SimPort: a multiplayer management game framework

Jeroen Warmerdam[1,2], Maxim Knepflé[1,2], Rafael Bidarra[1], Geertje Bekebrede[3], Igor Mayer[3]

[1] Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
P.O. Box 5031
2600 GA Delft, The Netherlands
r.bidarra@ewi.tudelft.nl

[2] Tygron – Serious Gaming & Media
Brasserskade 50
2612 CE Delft, The Netherlands
j.warmerdam@tygron.nl
m.g.knepfle@tygron.nl

[3] Faculty of Technology, Policy and Management
Delft University of Technology
P.O. Box 5015
2600 GA Delft, the Netherlands
g.bekebrede@tbm.tudelft.nl
i.s.mayer@tbm.tudelft.nl

Keywords

Serious games, Management games, Multiplayer games, Game framework, Port Planning

Abstract

The serious games industry needs game engines, or frameworks, that have been developed specifically with this sector in mind. This paper discusses the benefits of creating a serious gaming engine. This is demonstrated on the basis of SimPort, a novel multiplayer management game framework. SimPort has shown to be very powerful, functional and easy to use in the development of MV2, a game module simulating a major expansion project, the Maasvlakte 2 area, of the Port of Rotterdam in the Netherlands [8]. In addition, from MV2 real-life usage so far, players and tutors have concluded that this game is not only educational, but also a lot of fun to play.

1 Introduction

Playing computer games is becoming an increasingly popular activity in day-to-day life. This has led to considering the use of games for purposes other than simple entertainment, like education, management, decision and policy-support.

As serious games become a normal activity, different genres will appear and become accepted as such. This paper focuses on the Management Game genre (Section 2), briefly describing the current state-of-the-art. The core of the paper discusses the necessity for specific game engines that simplify and speed up the production of Management Games, and presents SimPort as an example of how this has been realised [10] (Section 3). Subsequently, some choices that were made during the production of SimPort are presented (Section 4)

Next, a description of how the engine evolved into its current state is presented, (Section 5), together with a practical evaluation of how real-life players experienced a game created with this engine. We also present recommendations for further research (Section 6). Finally, we draw some conclusions (Section 7).

2 Management Games

It is important to define what exactly the relevance of playing management games [9] is. For many decades, computers have assisted management somehow, but the role of computer games is essentially different. First this difference will be addressed, followed by an explanation of the aforementioned importance. Finally, two example games are presented and their relation is defined.

2.1 Decision visualisation

A well-known application of a computer system during the production and design of a complex system is the Decision Support application. Turban described a Decision Support System (DSS) as follows: *"an interactive, flexible, and adaptable computer-based information system, especially developed for supporting the solution of a non-structured management problem for improved decision making. It utilizes data, provides an easy-to-use interface, and allows for the decision maker's own insights."* [7].

When creating a management game, it is important to avoid that the players get the feeling that they are dealing with a DSS; instead, it should always be apparent that they are playing a game. Playing a game puts a person in a different mindset than when working on simulations. If this is not handled properly, not only are the benefits that a game has over a DSS lost, the benefits that a DSS has also remain unachieved. To avoid this, a designer should opt for a computer game that does not necessarily support the player in making decisions. Instead, he should create a game that visualises the results of players' choices as they are made. By visualising the results, players see the effects of their choices and become more immersed in the game.

2.2 The rise of management games

Designers and managers need to have a better understanding of complex system behaviour. To this end, they can benefit from experiencing the long-term behaviour and unanticipated and sometimes undesirable consequences of their complex system design before it has been implemented. The ability to manage complexity begins with an awareness of and insights in the nature of complex systems. [4] Direct experience of what can happen or what can go wrong is often very effective for raising awareness and insights. Nevertheless, in most cases, such direct experience is difficult to obtain from real infrastructures or other real-world systems without possibly serious consequences. Therefore, games are a

good substitute because they can generate learning experiences in a relatively fast and safe manner.

It is important to realise that there is an increasing trend to use Management Games in a large variety of settings. This is evident from the increasing interest in the subject in the media and research. Therefore, it is crucial that the development and production of such games be facilitated and assisted by adequate tools and techniques.

2.3 SimMV2

In late 2004 Delft University of Technology and the Port of Rotterdam combined their efforts to create a management game about extending the port of Rotterdam. This resulted in the first version of SimMV2 in the summer of 2005. In this game three players build the new port extension in 30 years simulated time.

The initial target audience where port experts. But students at the University also proved to be a good audience resulting in an actively played game. The game was initially setup as exclusively web-based, using mostly Flash [5] and Java Enterprise [6] in a web browser, therefore with limited capabilities. More features where being requested conflicting with the already growing limitations and network overhead. The need for a new version became clear.

2.4 SimPort

In late 2005 the two lead programmers of SimMV2 founded a new company called Tygron, and started construction of the next generation called SimPort. Having detailed knowledge of the possibilities and limitations of the SimMV2 game, they came up with a new framework. SimPort is totally built in Java [6] from bottom up, giving it more possibilities.
SimPort runs on any system that supports Java 5.0 and for which LWJGL [11] native interfaces exist.
SimPort was developed in house, together with the Delft University of Technology and the Port of Rotterdam, due to the lack of Java gaming engines and the price tags associated with other commercial engines.

3 Architecture

This section deals with the benefits of an engine for serious games. For this, the SimPort Framework is used as an example.

3.1 Benefits of a serious games engine

The production time of a serious game is often shorter than that of the average entertainment game, because the subject of the game, the client and the message that it conveys are, themselves, limited by time constraints. This is one of the major factors that hold the visual quality of the serious games back from their entertainment counterparts. As a result, developers of serious games will need to use an existing game engine, or framework, if they want to keep to this short production time, while still delivering a decent looking game.

The entertainment industry has provided developers with several game engines to produce their games, most of which come at a very steep price. These engines allow developers to focus their production on factors that make their game unique, and handle basic aspects like walking and collision detection for them. Much of this functionality will not be used in a serious game and, instead, a lot of time will still be spent on developing serious game-specific elements.

A game engine that has been targeted at serious game developers will do the same as the entertainment equivalent, except that (i) it also delivers those elements that are not present in entertainment games and (ii) it leaves those elements out that are not necessary. The former allows for more interesting development as developers can get to the actual game creation, while the latter makes for a cleaner and lighter package.

3.2 The Framework

The Framework needs to provide a number of features for it to be useful. For example, to allow for multiplayer sessions, all network communication needs to be handled by the Framework. Also, it needs to be expandable, to allow for development of game specific elements. The SimPort Framework fulfils both of these criteria. In this subsection both the server side and the client side of the Framework are described.

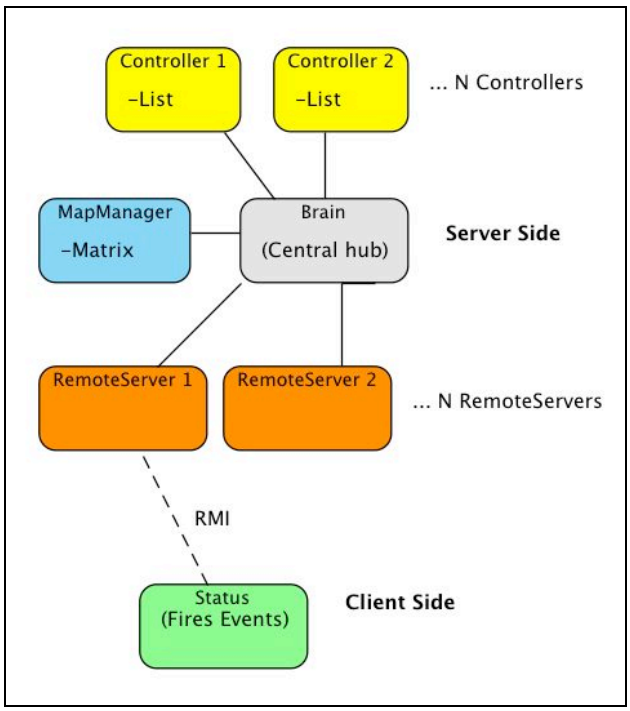

Figure 1. Architecture Overview

3.2.1 The Server

The SimPort Framework has been set up from the beginning as a multiplayer game framework, and combines a number of general-purpose parts. Functions like saving a game session to Extensible Markup Language [2] (XML) data and the handling of concurrency problems are typically done the same way

for every possible game implementation. This provides the designer the freedom to concentrate on the more game-specific elements of the game. A central controller object, called the Brain, allows the game developer to attach game specific Controller object. The game designer can extend a Controller object at will, to his own liking in order to make it suitable for his specific needs. Each Controller contains a list that can be filled with game specific objects extending the Item objects. By means of annotation [3], it can be defined which variable in the Item extending object should also be saved to XML data. An example of such a controller is the CustomerControl. It can have methods like "contact this customer" and keep a list of all the customers. There is also a MapManager if the game contains a map of its world. When the Framework is started these controllers will be initialized and the data in the list is filled from a saved XML file. The Framework also synchronizes the data in these lists with the client computers.

Besides the controllers, a server object must be extended. This object is a Remote Method Invocation [1] (RMI) object and is used by the clients to give their commands. Each method can connect to a controller in this way to execute commands in a manner that is concurrently safe.

3.2.2 The Client

The Framework can have multiple clients connected to it that do not necessarily have to have the same functionality. One client can be projecting a view of the game world to a wall, without other interactions, while another client can be used as the game interface for the players. Each of these clients has a Status object containing synchronised data from the server. One of these data instances is an abstract map. The developer has the option of implementing a game specific extension of this map. The extension defines the way the map is displayed. The Framework takes care of the actual rendering and updating. Any other client-side objects can subscribe to the Status object to receive game-wide updates.

The core of the client-side Framework is fairly rigid, as it has a wide range of built-in basic game interaction. Around this core the majority of the features are customizable. A central manager arranges the order in which the game progresses through different states. The developer can add states to this controller and control the different phases that the game moves through. This allows e.g. for customisation of introduction movies, evaluation screens and menu structures.

4 Implementation choices

The production of a game framework touches on many different subjects, because many different problems need to be solved. It is important to stay flexible, so as not to limit the range of applicability. On the other hand, it is important to moderate this flexibility so that developers have the tools they need to focus on the core of their games.

This section details two of the major choices that were made in the design and implementation of the SimPort game framework. Though these subjects are not the only choices that were made, they are two of the more interesting ones: (i) the Level Of Detail strategy chosen, and (ii) the manner in which data is kept synchronized between the clients and the server, by means of versioning.

4.1 Level of Detail strategy

One of the features that makes the client side of the SimPort Framework so interesting is the possibility to navigate through a three dimensional representation of the game world. This added functionality brings with it a certain level of complexity as the world is rendered. The framework tries to limit the load of this rendering as much as possible by a so-called Level Of Detail (LOD) strategy. Objects that are further away from the viewer need less detail for them to stay visually pleasing. The goal of a good LOD strategy is to be intelligent about what LOD is chosen for each object.

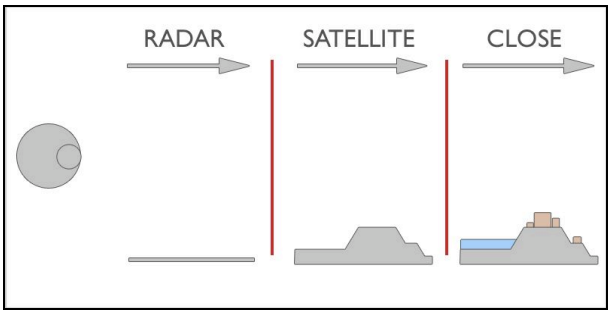

Figure 2. The LOD strategy

The current solution entails two views of the map. The first is an overview, as though the player was looking at a planning map. Company buildings and terrains area visible as outlined areas, with no structures visible. As the player comes closer, the view changes to something resembling a satellite view. The texture of the terrains becomes visible, as does the water, but still no buildings are visible.

Figure 3. The Radar View

The radar view is nothing more than a textured square, greatly reducing the necessary calculations. The

satellite view adapts according to the more traditional LOD strategies, as the map is broken up into smaller scene elements, which are only rendered when they are on the screen. This strategy works well, because when the view switches, not the entire map but only a part of it is visible.

The buildings in the world are handled by a very basic LOD strategy. When the viewer is far away from the building, the building is not shown at all. Only when the view comes close enough do the buildings become visible. As the framework goes through further iterations the same strategy that is used for the map can be used on the buildings, allowing for a more gradual transition from invisible to fully visible.

4.2 *Versioning*

Keeping versions of objects in the game is required to keep all client computers in phase with each other. When objects update a new version is spread. These objects are located in lists in the Controllers and map matrix. All objects have in common that they extend the Item object. An Item keeps track of its version number and the object's ID, which is also its position in the list. Each Controller also keeps an overall list version number.

To update to the latest version a client computer sends a version request to the server. To keep traffic to a minimum the client sends only one array containing integer numbers. The first integer represents the client's overall version status. This number is calculated by summing up all version numbers of each list and the map matrix. At the server side this overall number is compared to an overall server number. If these are not the same an update must have occurred, because if an object's number is updated the version number of the list is increased as well. When the to be updated list is found, the system cycles through it and takes out the objects with a version number higher than the client's version number. Finally the updated objects are sent back in small lists to the client. In most cases this results in comparing two numbers and sending a predefined array of integers. In case of an update only the updated object are sent. This can be different for each client computer. It is also possible for a client's Status object to make a subscription to only receive list updates or map updates. This is particularly useful, for example, in a controller application where no map is needed.

When the client's Status object is up to date, it is time to let the game know. Firing custom events to different parts of the game, containing the total list, does this. The map has a special event containing only the updated blocks.

5 SimPort Evaluation

In the previous sections we explained the importance of a serious game engine and introduced the SimPort Framework. This section will briefly describe how this engine came into existence and what its role can be in the serious game sector.

The creation of SimPort was triggered by the production of the game SIM Maasvlakte 2 (SimMV2), which, in turn, inspired the production of SimPort and its SimPort MV2 module.

5.1 *SimPort MV2*

The framework makes it possible to develop more serious games, of a similar nature, in shorter time. SimPort is not limited at all to the port area in Rotterdam but other ports can also be played. Furthermore, the application is not browser-based anymore, and supports 3D graphics and GUI, while maintaining LAN and web-based networking.

Figure 4. SimPort Game View

At the moment SimPort is ending its development cycle and the first games are being played at the University. In late 2006 the first commercial version will be released

5.2 *Real-life experiments*

SimPort and its predecessor have both been used in real-life situations. "Preliminary findings suggest that the game is of high quality, that players enjoy it and find it educational and instructive" [4] was the evaluation after the first few sessions of SimMV2 and the first life test session of SimPort as raised similar reactions. The game's original target group was experts, but it has shown its worth in other situations too.

Figure 5. Players at Work

Observations during the game and the results from questionnaires indicate that generally the players very

much enjoyed the game. The game's degree of immersion was fairly good: players had to be urged to stop for a lunch break a couple of times, and then found that they had come back early to start playing again. The active presence of two contacts from the Port of Rotterdam in one of the student sessions seemed to engage the students even more and improve their performance. It triggered interesting discussions and interactions between the professionals and the students, and also demonstrated the value of the game for education and training [4]. For a more detailed analysis and quantification, please refer to [4].

With a serious game like SimPort MV2 it is easy to fall into a simulation/decision support mood. It is important to have the players be caught in the gaming attitude, to achieve the learning benefits of playing the game. Players were very active during every phase of the game, and they fell into their respective roles easily. A game needs to be enjoyable and SimPort MV2 has definitely succeeded in this.

6 Recommendations

During SimPort's production cycle a number of questions arose pertaining to the creation of an all-encompassing serious game engine. The criteria that such an all-round engine would need to adhere to needs to undergo further research. It will be necessary to study whether the design of a catchall engine is truly viable, or whether it might be more productive to design a small number of separate, more specialised, engines. While the former is preferable, research is needed to determine whether it is viable.

7 Conclusions

The increasingly widespread use of computer games for purposes other than entertainment, so-called serious games, is finding an important niche in a variety of management applications. However, the design and development of such games cannot follow the same long production cycles of current commercial entertainment games. Therefore, all tools and techniques that help to concentrate on the central management game and simulation aspects are more than welcome by the serious game developers. In this paper we described a platform independent game framework that was specifically developed for this purpose. As such, it is especially suited for developing new management games, as it avoids having to start from scratch each time.

The framework was first utilized in the implementation of the SimPort MV2 module, a multiplayer game simulating the expansion process of the Port of Rotterdam along a period of 30 years. This game has been played several times by students and professionals together at the Delft University of Technology, Erasmus University and the Port of Rotterdam.

Out of these real-life experiments, players said they considered the game of high quality, they enjoyed it and found it quite instructive. Student players emphasized that they learned much about the complexity of the port project, while professionals stressed that the game can enhance communication and cooperation with the authorities of the Port of Rotterdam. But probably most important of all is the fact that the game provided an opportunity for professionals and students to look at the future of a complex project in an engaging and entertaining way; thus creating the mindset needed to learn from games.

8 Acknowledgements

The authors wish to thank the Port of Rotterdam for its continuous support and knowledge in developing these new games. Especially we wish to thank Anne-Kirsten Meijer, Jan-Willem Koeman, and Maurits van Schuylenburg for their contributions.

Edwin Branbergen and Hasso Schaap, students at the Faculty of Industrial Design at TU Delft, for developing the user interfaces of the SimPort MV2 game. Alexander Hofstede, colleague at the Faculty of Technology, Policy and Management (TBM) at TU Delft, for his outside expertise and help during stressful times.

We also wish to thank Gijs Buijsrogge and Teun Veldhuizen for assisting in the design of the original game SimMV2 basis of SimPort.

9 References

1. Pitt, E. and McNiff, K., 2001. Java.rmi The Remote Method Invocation Guide, Addison-Wesley, Edinburgh Gate UK
2. Extensible Markup Language (http://www.w3.org/XML/): XML Standard website.
3. Annotations (http://www.developer.com/java/other/article.php/3556176): Website on how to make custom annotations.
4. Bekebrede, G. and Mayer, I. (2006) 'Build your seaport in a game and learn about complex systems', J. Design Research. pp 2 & 25.
5. Flash (http://www.macromedia.com): Macromedia Flash website.
6. Java (http://java.sun.org): Sun's Java development website.
7. Turban, E. (1995). Decision support and expert systems : management support systems. Englewood Cliffs, N.J., Prentice Hall.
8. Port of Rotterdam (2004). Projectorganisatie Maasvlakte 2, Project initiatie document (Project Organization Maasvlakte 2, Project initiation document). Rotterdam, Port of Rotterdam: pp 87.
9. Duke, R. D. and J. L. A. Geurts (2004). Policy Games for Strategic Management: Pathways into the unknown. Amsterdam, Dutch University Press.
10. Mayer, I., Bockstael-Blok, W. & Valentin, E. (2004) A building block approach to simulation. An evaluation using Containers Adrift, In: Simulation and Gaming, 35 (1) pp 29-52. ISSN: 1046-8781.

11. Lightweight Java Gaming Library (http://www.lwjgl.org).

11 Biography

Jeroen Warmerdam is an MSc student in the faculty of Electrical Engineering, Mathematics and Computer Science of Delft University of Technology, The Netherlands. He is co-founder of Tygron and has been working on the SimPort project after starting its development during his BSc project.

Maxim G. Knepflé is an MSc student in the faculty of Electrical Engineering, Mathematics and Computer Science of Delft University of Technology, The Netherlands. He is co-founder of Tygron and has been working on the SimPort project after starting its development during his BSc project.

Rafael Bidarra is assistant professor Geometric Modelling at the Faculty of Electrical Engineering, Mathematics and Computer Science of Delft University of Technology, The Netherlands. He graduated in electronics engineering at the University of Coimbra, Portugal, in 1987, and received his PhD in computer science from Delft University of Technology in 1999. He teaches several courses on computer games within the CS programme 'Media and Knowledge Engineering', and leads the research work on computer games at the Computer Graphics and CAD/CAM Group. His current research interests in this area include procedural and parametric modelling, and advanced techniques for animation and path finding. He has published many papers in international journals, books and conference proceedings, and has served as member of several program committees.

Geertje Bekebrede is a PhD researcher in the faculty of Technology, Policy and Management at Delft University of Technology, the Netherlands.

Igor S. Mayer is an associate professor in the faculty of Technology, Policy and Management at Delft University of Technology, the Netherlands. He is also a director of the Delft Rotterdam Centre for Process Management and Simulation (www.cps.tbm.tudelft.nl). He is a co-founder and board member of Saganet – the Netherlands' Simulation and Gaming Association - and a member of the Netherlands Institute of Government (NIG).