Generating a 3D Simulation of a Car Accident from a Formal Description: the CarSim System

Arjan Egges, Anton Nijholt

Department of Computer Science
University of Twente
PO Box 217, 7500 AE Enschede
the Netherlands
Phone: (31) 53 4893686
Fax: (31) 53 4893503
Email: {egges,anijholt}@cs.utwente.nl

Pierre Nugues

GREYC laboratory
ISMRA
6, bd du Maréchal Juin
F-14050 Caen, France
Phone: (33) 231 452 705
Fax: (33) 231 452 760
Email: pnugues@greyc.ismra.fr

ABSTRACT

The problem of generating a 3D simulation of a car accident from a written description can be divided into two subtasks: the linguistic analysis and the virtual scene generation. As a means of communication between these two system parts, we designed a template formalism to represent a written accident report. The CarSim system processes template formal descriptions and creates corresponding 3D simulations. A planning component models the trajectories and temporal values of every vehicle that is involved in the accident.

1. INTRODUCTION

This paper presents the results of a prototype system to visualize and animate a 3D scene from a written description. It considers the narrow class of texts describing car accident reports. For example, such a system can be applied within insurance companies to generate an animated scene from a police report. The research is part of the TACIT[1] project [1, 2] at the GREYC[2] laboratory of the University of Caen, in cooperation with the ISMRA[3].

There are few projects that consider automatic scene generation from a written text, although many projects exist that incorporate natural language interaction in virtual worlds, like TRAINS [3], Ulysse [4, 5] and AnimNL [6].

Visualising a written story requires a different approach. The language analysis has to deal with texts where syntax and semantics are more complex than with spoken orders. TACIT bases its language processing techniques on a template filling paradigm, see [7] and [8] for more information about the conversion from text to template. This technique is successfully implemented in the FASTUS system [9], in which it is possible to obtain information from a static (written) text very efficiently.

2. FORMAL REPRESENTATION IN CARSIM

> "I was driving on a crossroad with a slow speed, approximately 40 km/h. Vehicle B arrived from my left, ignored the priority from the right and collided with my vehicle. On the first impact, my rear fender on the left side was hit and because of the slippery road, I lost control of my vehicle and hit the metallic protection of a tree, hence a second frontal collision." *Text A4, MAIF corpus, our translation.*

The text above describes an example of an accident from the MAIF[4] corpus. This corpus contains 87 accident descriptions that are written in French.

The task of simulating an accident is divided in two main tasks: extract the necessary and relevant information from the text, and create a 3D simulation from this (formalized) information. Figure 1 shows this division. The CarSim [5] system encapsulates the second task.

A final question is: why should we bother to make a 3D simulation of the accident? Proba-

[1]Traitements Automatiques pour la Compréhension d'Informations Textuelles

[2]Groupe de Recherches En Informatique, Image, Instrumentation

[3]Institut des Sciences de la Matière et du Rayonnement

[4]Mutuelle Assurance Automobile des Instituteurs de France. MAIF is a French insurance company.

[5]Car Accident Simulator

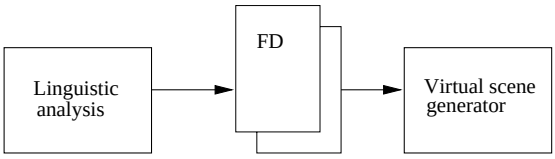

Figure 1: The two subsystems and the FD (Formal Description) as a means of communication.

bly a 2D environment will provide enough functionality to simulate almost every accident. We chose to use a 3D model, because of the extensibility. Perhaps, in the future, we might want an option in the system to see the accident from the driver's point of view. Furthermore, a 3D environment provides a way to view the accident in every possible angle.

We designed a formal model (M) that represents the accident. The CarSim system knows the following kinds of objects: moving objects (*dynamic* objects) and objects that do not move (*static* objects). Furthermore it knows *collision* objects. We define M as follows:

Definition 1 *Model M consists of three parts: a list M_S of k static objects $S_1, \ldots, S_k$, a list M_D of l dynamic objects $D_1, \ldots, D_l$, and a list M_C of n collision objects $C_1, \ldots, C_n$.*

In general, a static object can be defined with two parameters: one parameter defining the nature of the object and another parameter that defines the location of the object. Examples of static objects are: trees, crossroads, pedestrian crossings, etc.

Dynamic objects can't be defined by giving only their nature and position. Instead of a position, the *movement* of the object must be defined. In the CarSim formal representation, the movement of an object is defined by two parameters. The first parameter, the *initial direction*, defines the direction to which the object is headed before it starts driving. This direction can be to the North, South, East, or West. The second parameter is an ordered list of atomic movements. This list is called the *event chain*. Atomic movements can be driving forward, turning left and so on.

A collision is defined by giving the two objects, the collision coordinates and the parts of the vehicles that are involved in the collision (these parts can be the left side, the right side, the front or the rear, to keep things simple). A collision occurs between an *actor* and a *victim*. The victim can be either a static or a dynamic object, the actor clearly has to be a dynamic object.

With this formalism, most accidents can be described. For example, we give the formal description of text A4:

```
// Static objects
STATIC [
   ROAD [
      KIND = crossroad;
   ]
   TREE [
      ID = tree1; COORD = ( 5.0, -5.0 );
   ]
]

// Dynamic objects
DYNAMIC [
   VEHICLE [
      ID = vehicleB; KIND = car;
      INITDIRECTION = east;
      CHAIN [
         EVENT [
            KIND = driving_forward;
         ]
      ]
   ]
   VEHICLE [
      ID = vehicleA; KIND = car;
      INITDIRECTION = north;
      CHAIN [
         EVENT [
            KIND = driving_forward;
         ]
      ]
   ]
]

// Collision objects
ACCIDENT [
   COLLISION [
      ACTOR = vehicleB, front;
      VICTIM = vehicleA, leftside;
      COORD = ( 1.0, 1.0);
   ]
   COLLISION [
      ACTOR = vehicleA, front;
      VICTIM = tree1, unknown;
   ]
]
```

3. PLANNING

Planning complex events like collisions requires a well-defined and flexible planning architecture. General planning algorithms which apply methods incorporating artificial intelligence, are discussed in [10] and [11]. The CarSim planner is much more straighforward, because the planning process is not as complex as a lot of traditional AI planning problems, see also [12]. The total planning process is performed by using five different subplanners, which all perform a small part of the total planning task.

3.1. The preplanner

The preplanner is a planner that ensures the consistency of the formal description. If some values are not given (e.g. coordinates of a static object or initial directions of dynamic objects) or some values imply a contradiction (a vehicle turning left on a straight road), this planner tries to find (default) values and to solve the contradictions, if possible. This planner is a simple knowledge base, as discussed in [12].

3.2. The position planner

The position planner estimates the start and end positions of the vehicles in the simulation. By default a vehicle is placed 20 metres away from the center of the (cross)road. If two or more vehicles are moving in the same direction, they can't all be placed at this distance because they would overlap. Therefore, if there is more than one vehicle facing a particular direction, the second vehicle is placed at a distance of 26 metres from the center and if there is a third vehicle, it is placed at 32 metres from the center[6]. Regarding the end points of the vehicles, the vehicle that is placed closest to the center, will have its end point placed farther away from the center. The vehicle initially having a start point far away from the center, will have an end point close to the center, so that every vehicle traverses approximately the same distance.

3.3. The trajectory planner

Based on the (very global) description of the movement of every vehicle in the formal model, this planner constructs a trajectory, represented by a set of points in the Euclidian space. Every event in the event chain is converted to a list of trajectory points. A turn is approximated by a number of points lying on a circle arc. Overtaking is modeled by using a goniometrical function.

3.4. The accident planner

The accident planner uses the trajectory that is created by the trajectory planner. Since event chains only include atomic movements and no collisions, this trajectory is planned as if there was no collision at all. The task of the accident planner is to change this trajectory in such a way that it incorporates the collision. Some part of it has to be thrown away and an alternative part (which ultimately leads to the point of collision) has to be added to the trajectory. For every vehicle, actor or victim, the trajectory is thus changed in two steps[7]:

1. Remove a part of the trajectory.
2. Add a part to the trajectory so that the final result will be a trajectory that leads the vehicle to the point of collision.

3.5. The temporal planner

The temporal planner of the CarSim system is not a planner in the sense of the planners described in [10] and [11]. The temporal planner of the CarSim system plans the temporal values of the trajectory in two steps. Generally, a trajectory consists of a number of 'normal' trajectory points, followed by a number of trajectory points that represent a collision. First the segment that is not part of any collision is planned. After that, the system plans the remaining segment. In the CarSim system, every trajectory point has a *time value*. This is a value between 0 and 1, with 0 representing the beginning of the simulation and 1 being the end of it. The temporal planner tries to find time values for the trajectory points so that the collisions happen in a natural way.

For a more detailed description of the temporal planner, see [13].

4. SIMULATION OF TEXT A4

Given the accident description in Section 2, the CarSim system can generate an animated scene of the accident. Figure 2 shows the two collisions.

5. CONCLUSION

Right now, the CarSim system is able to generate an acceptable simulation of least 50 texts of the accident descriptions in the MAIF corpus. This boils down to a little less than 60%! The accidents that can't be simulated, are accidents with motorcycles, accidents with an unusual cause or complex interactive models (for example simulations depending on traffic lights that change color). The performance of a final system that incorporates automatic linguistic analysis will certainly be lower. However, we believe that information contained in templates can be obtained with a good accuracy using information extraction techniques. Linking CarSim with the linguistic part corresponds to the next stage of our project.

[6] In the CarSim system, the maximum number of vehicles that can have the same initial direction is *three*.

[7] For a more detailed description, see [13].

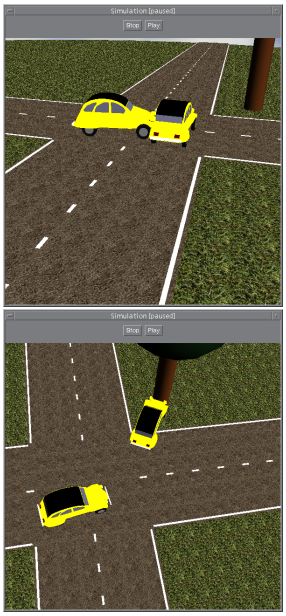

Figure 2: The two collisions in text A4.

6. REFERENCES

[1] F. Pied, C. Poirier, P. Enjalbert, and B. Victorri. From language to model. In *Workshop Corpus-Oriented Semantic Analysis in European Conference on Artificial Intelligence (ECAI)*, August 1996.

[2] C. Poirier and F. Pied. Analyse de constats amiables d'accident automobile. In *1er Colloque étudiant en Linguistique Informatique de Montréal*, June 1996.

[3] George Ferguson, James F. Allen, and Brad Miller. TRAINS-95: Towards a mixed-initiative planning assistant. In *Proceedings of the Third International Conference on AI Planning Systems (AIPS-96)*, May 1996.

[4] O. Bersot, P.O. El-Guedj, C. Godéreaux, and P. Nugues. A conversational agent to help navigation and collaboration in virtual worlds. *Virtual Reality*, 3(1):71–82, 1998.

[5] C. Godéreaux, P.O. El-Guedj, F. Revolta, and P. Nugues. Ulysse: An interactive, spoken dialogue interface to navigate in virtual worlds, lexical, syntactic, and semantic issues. In John Vince and Ray Earnshaw, editors, *Virtual Worlds on the Internet*, chapter 4, pages 53–70. IEEE Computer Society Press, 1999.

[6] N. Badler, W. Becket, B. Di Eugenio, C. Geib, L. Levison, M. Moore, B. Webber, M. White, and X. Zhao. Intentions and expectations in animating instructions: the AnimNL project. In *Intentions in Animation and Action*. Institute for Research in Cognitive Science, University of Pennsylvania, March 1993.

[7] *Proceedings of the fifth Message Understanding Conference*. Morgan Kaufmann Publishers, Inc., August 1993.

[8] *Proceedings of the sixth Message Understanding Conference*. Morgan Kaufmann Publishers, Inc., November 1995.

[9] Jerry R. Hobbs, Douglas Appelt, John Bear, David Israel, Megumi Kameyama, Mark Stickel, and Mabry Tyson. FASTUS: A cascaded finite-state transducer for extracting information from natural-language text. In Roche and Schabes, editors, *Finite State Devices for Natural Language Processing*. MIT Press, 1996.

[10] Nils J. Nilsson. *Artificial Intelligence, a New Synthesis*. Morgan Kaufmann Publishers, Inc., 1998.

[11] Yoav Shoham. *Artificial Intelligence Techniques in Prolog*. Morgan Kaufmann Publishers, Inc., 1994.

[12] P. Norvig and S. J. Russell. *Artificial intelligence: a modern approach*. Prentice Hall, 1995.

[13] J. Egges, P. Nugues, and A. Nijholt. CarSim: Automatic 3D scene generation of a car accident description. Technical report, University of Twente, 2001.