

Training Games and GIS

Marcelo G. Metello, Marco A. Casanova

Department of Informatics
Pontifical Catholic University of Rio de Janeiro

1 Introduction

By the end of the 20th century, the computer gaming industry already was one of the largest entertainment industries. Together with the movie industry, it attracted large investments in research in computer graphics, among other areas of computer science. As a parallel line of development, military and flight simulators also received huge investments. These simulators are akin to entertainment games in terms of their technical requirements.

More recently, the so-called *serious games* (Susi et al. 2007) also started to gain attention. The basic motivation lies in that the technology developed for games and simulators can also be applied to other areas, such as medicine, architecture, education, urban planning, and government (Smith 2007). The development of serious games poses specific challenges, however, since their requirements differ from those of entertainment games. In particular, serious games require simulation models that reproduce certain aspects of the real world. By contrast, entertainment games have much more freedom to simplify their real world model, which is convenient when developers face technical limitations.

We use the term *geospatial training games* to denote serious games designed for computer-based training that require the use of spatio-temporal data representing geographic features. Games designed for emergency response training are a typical example of this class of games (Metello et al. 2008).

This paper is organized as follows. Section 2 discusses four requirements that heavily influence the design of geospatial training games. Section 3 proposes a general architecture for such class of games. Section 4 focuses on how to model processes in this context. Section 5 provides an

example of a geospatial training game. Finally, Section 6 contains the conclusions.

2 Requirements for Geospatial Training Games

In this section, we discuss four requirements that heavily influence the design of geospatial training games: realistic user experience; interoperability with existing GIS; time flow control; player performance evaluation.

2.1 Realistic User Experience

Geospatial training games must offer players a realistic experience. This implies that the game must simulate real-world situations and must offer a multimedia user interface with a minimum degree of realism.

Indeed, although serious games and entertainment games may be built over the same technologies, they are essentially different with respect to the situation they simulate. Serious games must simulate real-world situations, while entertainment games do not.

As for the user interface of geospatial training games, the rendering of spatial data on the computer screen should have enough similarity with the geographical features depicted so as to help users reason spatially about the real world. Likewise, computer animation of spatio-temporal data should help users reason about dynamic geographic phenomena. Furthermore, animations may adopt different time scales without impairing how users assimilate the information. For example, if an animation shows in seconds the evolution in land use of a particular region during a period of some years, the animation must still give the user a sense of what happened with occupation of the region.

2.2 Interoperability with Existing GIS

Geospatial training games must interact with geographic information systems (GIS) to access spatio-temporal data.

Indeed, training games require the ability to use real-world data. In the case of geospatial training games, this means accessing the spatio-temporal data stored in a geographic information system. However, such systems have been traditionally more concerned with managing spatial data than spatio-temporal data.

In particular, among other problems, the technology developed for computer games focuses on displaying data in real time. A continuous attention on performance is needed to keep an acceptable frame rate for a better user experience. Furthermore, the game may potentially be multi-user, which makes the problem even more complicated. However, geographic information systems are not prepared to sustain the throughput geospatial training games require to refresh the data displayed. Therefore, specific indexing and caching mechanisms may be necessary to provide game-compatible frame rates (Metello et al. 2007).

2.3 Time Flow Control

Geospatial training games must offer control over the time flow. That is, they must offer the ability to stop the game, accelerate the pace of the game, and return the game to an earlier point in time.

Indeed, this requirement is important for the usability of training games. For example, even if an emergency situation may last for days, the simulation should obviously not take the same amount of time. Periods requiring no decision making should be fast-forwarded. Likewise, the ability to go back in time is also highly desirable to test alternative decisions.

To better support time flow control, the game may record simulation data. However, this requires investigating techniques to avoid two problems: an undue growth of the database; and slowing down the simulation.

Indeed, one should investigate alternative ways of storing spatio-temporal data other than storing a new version of a data item every time it is updated. As for the second problem, first note that storing simulation data may generate a massive number of database update requests. This poses an interesting problem on how much indexing should be used. Indexes can speed up queries, but they necessarily slow down update requests. In particular, spatial feature classification can play an important role with respect to optimizing the amount of indexing, if it can help differentiate features with high update frequency from nearly static ones.

2.4 Player Performance Evaluation

Geospatial training games must offer tools to evaluate the performance of the players.

Indeed, in the case of training games, player performance evaluation is often an important part of the learning cycle (Borodzicz and van Haperen 2002). A successful training system should improve the players' decision

making abilities. Therefore, the system must allow players to trace the results of the game back to their decisions (Zagal et al 2006).

If the evaluation process is totally manual, this requirement reduces to the ability of playing back a simulation, or at least the ability of providing a timeline registering the main events that occurred, including the players' decisions.

More sophisticated training games are aware of the players' action model. For example, in organizations with predefined procedures, the game may match the sequence of players' actions to the predefined procedures to check whether the players acted as expected. Going one step further, the game may be used to evaluate the effectiveness of the predefined procedures, detect their flaws and help with their evolution (Smith 2004). In this case, learning is not limited to the individual level, since we may consider that the evaluation of the effectiveness of the predefined procedures represents some kind of collective or institutional learning.

3 An Architecture for Geospatial Training Games

In this section we introduce a high level, modularized architecture for geospatial training games (see Fig. 1).

Indeed, serious games tend to rely on modular architectures (Westera et al. 2008). However, many entertainment games do not follow a modular architecture to achieve better performance. This is particularly true for action games, classified as *intensively performative* in Apperley (2006).

The *world representation* component is basically a data structure, stored in main memory or in any persistence device, which models a state of the

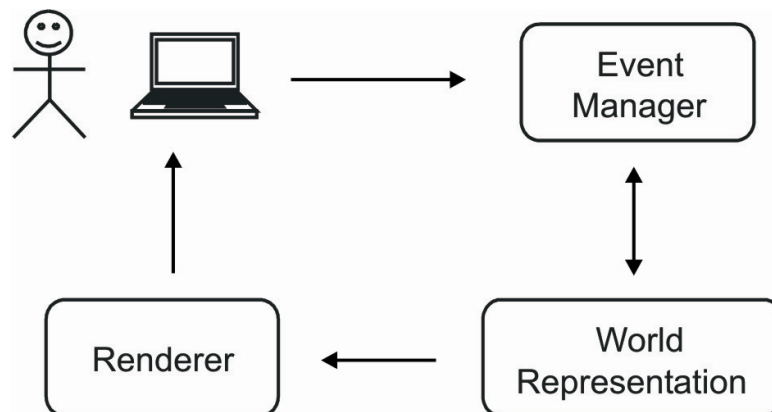


Fig. 1. The main components of an architecture for geospatial training games

simulated situation. It is a passive component since it does not act on anything and remains unchanged while not receiving external influence. The *event manager* is the component responsible for all activity in the game simulation. As the game flows, the event manager continuously updates the world representation. Finally, the *renderer* component is responsible for materializing the player view of the world.

All game activity is represented by *event streams*, as discussed in Section 4, in the sense that every change made to the world representation is carried out by some event. Both human player actions and simulation models generate event streams. The event manager is responsible for synchronizing all events and for leaving the world representation always in a consistent state.

Adding a human player as an interacting element in the simulation raises some interesting modeling challenges, as compared to fully automated methods traditionally used in GIS for simulations of dynamic phenomena. The human element is essentially different from all other simulated elements since his behavior is not deterministic. This fact makes it impossible to pre-compute the behavior of elements possibly affected by the actions of the player. For example, consider an emergency situation where some amount of oil leaked into the ocean. The dispersion of the oil on the water may involve complex calculations, which may be pre-computed. However, player actions, such as the placement of contention barriers, may affect oil dispersion, making the pre-computation useless in this case.

The non-deterministic behavior of human players requires more than simply modeling spatio-temporal data: it requires modeling processes, that may run in parallel and that may interfere with each other. In the previous example, the parallel processes are the dispersion of the oil and the execution of some contingency plan by the human player.

Another difficulty is that human player processes and computer-simulated processes are inherently asynchronous, as illustrated in Fig. 2. The situation is similar to that of database management systems, where different asynchronous transactions act on the same data items. In both cases, a synchronization mechanism is required to keep the data always in a consistent state.

Finally, the event manager must implement the synchronization mechanism in such a way as to facilitate time flow control, in special, rollback operations.

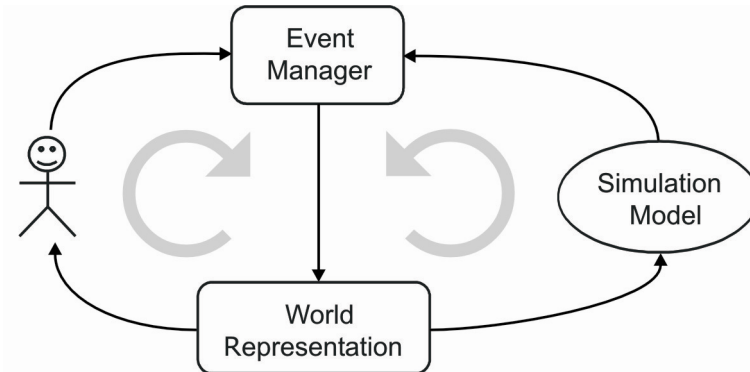


Fig. 2. Two inherently asynchronous processes: human player decision making (left) and automated simulation (right)

4 Process Modeling

We address in this section the central question of how to model processes. As usual, we classify simulation models with respect to their time representation as *discrete* and *continuous*, and we model player processes as workflows.

4.1 Discrete Process Modeling

In GIS, some of the most used dynamic modeling techniques, such as cellular automata (von Neumann 1966), are based on a discrete time scale. For each time t_i , the state S_i of the world representation is well defined. In this case, the perception of the process is represented as a sequence of states $S = (S_0, S_1, S_2, \dots, S_n)$.

Consider a continuous time interval $[a, b]$ such that $a \leq t_i \leq b$, for all $i \in [1, n]$. If we map the previously defined discrete time scale into this interval, the state of the simulation will be defined only for a few time values in the interval. Hence, this representation allows other processes to observe the state only at these few time values.

A first solution to the above problem is to force all processes that may possibly interfere to use compatible time scales. Therefore, every time one process needs to observe the state S_i of another process, S_i will be defined. By doing so, we are actually forcing the process models to be aware of the synchronization problem. The first drawback of this solution therefore is

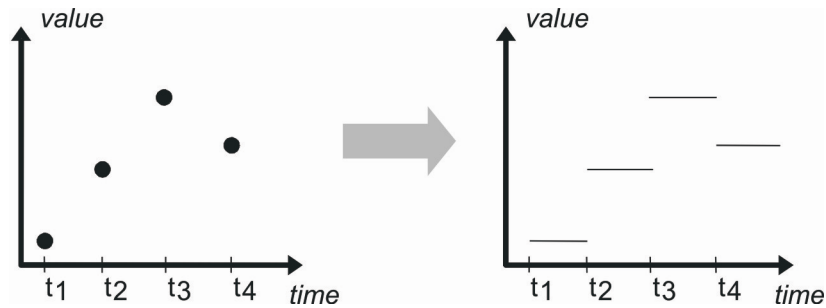


Fig. 3. Defining the state for all time values through a step function

that it introduces dependencies between the process models, which reduces the reusability of the models. A second drawback is that the solution is not compatible with other types of process models, such as continuous models.

An alternative solution is to interpolate the state between any two consecutive time values, which allows the state to be observed at any point in time during the execution of the process. The simplest form of interpolation is to define a step function f , in the sense that the state S_i remains unchanged during the interval $[t_i, t_{i+1})$. Fig. 3 illustrates a single-variable step function whose domain is the time interval in question. The advantage of this simple interpolation method is that, for $t \in [t_i, t_{i+1})$, $f(t)$ does not depend on S_{i+1} , which is essential, since S_{i+1} cannot be computed before t_{i+1} because other processes may interfere with the process in question during the interval $[t_i, t_{i+1})$.

However, the use of a step function f may introduce errors because other processes may produce new data that f does not take into account. If this error is not acceptable, the discrete model should adopt a finer time granularity. The hypothesis here is that the error can be reduced to any extent by increasing the granularity of the discrete model.

In summary, the approximation by step functions may also bring to the modeling stage certain questions related to process interaction. However, the impact should be less than in the first solution. Step functions may require processes models to be more precise, but they are never restricted with respect to their time scales.

4.2 Continuous Process Modeling

The most widely used continuous dynamic models are based on systems of differential equations (Lee and Zheng 2005). Such models are able to define the state at any point in a continuous time interval. If the state needs to

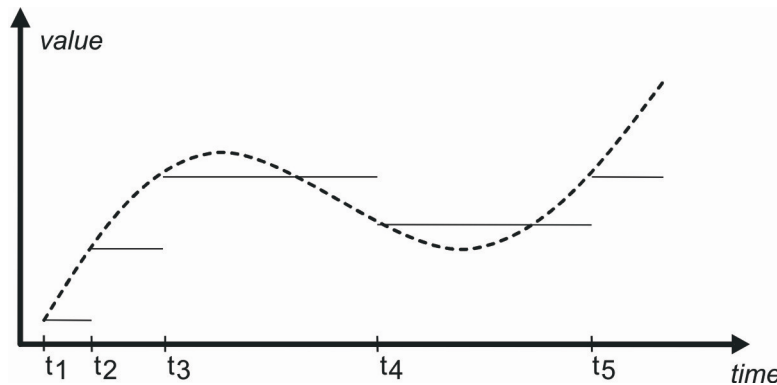


Fig. 4. A continuous function (dashed line) approximated by a step function (continuous lines)

be observed at time t , the system of equations is solved for t . Such systems provide potentially unbounded precision with respect to time.

This approach poses no problems for other asynchronous processes to observe the state at any point in time. However, it is somewhat limited because it only works with numeric floating point values. Besides, it is not clear how these models may be adapted to interact with other asynchronous processes.

In the scope of this paper, continuous process models will be restricted to simulated elements whose behavior are known in advance and which are not subject to change. In our oil leak example, the oil dispersion cannot be modeled as a continuous process simply because it is not known in advance where the contention barriers will be dropped.

It is also possible to map a continuous process model into a discrete one, by using step functions to approximate the functions that the equations define, as illustrated in Fig 4. Although making process interaction simpler, the infinite precision of continuous models is lost. Again, the granularity should be controlled to match the required precision, as discussed in Section 4.1.

4.3 Workflows

Workflows are widely used for business process modeling. For example, emergency plans may be modeled as workflows (Carvalho et al. 2001). Therefore, for player performance evaluation, it seems natural to model player activity as workflows with the purpose of comparing his actions with the predefined plans.

A workflow is essentially a set of actions and a control structure, which defines in what order actions should be executed in a given situation (van der Aalst 2003). Most workflow representations do not define the time at which the actions should be executed, and how long they will take to finish. In fact, many representations assume that actions are atomic.

Workflows do not necessarily define how actions affect the world. One approach is to define actions through their pre- and post-conditions, following the tradition of AI planning systems (Fikes and Nilsson 1971). But note that this representation still assumes actions to be atomic.

However, the assumption that actions are atomic may be too restrictive. For example, consider the action of walking. It may not be realistic to change the position of the character from the origin to its destination in one single instantaneous step. Instead, it is more realistic to simulate the trajectory of the character to the destination point through multiple state changes, so that his trajectory may be observed. If the world model requires that actions have duration and make changes to the world during their executions, we will have to model actions as processes, as described in Section 4.1 and 4.2.

4.4 The Event Framework

In this section, we outline how to reduce discrete process models (or discretized continuous processes) and workflow models to a unifying *event framework*.

The situation simulated in a game is entirely captured by the world representation and an event stream. From another perspective, the world representation and the event stream represent a basic separation between the static and the dynamic parts of the reality, an approach used in (Sowa 2000) to model knowledge about processes.

Intuitively, an *event* represents an activity carried out in a game. An event has a *start time* and an *end time*, which define the time interval during which the activity takes place. An event that has the start time equal to the end time is called an *instantaneous event*. Only instantaneous events are allowed to change the world representation. This restriction is necessary to keep a well-defined and observable state at all times.

All events must use the same *global time scale*. If some process is modeled in a *local time scale*, there should be a way to translate between the local and the global time scales.

To introduce abstraction levels, an event E , called a *parent event*, may have *child events*. Child events are restricted with respect to time so that no child event starts before or finishes after its parent.

In this even framework, a discrete process (or a discretized continuous process) is represented as a stream of instantaneous events, whose start times are the points of discontinuity and which change the state of the process. A workflow is represented as a single parent event and a stream of child events corresponding to its actions.

A detailed discussion on the synchronization model is beyond the scope of this paper. Basically, synchronization is achieved by serializing the event streams from all processes, respecting their start times. If two instantaneous events start at the same time, a tiebreak rule is used.

5 An Example of a Geospatial Training Game

In order to validate the proposed architecture, we implemented an emergency training game that simulates emergency situations in the context of oil terminals. The InfoPAE system (Carvalho et al. 2001) served as a basis for the implementation, since it includes a GIS module and is specifically designed to support emergency response.

5.1 Emergency Plans

To effectively handle emergency situations, teams must be trained to respond quickly and in a well organized manner. Practice has shown that this will happen only if the response strategies have been planned ahead of time, based on a risk analysis that considers the possible accidental scenarios. The strategies and procedures are usually documented as *emergency plans*, which comprise a set of instructions that outlines the steps that should be taken before, during, and after an emergency. A plan should also be backed up by a database of human and material resources, and a document repository.

Traditionally, emergency plans take a representational form. In more detail, to model knowledge about dynamic phenomena, Frasca (2003) discusses two different approaches: 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 an aircraft landing procedure. A representation of a specific landing could take the form of a video where an observer would be incapable of interfering. By contrast, a flight simulator would allow the user to interfere with the phenomenon in a way that simulates the real aircraft. This flexibility is only

possible due to the simulation characteristic of modeling the behavior of the elements, independently of any specific scenario.

In the context of the discussion in Section 4.3, the representational form of an emergency plan takes the form of a workflow. However, typically, the workflow does not model the dynamics of the actions in a realistic way, which would otherwise help emergency managers create more reliable plans. In other words, an emergency plan should take a hybrid form, where certain actions are coupled with simulations. For example, a plan may include an action that instructs the user to send two boats to place a barrier to contain an oil spill. If the action were backed up by a simulation of the sailing conditions of the boats (and by a simulation of the oil dispersion), it would be easier to determine where the boats should drop the barrier.

After the plan is developed, it should be tested to verify its effectiveness and to train and evaluate the operational readiness of the emergency teams. Testing usually takes the form of field exercises or drills, which can be very time consuming and expensive. Another point to consider is the difficulty of representing detailed and realistic situations, as required to effectively test the emergency plan. In such cases, the use of emergency training games can be very helpful.

5.2 Example of Running the Emergency Training Game

To exemplify the use of the emergency training game developed, consider an accidental scenario where a considerable volume of oil spills into a bay. This scenario was chosen because it involves simulation processes and player actions that interfere with each other. A fairly large number of applications developed for a company managing coastal oil terminals, using the InfoPAE system, in fact contain emergency plans that cover this type of scenario.

In a typical oil spill scenario, the initial goals are to attempt to control the oil 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 depend on the accidental scenario, whose description includes: the oil type and its characteristics; the location of the source of the leak and its nearest landmarks; an estimation of the amount of oil spilled; weather and sea conditions; and characteristics of the coastal area that might be affected.

We run a game that considered the oil spill scenario after the leak has stopped and that focused on oil contention. Clean-up operations for the oil

that reached the coastal areas were not considered. The goal of the game was to test the emergency plans for leaked oil containment to uncover possible flaws, as well as to help planning equipment installation and location.

The initial conditions of the game were specified in a document, and included the location, amount and type of leaked oil, maps of the nearby coastal areas, the location of all available equipment, and weather conditions.

The dynamic elements of the game were modeled as follows. The dispersion of the oil on the water was modeled by a cellular automaton, which considered: the current location of the oil spot; environmental conditions, such as wind direction and speed, defined globally; obstacles, such as coastal lines and containment barriers; the type of the contention barriers, which differ on their absorption and containment capabilities; the type of the coastal areas, which also differ on how they absorb oil (for example, beaches absorb much more oil than rocky coasts).

The state of each cell describes: the amount of oil in the cell; if the cell intersects a barrier or the coastal line, in which case the cell is considered an obstacle cell; the type of the obstacle, if it is the case. Note that the state of a cell varies according to amount of oil in it, and if a barrier happen to be dropped in the area the cell represents (coastal lines are supposed to be fixed, but this assumption might be revised to take into account the tide).

Boats were also considered dynamic elements of the game. Boats are responsible for dropping containment barriers and for carrying oil recovery equipment, such as pumps and skimmers. The initial location of the boats was defined in the document that described the initial conditions of the game.

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

The placement of a containment barrier is carried out by two boats, each one holding one end of the barrier. The command to place a barrier has parameters - the type of the barrier, the angle at which it should be put, the distance the boats should keep from each other, and the curvature that should be kept – and preconditions – the maximum distance between the two boats, the length of the barrier, and favorable environmental conditions, among others.

5.3 Benefits of using the Emergency Training Game

Some of the main benefits of using the emergency training game described in Section 5.2 include:

- to help finding flaws in the emergency plans
- to help testing whether available resources are sufficient to handle all accidental scenarios
- to help training emergency personnel and thereby improve their responsiveness
- to reduce costs, since simulation is obviously significantly cheaper than full scale exercises

6 Conclusions

We first listed four basic requirements that heavily influence the design of geospatial training games. Then, we introduced a very high-level architecture for geospatial training games that calls attention to the importance of simulating real-world phenomena and players' actions.

We moved on to discuss how to model processes, including players' actions. We introduced an event framework that integrates discrete modeling techniques used in GIS and workflows.

Finally, we described a prototype implementation of an emergency training game in the context of the InfoPAE emergency response system. Early results with the implementation indicated that the error introduced by step function approximations can be controlled by increasing the granularity of the event streams generated by the processes involved. The assumption proved valid for the case of oil dispersion and contention barrier placement. However, further work is required to define a generic process synchronization model that covers discrete processes and workflows.

References

- Apperley, T. (2006) "Genre and game studies: Toward a critical approach to video game genres". *Simulation & Gaming*, 37(1), 6-23
- Borodzicz, E.; 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

- Carvalho, M.T.; Freire, J.; Casanova, M.A. (2001) "The Architecture of an Emergency Plan Deployment System". IN: Proc. III Brazilian Symposium on Geoinformatics, Rio de Janeiro, Brazil.
- Fikes, R. E.; Nilsson, N. J. (1971). "STRIPS: A new approach to the application of theorem proving to problem solving". *Artificial Intelligence*, 2 (3-4).
- Frasca, G. (2003) "Simulation versus Narrative: Introduction to Ludology". In: Wolf & Perron (Eds.) *The Video Game Theory Reader*. Routledge.
- Lee, E.A.; Zheng, H. (2005) "Operational Semantics of Hybrid Systems". Invited paper in Proc. of Hybrid Systems: Computation and Control (HSCC) LNCS 3414: 25-53, Zurich, Switzerland.
- Metello, M.; Vera, M.; Lemos, M.; Masiero, L.; Carvalho, M.T.M. (2007) "Continuous Interaction with TDK Improving the User Experience in Terralib". In: Proc. IX Brazilian Symp. on GeoInformatics, Campos dos Jordão, Brazil.
- Metello, M.; Casanova, M.A.; Carvalho, M.T.M (2008) "Using Serious Game Techniques to Simulate Emergency Situations". In: Proc. X Brazilian Symposium on GeoInformatics, Rio de Janeiro, Brazil.
- 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 Thomson 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 der Aalst, W.M.P.; der Hofstede. A.H.M.; Kiepuszewski, B.; Barros, A.P. (2003) "Workflow Patterns". *Distributed and Parallel Databases*, 14(1): 5-51(47).
- von Neumann, J. (1966) *Theory of self-reproducing automata*. Illinois: A.W. Burks.
- Westera, W.; Nadolski, R.J.; Hummel, H.G.K.; Wopereis, I.G.J.H. (2008) "Serious games for higher education: a framework for reducing design complexity". *Journal of Computer Assisted Learning*, 24: 420-432.
- Zagal, J.P.; Rick, J.; his, I. (2006) "Collaborative games: Lessons learned from board games". *Simulation & Gaming*, 37(1): 24-40.