. So, the subjects can hardly distinguish between the computer-controlled ambient car and the human-controlled simulation car.

5 Conclusion and Future Work

In this paper, we presented the algorithms for generating ambient traffic that exhibit dynamic and realistic behaviors. The most striking feature of our algorithm is that the complexity does not depend on the size of the road network. Therefore a huge area can be used for the simulation without any extra cost. Experiments show that the behavior of the ambient cars is almost indistinguishable from the human-controlled cars.

In the future, the scenario control algorithm will be combined with the ambient traffic generation. The combination will produce more useful experience to the participants.

Acknowledgments

This research is partially supported by SsangYong Information & Communication Corp.; Korea Electronics and Telecommunications Research Institute; Systems Engineering Research Institute; Agency for Defense Development; Samsung Electronics.

References

[1] James Cremer, Joseph Kearney, and Hyeongseok Ko. Simulation and scenario support for virtual environments. *Computer & Graphics*, 20(2):199–206, April 1996. Special Issue on "Techniques for Virtual Environments".

[2] Pattie Maes. *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back.* In Pattie Maes, editor, *Designing Autonomous Agents*, pages 1–2. The MIT Press, 1990.

[3] John Rohlf and James Helman. IRIS performer: A high performance multiprocessing toolkit for real–Time 3D graphics. In Andrew Glassner, editor, *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series, pages 381–395. ACM SIGGRAPH, ACM Press, July 1994. ISBN 0-89791-667-0.