Using Serious Game Techniques to Simulate Emergency Situations

Marcelo G. Metello, Marco A. Casanova, Marcelo T. M. Carvalho

Departamento de Informática
Pontifícia Universidade Católica (PUC) – Rio de Janeiro, RJ – Brazil
{metello,casanova}@inf.puc-rio.br, tilio@tecgraf.puc-rio.br

Abstract. *This paper presents a simulation engine we implemented to support an interactive emergency simulation game. We first discuss various issues on building serious (non-entertainment) games involving the simulation of real life situations. The discussion also considers the use of geographical information systems and dynamic modeling techniques. Then, we present the architecture of the simulation engine and discuss the main aspects regarding its application to emergency plans. Finally, we briefly describe a prototype built to validate the proposed architecture and the requirements raised in the discussion.*

Resumo. *Neste trabalho apresentamos um simulador implementado para atender as necessidades de jogos interativos simulando situações de emergência. Inicialmente, apresentamos os problemas inerentes ao desenvolvimento de jogos sérios (sem propósito de entretenimento) que envolvem a simulação de situações do mundo real. A discussão também considera a adoção de sistemas de informação geográfica e de técnicas de modelagem dinâmica. Em seguida, apresentamos a arquitetura de um simulador e discutimos os aspectos principais levantados pela sua aplicação no contexto de planos de emergência. Por fim, descrevemos brevemente um protótipo construído para validar a arquitetura proposta e os requisitos levantados.*

1. Introduction

The first computer games were created in the early 1960's. Since then the computer game industry grew into a large segment that now plays a relevant role in the evolution of several areas of Computer Science, such as human-computer interfaces, computer graphics, artificial intelligence and, more recently, computer networks [Smed et al 2002].

The widespread adoption of computer games for entertainment purposes, the continuous decrease of hardware cost and the success in military simulations made gaming technologies attractive to some "serious" industries such as medicine, architecture, education, city planning, and government applications [Smith 2007]. The term *serious games* [Susi, Johannesson and Backlund 2007] has been used to denote games used for such non-entertainment purposes. The application of gaming technologies to these areas presents some peculiar challenges, since their requirements can be quite different from those of the entertainment industry. Usually, serious games

need to work on models which reproduce certain aspects of reality. On the other hand, entertainment games have much more freedom to create and modify their own reality, which can be quite convenient, especially when the developers face technical limitations. Even though entertainment games may require realistic audiovisual player experience, they do not need to reproduce realistic situations.

Apperley (2006) classifies video games into four genres according mainly to the characteristics of the player interaction. Specifically, the *simulation* genre better defines the applications we focus in this paper. Still according to the author, simulation games can be further analyzed with respect to their degree of realism, which is the main aspect that characterizes serious games. We will therefore use the term *serious simulation games* to denote serious games that belong to the simulation genre.

The fact that serious games may require realistic simulations justifies the effort to integrate gaming techniques with traditional *geospatial dynamic models*. The purpose of geospatial dynamic models is to describe processes that have some important spatial aspect. Such processes may include natural phenomena and human action on Earth. Examples of geospatial dynamic models are extensively found in the literature related to fields such as hydrology, climate changes, land use, population dynamics and many others. Such models improve our understanding of dynamic phenomena as they make it explicit the causal relationships between the elements involved.

Since geospatial dynamic models attempt to describe real phenomena, they help meet the requirements of serious games for realism. Hence, the effort to make dynamic modeling engines interoperate with simulation game engines is perfectly justifiable. Ideally, the player interaction capabilities of computer games and realistic dynamic modeling techniques should be integrated in a complementary way.

This paper illustrates the combination of these two approaches to simulate an emergency response activity. An emergency situation occurs when an incident can cause damage to human health and the environment. Once it occurs, the best approach to control it is to respond quickly and in a well organized manner. Testing the performance of an emergency response team is mandatory to ensure minimum impact of the incident. Testing usually takes the form of field exercises, but simulation has also been successfully used. We discuss the advantages of adding simulation capability to an emergency information management system, such as InfoPAE [Carvalho et al. 2001], a system used to manage emergency situations at Petrobras, the Brazilian oil company.

The paper is organized as follows. Section 2 lists the requirements for serious simulation games. Section 3 describes a simulation engine that integrates traditional dynamic models, and discusses issues related to player interaction in simulation games with geospatial aspects. Section 4 presents the emergency simulation game. Finally, Section 5 contains the conclusions and enumerates future work.

2. Requirements for Serious Simulation Games

This section lists some of the requirements for serious simulation games that are not fundamental to other classes of computer games.

2.1. Realism

There should be a minimum acceptable degree of realism in serious simulation games. This suggests that the simulations involved in this kind of game should run on data that represents real objects, which is precisely the difference between graphical and geographical data.

In many cases, the ability to access data in existing systems, such as geographical information systems (GIS), will work as a subrequisite to realism. Indeed, in order to create a realistic simulation, the system has to work with real data, which is likely to be stored in existing information systems and databases.

2.2. Time Flow Control

The ability to stop, accelerate and go back in time can be quite important when designing serious simulation games. It is not difficult to imagine situations where it would be desirable to pause or to replay a simulation multiple times from a given point. In other situations, it may be desirable to accelerate the time flow, especially in periods without much player activity.

This requisite says that serious simulation games may require much more control over the time flow than entertainment games. Some serious simulation games may require that entire simulations be recorded for future replays.

2.3. Learning Support

Simulation games devoted to training require player evaluation as part of the learning process [Borodzicz and van Haperen 2002]. In many cases, this requirement means that the system should be able to play back a simulation for evaluation purposes, which involves saving simulation data.

Apart from simply saving simulation data, the system may be required to trace back player decisions for payoff evaluation [Zagal et al 2006]. In this case, the underlying simulation system should be aware of the player action model.

There are also cases where organizations will have predefined procedures and protocols. When this happens, it may be necessary to match the sequence of player actions to these predefined procedures to check whether each player acted as expected by the organization. Going one step further, the simulations can be used to evaluate the effectiveness of the predefined procedures, detect their flaws and help with their evolution [Smith 2004].

It should be noted that the requirement of learning support is not limited to individual learning. The evaluation of the effectiveness of predefined procedures represents some kind of collective or institutional learning.

3. The Simulation Engine

This section provides a high level description of the architecture of the implemented simulation engine, and discusses some interesting issues related to it.

In the context of this work, a simulation refers to a set of elements and events evolving in a spatio-temporal representation of the world. That is, the world state is represented by spatial and non-spatial data, and the state changes as time flows. The

changes in the state of the world are affected by input given by the players, which in part justifies why the simulation engine may also be considered a game engine.

3.1. Architecture

Figure 1 illustrates the high level architecture of the system. The simulation engine is responsible for continuously updating the state of the world as the simulation time flows. The renderer is responsible for generating visual input for the players, and is not considered as part of the simulation engine. This makes it possible to implement a multiplayer game with different renderers for different players, and to generate different views for each player, according to which information they are allowed to see.

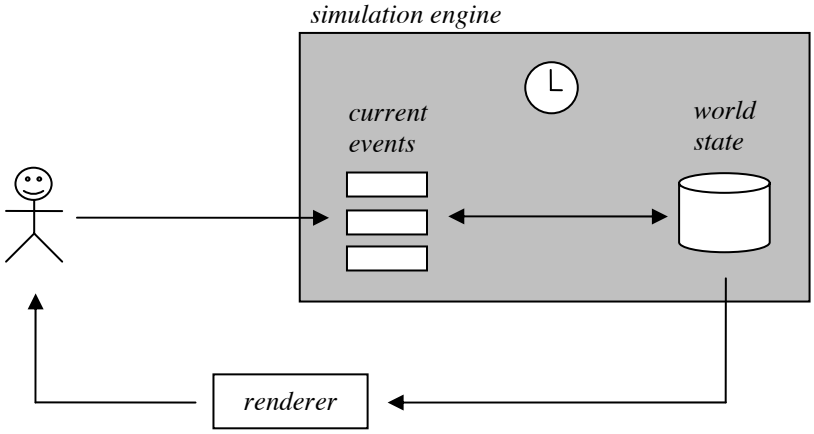

Figure 1. The high level architecture of the system.

All changes in the world state are carried out by special entities called *events*. There are two types of events: *player events* and *system events*. Player events represent player activity in the world. The way player input alters the state of the world is always defined by player events. On the other hand, system events represent all activity that is not directly caused by any player.

Events have duration. Therefore each event has a start time and a finish time. As an example, the event of moving a boat to a specified location at sea would start at the moment the order to move is given and finish at the time the boat gets to the specified location.

The lifecycle of a player event starts when the player issues a command to execute it. The player event then starts its activities by generating a series of instructions. Each instruction is basically a piece of code that somehow alters the state of the world in the simulation. The instructions are always instantaneous with respect to simulation time. Each instruction is assigned a timestamp, which will provide a means of ordering instructions originated from different concurrent events. In the case of the moving boat player event, each instruction would change the location of the boat to the next point in its trajectory to its destination. Note that, since each player event generates its instructions, they are responsible for determining the granularity of their execution. It is also possible for a player to have multiple events running at the same time.

There are three possible outcomes of the execution of a player event. It may *finish* its execution successfully, it may be *cancelled* or it may *fail*. A player event can

be cancelled in three situations. The player may issue a command to cancel one of his running player events. If the player issues a command to execute one player event that conflicts with another player event, one of them is cancelled by the system. As an example, if the player has ordered a boat to go to location *L* and, before the boat gets to *L*, he orders the same boat to go to another location *L'*, the event of moving the boat to *L* is cancelled. As the last possibility, when the simulation finishes, all running events are cancelled. Fig 2 illustrates the state diagram for player events.

The event model implemented by the simulation engine should be general enough to implement various kinds of serious games. One possible indication of this generality is that it is very similar to John Sowa's process model [Sowa 2000].

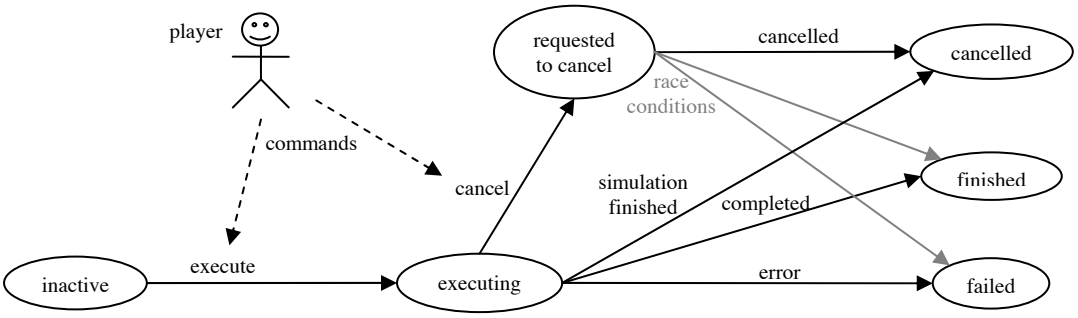

Figure 2. The state diagram of player events.

3.2. Issues related to dynamic modeling

Dynamic modeling frameworks, such as PCRaster [van Deursen 1995], Swarm [Minar et al 1996] and Nested-CA [Carneiro 2006] usually provide or use a specific language to describe dynamic models and an engine to run these models on real data, usually provided by GIS.

Since the event model used to implement the simulation engine is very generic, it is usually possible to implement most dynamic models as system events. In fact, it should be possible to emulate any dynamic model engine inside one system event. In this case, the event would issue an instruction every time the dynamic model engine should alter the state of the world. In fact, this is how the simulation engine proposed in this paper integrates traditional dynamic modeling and games.

Another issue is related to running time. Simulation engines and frameworks designed for dynamic modeling seldom show any concern for running the models in real time, which is a requirement for interactive games. The term *real time* is used here to denote that the time flow in the execution of a temporal model should be synchronized with the real time flow for better user experience.

The technology developed by the gaming industry is focused on displaying data in real time for highly interactive gaming. If we go through the process of game development, there is a continuous attention on performance requirements in almost every part of the code. Performance improvements are tried everywhere to keep an

acceptable frame rate for better user experience. This objective is always present throughout the game development process.

If the dynamic models and the spatial data are not too complex, the real time requirement can be implemented simply by adding a time controller to an existing dynamic modeling engine. However, since GIS are known for their heavy datasets, techniques for optimizing spatio-temporal data manipulation may be necessary [Wolfson et al 1998, Siebeck 2004].

3.3. Issues related to GIS

GIS have been traditionally more concerned with displaying spatial data than temporal data. Only recently the major GIS, other than military and flight simulators, started to take into account time and user experience, much in the same way as for the gaming industry. The increasing popularity of training games certainly contributes to this change.

GIS are known for their heavy spatial datasets. This is certainly one of the main reasons why it is difficult to display animated GIS data at a minimum acceptable frame rate. Metello et al. (2007) show how the fast computer graphics techniques used in gaming may be used to display GIS data at higher frame rates.

3.4. Issues related to multithreading

Most information systems and GIS follow a single-thread paradigm in their software architecture. The paradigm is characterized by a synchronous model of work where the user issues a command and waits for the response of the system before he can issue more commands. This type of architecture is clearly not adequate for interactive simulations, since the simulation should be run in real time and cannot afford waiting for user input. The simulation must continue even if the user does not issue any command. Besides, the system must also support multiple players, which is another argument against the single-thread paradigm.

In more dynamic applications, such as entertainment action games, the system executes a process continuously, regardless of commands input by the user in an asynchronous way. The differences in the execution flow of both kinds of architectures are illustrated in Fig. 3. In the workflow on the right, there are two processes running asynchronously. In the simulation engine, the interaction between these two processes is handled simply by queuing *player events* when they are created by a player. Of course, the queue must be thread-safe.

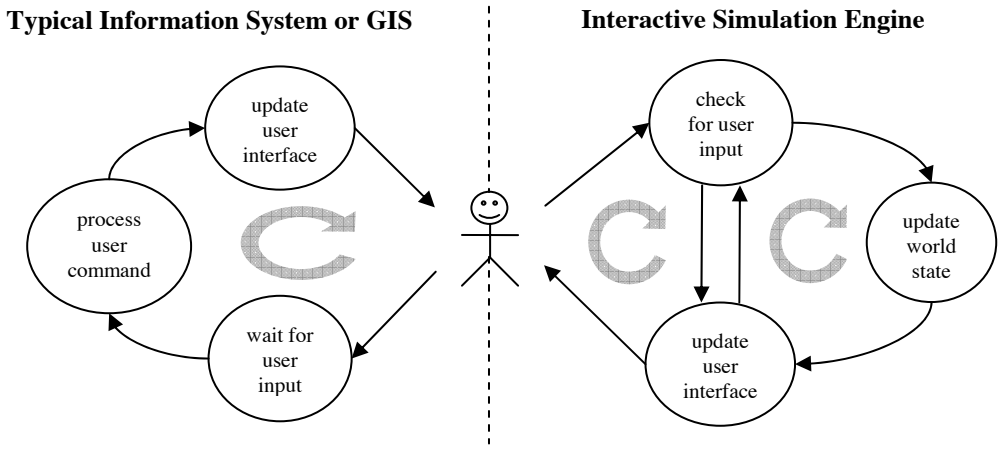

Figure 3. The kind of architecture required for interactive simulations (right)

4. The Interactive Emergency Simulation Game

In order to validate the proposed architecture, a game was implemented with the objective of simulating some specific emergency situations. In the context of this paper, an emergency is an incident, like an oil spill, that requires response action to be controlled and to mitigate effects, as loss of life or damage to property and natural resources.

Preventing the incident is always the best for avoiding damage to human health and the environment. However, once an emergency occurs, the best approach to control it is to respond quickly and in a well organized manner. This will happen if response strategies have been planned ahead of time. One of the elements for this planning is the emergency plan, which comprises a set of instructions that outline the steps that should be taken before, during, and after an emergency. The emergency plan is based on a risk assessment that looks at all the possibilities of what could go wrong. To assist and support a better response action, the plan contains, besides the set of instructions, the response strategies, a list of contacts and personnel, a material resource list, and refers to a vast documentation.

After the plan is developed, it is important to test it to check whether it works as expected and to train and evaluate the operational readiness of responders. Testing usually takes the form of an exercise or drill, what can be very time consuming and expensive to organize periodically. Another point to consider is the difficulty for representing detailed and realistic situations required to effectively test the emergency plan. Use of simulation games in these cases can be helpful.

This section further discusses the motivation outlined above and the implementation of a simulator.

4.1. Representation versus simulation: limitations of planning response ahead of time

In [Frasca 2003], the author discusses two different approaches for modeling knowledge about dynamic phenomena: representation and simulation. According to the author, the

main difference between both forms is that simulation attempts to model the behavior of the elements involved in the phenomenon while representation is limited to retaining the perceptual characteristics of it. To make it clear, the author gives the example of a plane landing procedure. A representation of a specific landing could be a film where an observer would be incapable of interfering. On the other hand, a flight simulator would allow the player to modify the behavior of the system in a way that simulates the real plane. This flexibility is only possible due to the simulation characteristic of modeling the behavior of the elements independently of any specific scenario.

An emergency plan takes, traditionally, a more representational form. It contains response strategies planned for different type of scenarios, but it cannot tell whether the plans are well suited for all cases. The set of instructions contained in an emergency plan consists basically of workflows of actions. They do not take into consideration the preconditions or durations of each action. Moreover, they do not take into consideration the specific spatial characteristics of all scenarios. For example, a plan can describe the action of sending two boats to intercept an oil spot. However, it may not be possible to do that before the oil reaches the coast in some specific conditions. If emergency managers were able to simulate the whole process in a more realistic way, it would certainly make the emergency plans more reliable.

In order to meet all these needs, a simulator was developed with the purpose of simulating emergency scenarios. Some of the main advantages of building an emergency simulator and integrating it with an emergency management system include:

- Simulations help finding flaws in emergency plans
- Testing whether available emergency response resources configuration are enough to handle any scenario requirements
- Simulation games provide training that help improve personnel performance
- Computer simulation cost are significantly lower than functional or full scale exercises

The first scenario considered to test the simulator was the spill of a considerable volume of oil into the ocean. This scenario was chosen because it involves elements of dynamic modeling and user actions that interfere with each other.

4.2. The Oil Spill Scenario

In a typical oil spill scenario, the initial goals are to attempt to control the leak at the source of the spill, and to limit the propagation of the floating oil as much as possible. These goals are typically achieved by using containment, recovery and clean-up strategies, which are implemented through specific operational procedures. These procedures, however, depend on the characteristics of the scenario, such as:

- Type of oil and its characteristics
- Local of the source of the leak and its nearest landmark
- Estimation of the amount of oil spilled
- Weather and sea conditions
- Characteristics of the shoreline that may be affected

4.3. The Emergency Simulator

The first simulation implemented considers the oil spill scenario after the leak has stopped and focuses on oil contention in water bodies. Clean-up operations for oil that reached the coast were also not considered. The goal of this simulation is to test emergency plans for leaked oil containment. The simulator is expected to uncover possible flaws in the plans as well as to help planning equipment installation and location.

The initial conditions of the simulation are specified in a document which is read by the system. This document includes the location, amount and type of leaked oil, maps of the affected area, the location of all available equipment and weather conditions. The dynamic elements of the simulation are described next.

The leaked oil is represented by a hexagonal cell space, where each cell stores the amount of oil in it. The movement of oil is modeled using a cellular automaton, which considers not only the state of each cell but also the weather conditions, such as the wind direction and speed, for example. Currently, weather conditions are globally defined.

The oil movement model must also consider elements that will act as obstacles, such as shorelines and barriers used to contain the oil spilled. Both are represented as polylines in the simulation and have similar treatment. Each cell that intersects either a barrier or a coast line is considered an obstacle cell.

When the oil reaches the coast, its movement must take into consideration the type of the coast. For example, sand coasts absorb oil in a much greater amount than rocky coasts. It will be helpful to use a detailed map of coast types for all locations that can be reached by the oil spill. Likewise, different types of barriers have different absorption and containment capabilities.

Each cell in the cellular space must contain information as whether it represents an obstacle or not. Note that this state may change during the simulation when barriers are used. Since different obstacles have different containment and absorption characteristics, the obstacle cells must also keep this information. All the information about obstacle cells is of course used as input to the cellular automaton as well.

The main dynamic elements in the simulation other than the oil itself are the boats used to support operational response to the emergency situation. Boats are responsible for placing barriers to contain oil. They may also carry some oil recovery equipment, such as pumps and skimmers. The initial location of the boats is defined in the document that describes the initial conditions for the simulation.

The movement of the boats is simulated by taking into account their speed and cargo capacities, as well as weather conditions, such as wind, sea currents and tide. During the simulation, players guide the boats through way-points. They may place way-points wherever they want and send the boats to any of them. Of course, boats may also encounter obstacles such as islands. In this case, they just stop and wait for further instructions.

The placement of barriers is an operation executed by two boats. Each boat holds one end of the barrier. The idea is that, when the oil spot passes between the boats, it gets blocked by the barrier. After that, the recovery operation starts. The

command to place a barrier needs some parameters, such as which barrier should be used, the angle at which it should be put, the distance the boats should keep from each other and the curvature that should be kept. In order to execute the action of placing the barrier, some preconditions must be met. The two boats must not be too far away, the barrier must be long enough for the distance and curvature given as parameters, and the weather conditions must not prevent the placement of the barrier.

5. Conclusion and Future Work

The architecture used in the simulation engine proved to be a way of integrating traditional dynamic modeling with interactive computer games. This is crucial to give proper realism to a simulation, as an emergency response simulation.

Although the simulator described helps training and evaluating emergency plans, the analysis of the performance of the players is still a manual process. Future versions should consider the possibility of integrating the flow of player actions with the predefined workflows in emergency plans in an automatic way.

Another issue of interest is to integrate the simulator with an emergency information management system, which usually holds a detailed database of emergency plans for different types of scenarios, but claims for representing detailed and realistic situations where to test emergency plans. InfoPAE [Carvalho et al. 2001] provides an example of such systems. InfoPAE was designed to manage emergency situations at Petrobras, the Brazilian oil company. Besides the emergency plans, the database also provides detailed data about procedures, material resources, facilities, available equipment and geographical data.

Using the simulator with such systems will provide a good environment to test the emergency management circular process through which managers prepare for emergencies, respond to them, recover from them and mitigate their effects, and prevent future emergencies from occurring. In this sense, simulation plays a critical role to assess, improve performance and prepare for future emergencies.

The scenarios used to test out prototype were derived from the InfoPAE databases. It might be interesting to create a script language for defining different emergency scenarios, even if they are hypothetical. This could help improving the training process by generating sequences of scenarios with increasing difficulty levels, just like in entertainment games.

References

Apperley, T. (2006) “Genre and game studies: Toward a critical approach to video game genres”. Simulation & Gaming, 37(1), 6-23

Borodzicz, E. and van Haperen, K. (2002) “Individual and Group Learning in Crisis Simulations”. Journal of Contingencies and Crisis Management 10 (3), 139–147 doi:10.1111/1468-5973.00190

Carneiro, T. (2006) “Nested-CA: A Foundation for Multiscale Modelling of Land Use and Land Cover Change”. Doctorate Thesis from the Post Graduation Course in Applied Computer Science, INPE - Sao Jose dos Campos, Brazil

Carvalho, M.T.; Freire, J.; Casanova, M.A. (2001) “The Architecture of an Emergency Plan Deployment System”. Proc. III Workshop Brasileiro de GeoInformática, Rio de Janeiro, Brasil, Oct. 2001.

Frasca, G. (2003) "Simulation versus Narrative: Introduction to Ludology". In: Wolf & Perron (Eds.) The Video Game Theory Reader. Routledge.

Metello, M, et al. (2007) “Continuous Interaction with TDK Improving the User Experience in Terralib”. Proc. IX Brazilian Symposium on GeoInformatics.

Minar, N., Burkhart, R., Langton, C. and Askenazi, M. (1996). “The Swarm Simulation System: A Toolkit for Building Multi-agent Simulations”. Working Paper 96-06-042, Santa Fe Institute, Santa Fe.

Siebeck, J. (2004) “Concepts for the Representation, Storage, and Retrieval of Spatio-Temporal Objects in 3D/4D Geo-Information-Systems”. Rheinische Friedrich-Wilhelms-Universitat Bonn, Diss.

Smed, J., Kaukoranta, T., Hakonen, H. (2002) “Aspects of Networking in Multiplayer Computer Games”. The Electronic Library, Volume 20, Number 2, 2002, pp. 87-97(11)

Smith, D. (2004) “For Whom the Bell Tolls: Imagining Accidents and the Development of crisis Simulation in Organizations”, Simulation & Gaming, 35(3): 347-362

Smith, R. (2007) “Game Impact Theory: Five Forces That Are Driving the Adoption of Game Technologies within Multiple Established Industries”. Games and Society Yearbook.

Sowa, J. (2000) “Knowledge Representation: Logical, Philosophical, and Computational Foundations”. Brook/Cole, a division of Thomsom Learning: Pacific Grove, CA.

Susi, T., Johannesson, M., Backlund, P. (2007) “Serious Games – An Overview”. Technical Report HS-IKI-TR-07-001, School of Humanities and Informatics, University of Skövde, Sweden

Van Deursen, W.P.A. (1995) “Geographical Information Systems and Dynamic Models”. Ph.D. thesis, Utrecht University, NGS Publication 190, 198 pp. Electronically available through www.carthago.nl

Wolfson, O., Xu, B., Jiang, L., Chamberlain, S. (1998) “Moving Objects Databases: Issues and Solutions”. SSDBM, p. 111, 10th International Conference on Scientific and Statistical Database Management

Zagal, J.P., Rick, J., Hsi, I. (2006) “Collaborative games: Lessons learned from board games”, Simulation & Gaming, 37(1), 24-40