Towards routine large-scale, discrete spatial event simulations

Ellen-Wien Augustijn-Beckers[1] & Rolf A. de By[1]

[1]I.T.C., Hengelosestraat 99, P.O.Box 6, 7500 AA Enschede, The Netherlands
Telephone: +31 (0)53 4874444
Fax: +31 (0)53 487 4400
Email: augustijn@itc.nl, deby@itc.nl

1. Introduction

Simulations have great potential as tools to lead to a better understanding of the world's processes. Much previous research on spatial simulations was cell-based (cellular automata) but more recently, great advances have been made in object-oriented (vector-based) discrete event simulations. In most of these simulations the spatial component is underdeveloped. Many discrete event simulations are applied in fields where the spatial component is irrelevant. They are dynamic models of factories or manufacturing systems, elevators and other non-spatial scenarios.

Our knowledge of systems and simulations has reached a point where object-based behaviours can be directly translated into computable rules and used to generate systems. (Benenson and Torrens 2004). Compared to traditional simulation methods, object- or agent-based modelling is less abstract and closer-to-reality, since it explicitly attempts to model the behaviour of individuals (Wagner and Tulba, 2003). Worboys even states that the next real breakthrough in computer modelling of geographic phenomena comes when we move from an object-oriented to an event-oriented view of the world (Worboys, 2005)

Existing simulations are often application-oriented. They are not based on a general conceptual framework that allows for integration with a standard or spatial DBMS and/or Geographic Information Systems. Wagner and Tulba (2003) argue in favour of the integration of agent-oriented modelling software and information systems (including GIS) and agent-based simulation.

The main objective for our study was to gain understanding on how to routinely generate spatial simulations (agent-based discrete event simulations) that are linked to a standard spatial database or GIS. This resulted in a conceptual framework for simulation system development and a traffic simulation testing scenario.

2. Conceptual framework

Different fields of research contributed methods and concepts that were integrated in our conceptual framework. These include classical discrete event simulation, event-based approaches, geospatial event modelling, agent-object-relationship modelling, and object-oriented modelling (Zhou 2006).

A simulation has a data model or static model (using constructive geometry), a behaviour model and a dynamic model (simulation). The data model includes the moving

and non-moving objects, in our pilot example the road network, cars and travel plans. The fact that an object does not move does not imply that it is without behaviour. An example of non-moving objects with behaviour is the intersections: though they do not move, they are active as they implement rules of priority (which car will cross the intersection first).

The behaviour model hides in the active class definitions. The behaviour production system consists of behaviour rules, actions, conditions, memory rule interpreters and arbiters. Together, they define an agent's behaviour. Behaviour rules explicitly store the interaction pattern by defining what-if scenarios. Each rule consists of four parts, namely (a) a class name *n*, (b) a condition *c(s,o)*, (c) a characteristic *a*, and a nameless function *f(s,o)*. The operation of these rules has been coined 'spatial awareness', their application is roughly as follows. An object *s* (for self) is capable of detecting which other objects *o* are in its vicinity; the rule determines how *o* affects *s*'s behaviour. The rule only has an effect if *n* is the name of *o*'s class and the condition *c(s,o)* holds. If so, *s*'s characteristic *a* (speed is the normal example here) is affected and the effect is determined as *f(s,o)*. In case that *a* is speed, *f(s,o)* provides us with a suggested speed for *s*, due to *o*'s vicinity to *s*. Object *o* will collect all such suggestions from objects in its vicinity, and then decide what to do with them: With speed it will next assume the lowest suggested speed. Two specials of nearby objects complete the story: *s* suggests to itself its maximum cruising speed, and the road on which it travels suggests its maximally allowed speed. This approach is generic as we allow nearby objects to influence any characteristic of the object, given a condition.

The dynamic model of a discrete event simulation is based on a chronology of events. Each event occurs at a time instant and marks a relevant change of state in the system. Conceptual components of such a dynamic model are the clock that keeps track of simulation time, and the event list. Examples of events in our simulation are: a moving object arriving in a new location, and a traffic light changing light. Each event has a time of occurrence; the simulation schedules events in its event list(s).

3. The prototype simulation

The objective of the prototype was not to construct a realistic traffic simulation including all possible agents necessary to model the complexity of realistic city traffic. It was meant to demonstrate the applicability of the conceptual framework, to test the link between the database and simulation, and to investigate the effects of scaling up the number of agents.

The programming language used is Python, in combination with the package SimPy. First, the simulation was developed as a stand-alone simulation. After initial testing the simulation was adapted to allow a connection with our spatial database. This connection was needed to load the simulation with a realistic amount of input data to test the performance of the system.

3.1 Data model

The data model or static model was implemented with the PostgreSQL/PostGIS spatial DBMS. The database system is used to store a single lane, directed single modal network. It also stores information about moving objects and their travel plans, which include start location, destination and start time. Our moving objects for now model cars, and these will start at a given start time and location, and will follow their associated travel plan until they reach their destination. Each car is associated with a single travel plan and this travel plan will remain unchanged during the simulation. The network is enabled with an M domain to allow measurement along the road segments. This ensures that the car can start and end its journey at any position along the road segment. Travel plans indicate on which road segments the car will travel and in which order these road segments will be visited. The travel plan also stores information about the direction in which the care will move over the road segment, differentiating between the to-from and the from-to direction.

3.2 Behaviour model

Only the car, intersection, road segment and traffic light classes currently have behaviour defined. Traffic lights behave independently: they display periodic change of light (green, yellow, red), but do not have interaction with any of the other agents. They are, however, 'visible'.

The intersection has interaction with the cars. The behaviour of an intersection is to regulate the occupation of the intersection space; only one car can pass at any time. It implements rules of traffic priority, and thus it will assign a priority to all waiting cars. It will do this by maintaining a waiting queue and a swerving queue (active queue).

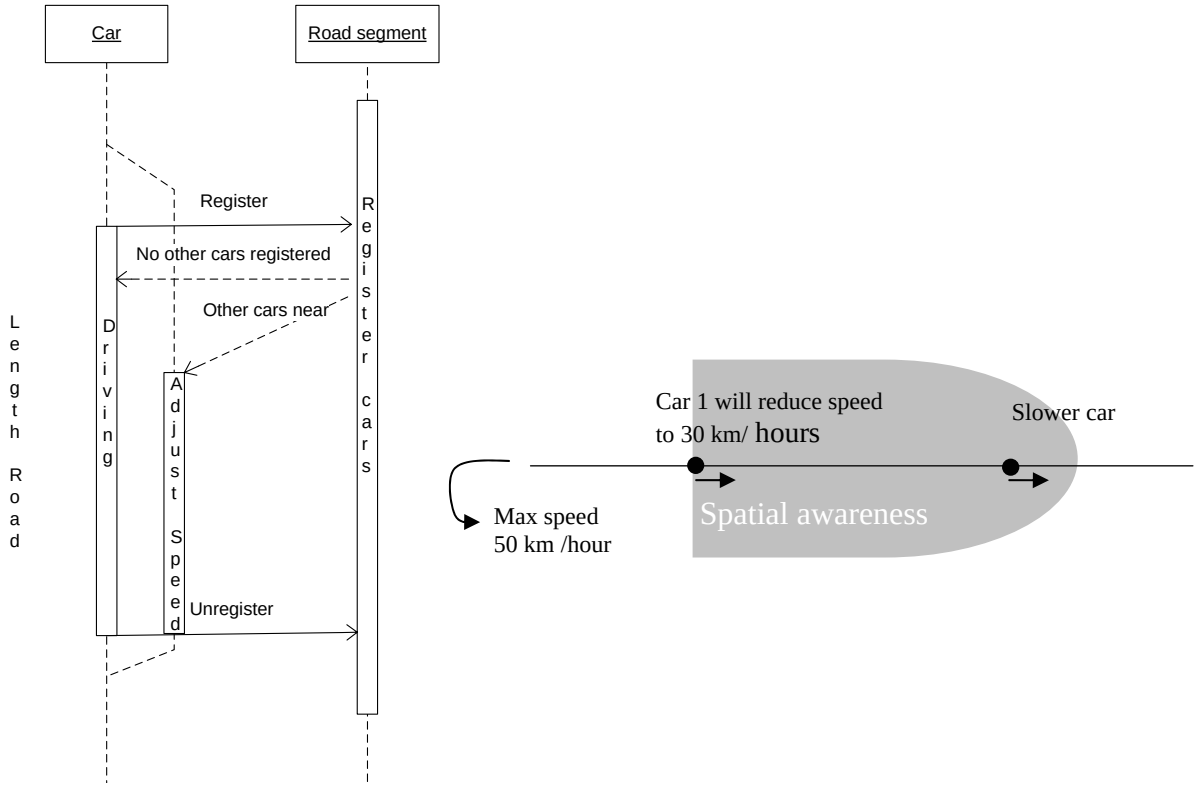

Figure 1 (left) Sequence diagram, (right) cars will adjust their speed when slower cars are within their spatial awareness.

The road segment has interaction with the cars. Behaviour of the road segments is the registration of the location of cars. Each road segment register's all cars along its road at a specific time, and when requested provides the information of the position of other cars so that a car knows which other moving objects are close and can use this information to adjust its speed or come to a stop in order to prevent collision. The road segment does not communicate the last registered position of a car but the position of this car at the requested time.

During its journey, a car will possibly meet and interact with other objects like cars, intersections and traffic lights, until it reaches its destination. The car is the only moving object that changes its position, has a speed and direction of movement, and interacts with other cars to adjust its behaviour. The speed of the car will vary according to restrictions imposed by the road segments (maximum speed limit) and the closeness of obstacles. Cars do not actively change direction, since such changes simply follow from the segments in its travelplan. Change of speed is implemented as a step function, not as gradual acceleration or deceleration. Concerning the dynamic interaction model, only cars act interactively with other types of objects/agents. The location of a car will induce it to request information from other objects/agents. This behaviour is the awareness that enables an agent to sense the presence of other agents in its vicinity and behave accordingly.

3.3 Simulation model

For performance reasons, interaction between the simulation and spatial database at present only occurs at beginning and end of a simulation. At start time, road network, cars and travel plans will be loaded. At the end, collected waypoints for all moving objects are retrieved from the simulation and uploaded to the database for later analysis.

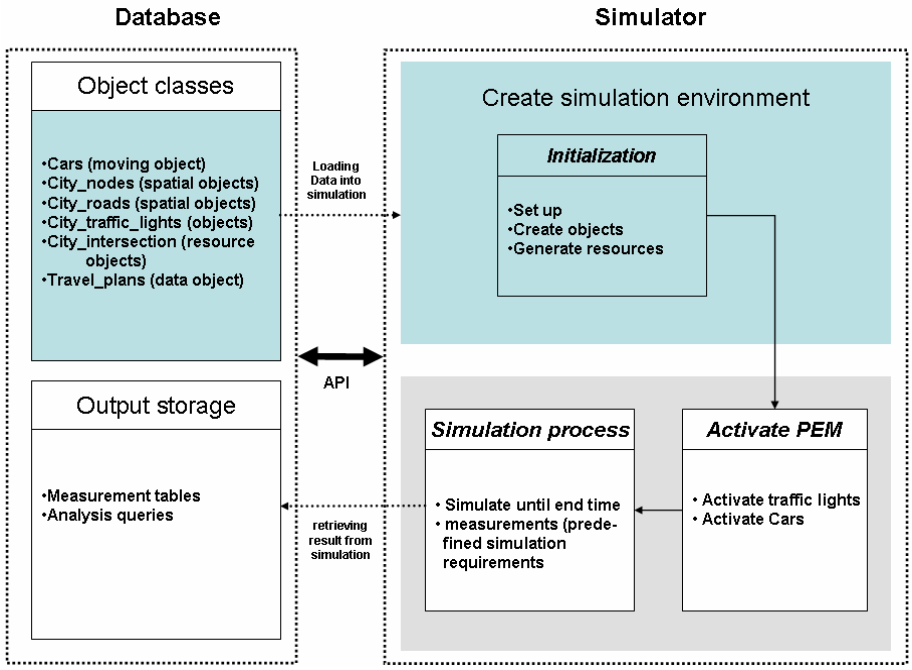

Figure 2, Integration of the simulation and DBMS (Karim 2007)

4. Up scaling of the system

In the first phase, our prototype system was developed and tested with only a few roads and cars. After the connection with the database the number of roads was scaled up to approximately 4000 roads, 250 traffic lights, and the simulation worked with 400 cars.

The performance of the system is good; it takes approximately 10 seconds to run the simulation with the numbers of agents, on a standard Windows XP desktop machine.

5. Conclusions

The key technology for building discrete event simulations for spatial phenomena that interact with generic DBMS and GIS systems is available. An open source programming language like Python provides good possibilities of linking to a DBMS. The discussed example shows the applicability of the techniques provided by object-oriented modelling/programming and discrete event theory.
Behaviour rules provide the possibility of implementing spatial awareness that enables an object to respond to other agents when they occur in its vicinity.

6. References

Benenson, I and Torrens, P.M. 2004, *Geosimulation: object-based modeling of urban phenomena* Computers, Environment and Urban Systems, 28 (1-2): p. 1-8

Karim, F. 2007, Upscaling a spatial discrete event simulation with DBMS, MSc thesis, ITC Enschede.

Wagner G and Tulba F, 2003, *Agent-Oriented Modeling and Agent based Simulation,* ER 2003 Workshops, LNCS 2814 pp 205-216, Springer-Verlag Berlin Heidelberg.

Worboys, M, 2005, Event-oriented approaches to geographic phenomena, International Journal of Geographic Information Science, Vol 19, no 1, January 2005, 1-28

Zhou, Y.H. ,2006, *Explorative research on methods for discrete space/time simulation integrated with the event-based approach and agent concept,* ITC Enschede, the Netherlands.