# Naughty Agents Can Be Helpful: Training Drivers to Handle Dangerous Situations in Virtual Reality

Yongwu Miao*, Ulrich Hoppe*, and Niels Pinkwart**

** Institute for Computer Science and Interactive Systems*
*The University of Duisburg-Essen, Germany*
*(miao, hoppe)@collide.info*

*** Human Computer Interaction Institute*
*Carnegie Mellon University, USA*
*nielsp@cs.cmu.edu*

## Abstract

*Recently many car driving simulators have been developed and used for educational purposes. They can provide special training opportunities that are not available frequently in reality. However, currently existing simulators often rely on predefined situations and scenarios. Training students to deal with spontaneously occurring dangerous traffic situations is difficult when using hard wired "emergency" situations that occur in a predictable manner. In this paper we present a new approach to provide special learning opportunities. We adopt intelligent agent technologies to develop the "problem creator", a pedagogical agent who can deliberately cause dangerous situations. Introducing the problem creator into our collaborative 3D virtual car driving environment makes it unpredictable when, where, and what type of abnormal situations the students will be confronted with. These irregularly occurring challenges model the reality of car driving well.*

## 1. Introduction

Car driving is a kind of high-performance task where the driver must continuously adjust his behavior to changes in his environment. Sometimes, a driver has to make decisions and take actions quickly to handle unfamiliar and even dangerous situations that occur spontaneously. Without any skill to react to such an urgent event once it occurs, the driver may cause (or at least be involved) in serious accidents and even pay his life. Therefore, training to handle (and intuitively anticipate) abnormal situations should be a very important part in car driving training programs. However, it is currently impossible or too expensive to set up such training situations in reality.

The advantages of simulators for training high-performance tasks like car driving have been pointed out in literature [5, 10]. Using a simulator, a trainee is able to practice the handling of potentially dangerous emergency situations in virtual reality. Because these "emergency scenarios" are usually hard coded or scripted [3, 11], a simulator can repeatedly present traffic situations to students, replay driving failures to learners, and repeat exercises in a training course until the trainee can handle them successfully. Yet, the problem is that if the same "emergency situation" occurs repeatedly, a trainee can easily predict what will happen and thus prepare for the situation. This is unrealistic, since in the reality of car driving emergency situations occur unpredictably.

In this paper, we present a new approach to train learners to handle abnormal situations. We employ the *problem creator*, a kind of pedagogical agent that can deliberately create abnormal situations within a collaborative 3D car driving simulation environment. When, where and in which order the abnormal situations will be created is unpredictable. In this paper, we will give an overview of our collaborative 3D car driving simulation. Then the design and implementation of a problem-creator will be described. We also discuss the usage of the problem-creator and compare related systems. Finally, we draw conclusions of our work and point out future directions.

## 2. A Collaborative Car Driving Simulation Environment

This section provides an overview of our collaborative 3D car driving simulation environment. In this learning environment, multiple users can drive cars in a shared virtual driving place. Thus, specific traffic situations can involve the cars of multiple

learners. A detailed description of the technical approach is available in [7, 8].

In order to support multiple users to virtually drive their cars in a shared driving place, we developed a real-time groupware application. Figure 1 shows the principles of information flow within our collaborative car driving simulation system, which employs a classical server/client architecture. We use the TSpace system [6], a tuple space based solution, to develop the server. The server maintains the current state of a driving place as a set of tuples, some of which are static while others are dynamic. Each client application, running on user's local machines, interacts with the remote space.

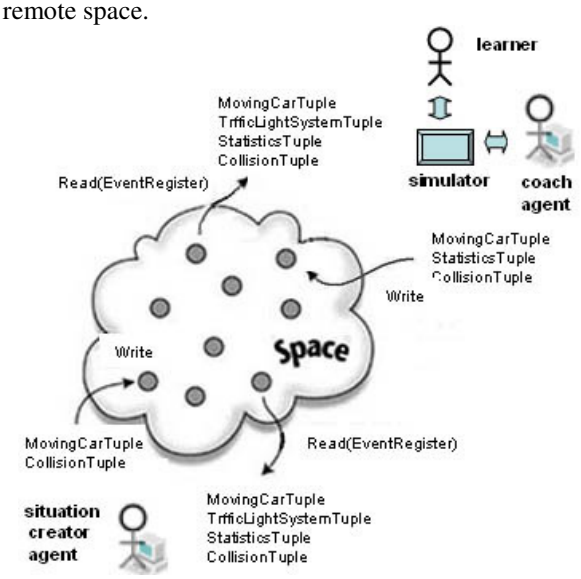


**Figure 1: Interaction between tuple space, learners, and intelligent agents**

Each learner controls his virtual car and views the shared driving place through a car driving simulator (locally running application). The state of his car is represented as a tuple in the space. The elements of such a moving car tuple are position, orientation, direction, speed, and the state of the brake and indicator lights. As the learner takes actions (e.g., speeding up, braking, changing direction, and turning on/off a light), the state of the car will change in the local data, and the responsible client application will remotely rewrite the corresponding tuple in the server. Other clients that are interested in knowing the update of the tuple will be informed. These clients will then read the tuple and update their local data. Thus, consistence of the dynamic data is preserved.

As shown in figure 1, each learner's application client includes a *coach agent*, a pedagogical agent that can provide the learner with situated instructions (feedback on the user's driving performance, warnings, etc.). Conceptually, the described static and dynamic objects in a driving place are elements of so-called "situations". A situation is defined as an identifiable configuration of a set of static and dynamic objects surrounding a user's car. We used JESS [4], a rule-based logic programming system, as the technical platform to encode these objects as facts. The production rules used within the JESS engine represent the following knowledge: situation detection, expert knowledge, and pedagogical knowledge. In summary, a coach agent can "look over the shoulders" of a learner and provides advice when detecting a mistake. Note that the performance data of a learner (e.g., how many times the learner committed a certain type of errors) is stored in the server as a tuple as well, and has an impact on the feedback given by the coach agent.

While a coach agent does not appear directly in the shared driving place, a *situation creator*, another pedagogical agent represented as a car in the virtual driving place, can move around in the shared driving place. He drives according to traffic rules and can intentionally create situations for learners. As described in [9], a situation creator has special knowledge to create situations according to the needs of learners. The created situations are *normal* and often occur in the real world – e.g., a situation creator is able to bring learners into situations where they have to respect certain rules about right of way at traffic junctions. The situation creators are opportunistic and intentionally look for chances to create such "normal" situations that offer learning opportunities to trainees.

## 3. The Design of the Problem Creator

The purpose of the work described in this paper is to train learners how to handle *abnormal* situations by setting up dangerous situations in virtual reality. We extend the concept of the situation creator and develop the *problem creator*, a pedagogical agent that can intentionally create abnormal situations which challenge learners. In contrast to the normal situation creator, the problem creator will not always respect the traffic rules (thereby causing abnormal situations). Another difference is that, while the situation creator continuously attempts to create "standard" traffic situations (in order to make students get used to them and handle them adequately), the problem creator has a large degree of unpredictability and just spontaneously creates few but highly abnormal and dangerous situations. Together, these two agent types model quite well realistic traffic situations.

A problem creator, represented in the shared driving place e.g. as a vehicle, a pedestrian, a bicycler, a motorbike, or an animal, sometimes behaves irrationally. Its behavior intends to confront learners with an emergency. Learners will be trained to make decisions and take actions quickly in order to avoid accidents. Three examples: a problem creator could be designed as a boy who walks along the street. This boy may suddenly dash forward onto the road immediately in the front of the learner's car. Another example is a simulation of a drunk driver who cannot control his car properly and leaves his lane, frontally approaching the learner's car. This puts the learner into a dangerous situation where he has to react immediately to avoid a crash. A third example is a so-called *crazy-driver*, which we will describe in detail in the next section. Similar to all problem creators, a crazy-driver moves on the roads rationally most of the time. However, it may brake suddenly and without any reason in front of a learner's car.

A problem creator is generally designed with two states. In the rational state, it behaves normally and according to the traffic rules. The learner can not distinguish it from other learners' vehicles or normal situation creators. For example, a child walks along the sidewalk, a drunk driver stays on his lane, and also the crazy-driver does not brake without reasons. In the irrational status, they all behave abnormally and intentionally cause trouble. It is important to note that a problem creator is not always in the irrational status, thereby retaining the spontaneous character of the situations they invoke.

A problem creator has domain knowledge, specific knowledge, pedagogical knowledge, and access to user models. While the domain knowledge is used to control the behavior of the simulated object in rational state, the specific knowledge is used to create dangerous situations in the irrational state. Note that both domain knowledge and specific knowledge are not directly taught to learners by the agent. The pedagogical knowledge is represented as a set of pedagogical strategies that determine when, how urgently, and how frequently to create dangerous situations. The user model maintains information about how often a learner has already encountered dangerous situations of a specific type, about his past performance in these dangerous situations, and the time of the last abnormal situation.

Since a problem creator does not communicate directly with users, it has no explicit interaction model (in contrast to typical pedagogical agents). However, a problem creator indirectly communicates with learners through the simulation environment. Technically, it works as a repeated process which starts by capturing the current state of the environment and ends by acting on the environment. This cycle is similar to the perception/action cycle of human users who also continually adjust their driving actions based on changes in the environment.

## 4. The Implementation of a Crazy-driver

As a special situation creator, a problem creator has the same system architecture and generic work procedure as a situation creator. These are described in [9]. In this section, we describe the implementation of the crazy-driver, a particular problem creator. Figure 2 shows its algorithm. The two large gray rectangles represent the crazy-driver agent and the simulation environment. Inside the agent rectangle, each white rectangle represents a functional component and each diamond represents a decision. Each solid line arrow indicates the control flow, and a dashed line arrow indicates the interaction between the agent and its environment. A processing cycle consists of two phases separated by a dashed line in the diagram. The task in the first phase is seeking or maintaining a goal, and in the second phase the agents attempts to reach the goal through making and executing an action plan. Note that a crazy-driver has an independent process thread that randomly switches the state of the agent between rational status and irrational status.

In each processing cycle, the crazy-driver first monitors the state of the driving place and the performance of the learners. The crazy-driver then checks its current status. If in rational status, it will behave like a skilled driver and take a random route on the streets. Otherwise, it will check whether it already has a goal or a goal has been achieved. A goal of a crazy-driver is to brake in front of a specific target car controlled by a learner. If a goal has been achieved or no goal exists yet, the crazy-driver seeks a new goal. To do so, it looks for learner-controlled cars. The crazy-driver puts all candidates in a queue sorted by distance between the agent's car and learners' cars and by driving direction. The agent then first selects the learner whose car is closest to and behind his own car. When a candidate is selected, the crazy-driver looks up in the learner model whether the candidate needs training in reacting to the emergency situation (braking of the front car) according to his past performance. If the candidate needs training, the agent will treat this candidate as the target (set a new goal). Otherwise, the agent considers the next candidate until the queue is empty. If no candidate can be found in the queue, it means that the agent fails to seek a goal in this cycle

and will behave just like a skilled driver (i.e., drive correctly with no specific destination). However, it will try to seek a goal in the next processing cycle.

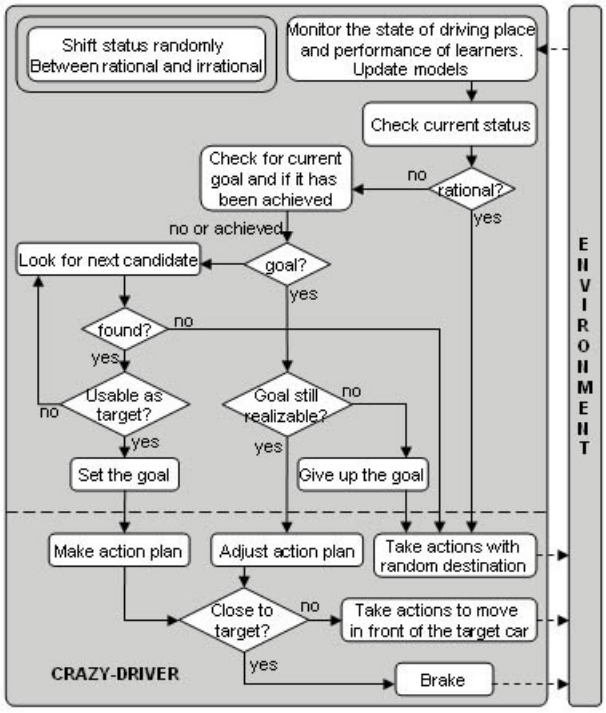


**Figure 2: Flowchart of the agent processing**

If a crazy-driver already has a goal, the goal will be evaluated in each processing cycle. Due to the nature of the dynamic simulation, it may happen that the goal cannot be realized any more. Whether the goal is still realizable or not depends on the current state of the learner's car (e.g., position, direction and velocity), the driving place (e.g., road network) and the state of the agent's car. E.g., if a target car, for any reason, moves away and it is too difficult for the agent to get in front of the target car, the goal is not realizable any more. The agent then gives up the goal and behaves like a normal driver in this cycle. For the crazy-driver, a simple rule is: as long as the target car is moving in the same direction on the same road and is behind the agent car, the goal is still realizable.

If an agent has a realizable goal (newly set or not), it performs actions to achieve it. To do this, it first creates an action plan that can be adjusted later if needed. For example, if there are other cars in between the agent car and target car, the crazy-driver agent will slow down and let other cars overpass. It will always try to reach the position just in the front of the target car. Having achieved this, it then adopts certain strategies (e.g., speeding up or slowing down) to seduce the target to violate the safety distance rule. Whenever the target car is close to a distance of unsafety, the agent will then brake suddenly.

## 5. A Usage Scenario

In this section we describe a brief usage scenario that shows how the problem creator and the other agents work from the perspective of a learner.

When a learner logs into the system and chooses a driving place, he can see a lot of other vehicles moving in the driving place through his simulator. However, he does not know which of these cars are controlled by other learners and which are controlled by agents. He can navigate his car through the driving place. Whenever a situation occurs that the trainee does not handle well, the coach agent provides advice. It also gives feedback on successfully mastered situations. Occasionally, abnormal situations may take place (e.g., a child runs onto the road or a car enters his lane and approaches him frontally). Such abnormal situations may be caused by co-learners since they often make mistakes. However, it is more probable that the abnormal situations are caused by the problem creators because they deliberately create dangerous situations.

For example, when a learner moves the car on the way, suddenly the brake-light of the car in the front goes on, and the learner has to stop immediately to avoid a collision. If he is successful, he can continue driving in the shared driving place. Note that his sudden braking creates an emergency for the cars behind him, which may be the cars of other learners. If a driver does not solve this situation and collides with the car of the problem creator, as a consequence the learner will be informed by the coach agent how serious the accident is and what he should do in such a situation. Then he will resume from a parking place. After some training time, the learner has understood the importance of, while driving in "normal" traffic, still being always prepared for unpredictable dangerous situations, and has practiced how to react in these situations.

## 6. Related Work

This section discusses related work in two aspects. We first compare our approach to other car driving simulators, and then compare the problem creator with the "trouble-maker", a similar pedagogical agent.

As mentioned in the introduction, abnormal situations in existing car driving simulators are usually hard coded or scripted [3, 11]. Repeated and predictable occurrences of the same predefined "emergency" in a training program may lead the trainee

to a false impression. He may well be able to handle the repeatedly occurring emergency situation successfully. However, this does not imply that he can deal with the same type of situation occurring in different circumstances, particularly if he has trained the same repeated process with the simulator a lot of times. Our agent-based approach ensures that similar situations will occur unpredictably and in different contexts. In addition to the unpredictable behavior of the agents, also the multiple participating human users who practice in the same virtual driving place make it impossible for a learner to foresee exactly how they will be challenged and how they will have to react.

Conceptually, the problem creator is similar to the trouble-maker agent, a special learning companion [1, 2] that sometimes disrupts learners deliberately by giving incorrect answers and by simply contradicting them. Such kinds of learning companions confront students with other's opinions and urge them to justify their own ideas. However, the problem creator has no direct interaction with the learner. Instead, it indirectly creates dangerous situations for learners in a 3D collaborative simulation environment.

In summary, our approach as presented in this paper can be distinguished from existing systems through its use of intelligent agent technology to provide special training situations in a collaborative 3D simulation environment.

## 7. Conclusion and Future Work

While simulation can not and should not be used to completely replace driving in a real car, the hours spent on the simulator can reduce the number of hours the learner needs to spend in a real car. Using the simulator for training can save time and costs. In particular, the learners can learn to handle dangerous situations (e.g., children, gap acceptance, dangers of alcohol, changing lines, open doors, and so on) without taking risks. Currently, car driving simulators are mainly used for training learners to react to emergencies by repeatedly using predefined scenarios. In this paper, we presented a new approach. An intelligent agent named problem creator was developed which can deliberately create dangerous situations within a collaborative simulation environment. The design and implementation of a problem creator, which can confront learners with abnormal situations unpredictably, has been described.

We have implemented a prototype system, which currently can only create three types of situations. We will extend it for creating more kinds of abnormal situations. Also, we are planning studies which measure the effect of the different agents (coach, situation creator, and problem creator) on the learner's skill gains.

## References


[1] Aïmeur, E., Dufort, H., Leibu, D., & Frasson, C., (1997). Some justifications about the learning disturbing strategy. In *Proceedings of AI-ED'97*, Japan, pp. 119-126.

[2] Aïmeur, E., & Frasson, C. (1996). Analyzing a new learning strategy according to different knowledge levels. *Computers & Education*, 27(2). pp. 115-127.

[3] Allen, R. W., Rosenthal, T. J., Parseghian, Z., & Markham, S. (2001). A scenario definition language for developing driving simulator research, evaluation and training courses. In *Proceedings of the 6th Driving Simulation Conference*, September 5 - 7, 2001, Nice, France.

[4] Friedman-Hill, E. (2003). *Jess in Action : Java Rule-Based Systems*. Manning Publications.

[5] Hays, R. T., & Singer, M. J. (1989). *Simulation Fidelity in Training System Design*. New York: Springer-Verlag.

[6] Lehman, T., McLaughry, S., & Wyckoff, P. (1999). TSpaces: The Next Wave. In *Proceedings of Hawaii International Conference on System Sciences (HICSS-32)*, January 1999.

[7] Miao, Y. (2004). Supporting Situated Learning for Virtual Communities of Practice: Representation and Management of Situated Knowledge. In *Proceedings of ICALT'04*, pp. 490-494, Joensuu, Finland.

[8] Miao, Y., Pinkwart, N., & Hoppe, U. (2006). Conducting Situated Learning in a Collaborative Virtual Environment. In *Proceedings of the 5th International Conference on Web Based Education*, pp. 7-12. Anaheim, CA: ACTA Press.

[9] Miao, Y., Hoppe, U., Pinkwart, N., Schilbach, O., Zill, S., & Schloesser, T. (to appear). Using Agents to Create Learning Opportunities in a Collaborative Learning Environment. In *Proceedings of the 8th International Conference on Intelligent Tutoring Systems*, June 26 - 30, 2006, Taiwan.

[10] Patrick, J. (1992). *Training: Research and Practice*. London: Academic Press Limited.

[11] Wolffelaar P., Bayarri S., & Coma, I. (1999). Script-based definition of complex scenarios. In *Proceedings of the 4th Driving Simulation Conference,* July 7 - 8, 1999, Paris, France.