Design and Implementation of the Jomini Engine: Towards a historical Massively Multiplayer Online Role-Playing Game

David Bond[1], Hans-Wolfgang Loidl[1],Sandy Louchart[2]
[1] Heriot-Watt University, Riccarton Campus, EH14 4AS, Edinburgh, UK
{D.A.Bond, H.W.Loidl}@hw.ac.uk
[2] Digital Design Studio, The Glasgow School of Art, The Hub, Pacific Quay, G51 1EA, UK
s.louchart@gsa.ac.uk

Abstract. This technical paper describes the design and implementation of a game engine for historical massive multiplayer online role-playing games (MMORPGs). We explore the game and system design space with a focus on historical accuracy and provide a detailed discussion of key design goals for the support of a large-scale, distributed and scalable MMORPG. The *JominiEngine* is a modular and extensible system architecture built on C# and integrates Riak, a non-SQL database, at the core of the persistent data store in order to facilitate scalability. "*Overlord*", the first instance of the engine is set in the medieval time period and uses rigorous UML design methodology for both game and system design. In order to enhance the immersive gaming experience, a separate Unity-based client can be used to interact with the game engine..

1 Introduction

The benefits of exploiting game technologies for learning and education are well documented and substantiated by current and predicted growths in both the serious games and gamification industries [1]. Researchers have highlighted the benefits of learning through digital games and simulations. According to Gee, digital games provide context for learning and through hard fun [2], a desired experiential state in entertainment gaming leading to the experience of flow [3], which can facilitate effective learning. Gee also highlighted similarities between models of learning in education and game mastery [4], the capacity for games to support deep learning [5] and offering a safe environment where the consequences of failure are lower than in traditional education approaches [2]. Lee et al [6] extended on Gikas and Van Eck [7] and mapped gaming genre and activities across Gagne's intellectual skills [8] and Bloom's taxonomy of the cognitive domain [9]. In this context, MMORPGs provide the added benefit of engaging large groups of remotely located users [10] and, through their design, facilitate collaborative learning approaches [11]. They also offer the opportunity to collaboratively engage students with a wide range of learning domains and convey experiential learning [12] through an immersive simulation environment. For instance, Second-Life has been widely used in various educational disciplines [13] and Lee et al [6] investigated a Massively Multi-Player Online

approach (MMO) towards supporting English [14] and History education in the Rochester Castle MMORPG [15].

In this technical paper, we describe the *JominiEngine*, a game engine framework exploring the design principles of modularity and scalability towards developing a fully-fledged educational MMORPG. We discuss the core game engine with regards to the implementation of a concrete game model. The causal and dynamic nature of the history and heritage domain maps well on existing MMOPRG mechanics and "*Overlord: Age of Magna Carta*" ("*Overlord"* for short), our RPG, has a medieval setting (1194-1215). It illustrates the instantiation of the *JominiEngine* in relation to a concrete, playable experience and system design principles, which we believe to be essential for a successful educational MMORPG deployment. We thus establish the current context for historical MMORPG development through an educational and technical review of current implementations before we explore the primary technical issues related to the provision of scalability for a large playable universe. Finally, we discuss the game model and system design with regards to latency considerations, support for redundancy, distributed architectures and inter-operability with modern commercially available game engines. We conclude with a discussion of our current implementation of the *JominiEngine*, its planned future developments and its potential as a teaching and learning support tool.

2 Related Work

Historic Wargames (HW) represent a significant part of well-established game traditions that includes both wargames and war simulations. The wargame genre pre-dates digital gaming and has traditionally been rooted in analogue game-play, such as historical and fantasy/sci-fi simulations. The recent digitization of the genre has opened up new avenues, possibilities and a renewed appeal to current generations of gamers. Beside its entertainment values, Kirschenbaum [16] points to wargames as relevant tools for teaching and reinforcing creative decision-making; a core aspect of wargaming, recognised since Georg von Reisswitz first introduced his wargaming rules in the early 19th Century [17]. HWs represent a popular genre and an increasing number of commercial history-based games have become available, many based on historical conflicts. These games often contain a high degree of historical accuracy and vary both in, firstly, their level of abstraction (from 'traditional' wargames in which units are abstractly represented as counters on a hexagon map, to those that use cutting edge animation technology to allow the player to participate in battles) and, secondly, scale (from the modeling of a single battle, to the simulation of entire civilisations and time periods). HW's also often include role-playing elements; for example, the Crusader Kings series (which uses Paradox's ClausewitzEngine) [18]. Although, not historically accurate, Clash of Clans [19], a 'freemiuim' mobile strategy MMO, was ranked 3rd (in 2013) in the global list of iOS/Android apps revenues [20].

Kirschenbaum described wargames as "a vehicle for its participants, either through role-playing or the arbitrary rule-based constraints of the game world, to critically examine their own assumptions and decision-making processes" [16]. This is

supported by a broadening range of simulations in domains such as management or cultural heritage. Anderson et al. [21] investigated the use of serious games in the field of cultural heritage and drew attention to the increasing provision of modding tools as an intrinsic part of many commercial games, allowing for them to be adapted for educational purposes. From a design perspective, accuracy is a key feature for any wargame or historical simulation. Dunnigan argued for realism in HWs [22] but also made a case for balancing historical accuracy with fun gameplay and advanced the notion of "dynamic potential" (i.e. mechanisms through which the player might interact with the game world in order to alter history) in order to better integrate both aspects of the design. Whilst this places an extra burden on the game designer who needs to ensure accuracy, both in game data and mechanisms, it allows for a contextual positioning of relevant aspects such as geography, society, and military technology and doctrine. With regards to HWs specifically, two essential sources for wargaming, emerged from the aftermath of the Napoleonic wars: von Clausewitz [23] wrote on the principles and conduct of warfare, covering topics such as the foundations of strategy, the importance of information and planning, and the best uses of offense and defense and von Reiswitz [17] produced what is regarded as the first modern set of wargaming rules, aimed specifically at the military profession. It defines a rigid set of rules and charts that can be used to simulate the events and effects of battle. In the specific context of developing *Overlord*, we further referred to relevant authorities, including Sumption [24], and Nofi and Dunnigan [25] in representing the Hundred Years War, Oman [26] in examining medieval military practices, Ross [27] in designing game-world data structures for MMORPGs, and Adams [28] for the development of in-games formulas for the development of game mechanics.

Most existing MMORPG engines use a relational database management system (RDBMS) as the underlying data storage mechanism. This design choice profits from guarantees in terms of atomicity, consistency, isolation, and durability (ACID) but incurs extra runtime costs to deliver these guarantees. This can lead to delays, hampering the user experiences: for the EVE engine, Emilsson [29] states that *"the main bottleneck that we have had to overcome is I/O performance of database storage"* and reports typically 2500 transactions per second. To address this problem, the game-world in EVE is partitioned into regions, each served by its own SQL server, but all of them comprising a single universe (or "shard") and utilizing a single back-end RDBMS. This internal separation avoids a bottleneck, but requires regular data exchange on the fringes of regions. Other techniques have been devised to allow "caching" of frequently used data, avoiding RDBMS access in these cases. Again, this leads to added complexity in the implementation, and the efficiency depends on the choice of which data to cache. One notable survey of storage methods in MMOs compares MySQL, with the CouchDB and Riak, both NoSQL (non-relational) databases [30]. The results show that both NoSQL databases outperformed the SQL database: with 200 concurrent players by 7.6% for CouchDB and 23% for Riak. Another study [31] compared a range of NoSQL databases with the Microsoft SQL Server RDBMS, looking at the performance of basic operations (create, read, write, delete, fetch keys). The study summarises that, whilst some NoSQL databases (e.g. MongoDB) consistently outperformed MS SQL for almost all operation types (for some write operations by a factor of 10), overall the results varied greatly across

products, and across different operations, indicating a need for more research in this area.

3 The *JominiEngine*

Providing scalability for an MMORPG means catering for thousands of players and a large, playable universe. We address these issues primarily through the following choices in our system design: (i) we use a non-relational (NoSQL) database-management system (Riak) in order to reduce latencies in accessing the underlying (global) database; (ii) we build on Riak's support for redundancy, through transparent data-replication on distributed architectures; and (iii) we use a main-stream, general-purpose programming language, C#, with good inter-operability to the data-base and to other widely available game engines, such as Unity. The design of a new game engine, rather than extending an existing one, gives the opportunity to address scalability and modularity into the design from the start, exploring design choices for the system and the use of the engine itself. Throughout the system and game design we use UML diagrams for modelling (complete diagrams are included in Bond [32]). We primarily focus on technical issues first in order to build a generic engine that can instantiate rules flexibly and be deployed across a range of domains. Fig 1 (below) provides an overview of the game model and the central components; actors (NPC, PCS), relationships and management of resources (in-game currency, tradable/non-tradable) and conflict resolution, which in the historical context means combat, with armies as a tradable resource.

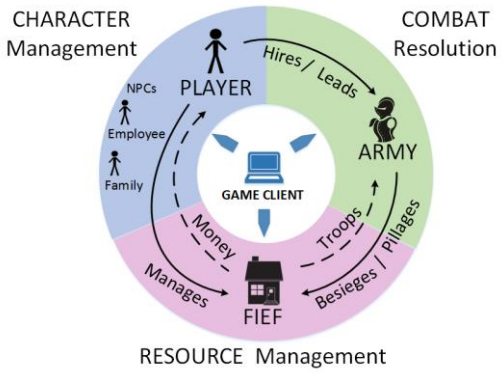

Fig.1: Basic game model

The *JominiEngine* combines real-time interaction and turn-based mechanics, where specific actions such as conflict resolution trigger immediate events, and state changes which affect fundamental game resources are processed during an 'update', which calculates the impact of resource management policies over one game turn (typically one season), and updates available resources (e.g. money, tax income). Additionally, communications are out-sourced to external mechanisms such as using bulletin boards or social networking infrastructure (player-to-player) or expressed

internally for aspects specific to the game engine (game-to-player). Thus, game events are communicated to players either immediately when events happen (synchronously) or at a later point when an update is performed (asynchronously). Similarly, administrative functions such as user management are out-sourced to an external engine. A tailored administrative interface to the component is still necessary, and currently implemented via an admin client.

3.1 Technical Considerations

Investigating a Distributed NoSQL database for Scalability: One design principle is to avoid bottlenecks in accessing the persistent data storage.. We therefore opted for a low-latency, NoSQL solution over a more main-stream SQL database, as the strong consistency guarantees offered by the latter is often not fully required in a digital game context. In contrast, low database latency (i.e. the response time to a request for updating the game data), is very important for an enjoyable gaming experience and for the scalability to thousands of players. These requirements are similar to those in big data applications, where consistency is sometimes weakened for performance. Thus, NoSQL databases have been developed that often provide a simpler view of a distributed, key-value store, without a full-blown SQL query language. Riak [33] is one such database and additionally provides built-in redundancy and the possibility to distribute the database over a network of nodes, both highly desirable aspects for MMORPG game systems.

Ensuring Modularity in the *JominiEngine*: The *JominiEngine* is composed of several key components (Fig 1), namely; combat resolution, character management, and resource management. The representation of the relationships between these components and game interfaces are critical in determining the modularity of the overall design. As such, we followed an object-oriented design throughout in order to isolate data that is internal to main game objects (i.e. PCs, fiefs) and define a set of operations that can be performed on these objects, thus ensuring modularity through the systematic definition of an engine-internal API for the main operations of the sub-components in the engine. This approach is advantageous in that it is relevant to large-scale system design and offer opportunities for future deployment such as the embedding of a simple scripting language such as Lua [34] into the engine, which would simplify content authoring and increase mod-ability of the engine.

3.2 Game model

Overlord serves to illustrate the *JominiEngine* game model and its three main areas of functionality (i.e. fief management, combat & army management, and household management - Fig 1). These are represented, in the game model through the modeling of time, goals and resources management.

Time model: As previously discussed, the *JominiEngine* time model is a hybrid of real-time and turn-based and allows for a combination of real-time cooperation between players, whilst also providing a flexible management of time (‘days’ in the game). where it is used as a resource that can be ‘spent’ on various actions.

Goals: A game should have concrete goals, leading to an achievable victory [35], but it should also provide a degree of freedom for its players [36]. Additionally, many players of historical games are looking to compare their own performance with that of historical figures [23]. The *JominiEngine* was designed to provide a variety of victory conditions, encouraging teamplay, and allowing victory to be decided based on total victory, the achievement of historical victory conditions, or team scores.

Resources: Game resources management is fundamental to enjoyment, game dynamics and the way in which players interact. In a historically-based game, resources must also be depicted authentically. Crawford [37] suggests two main techniques for allocating resources: i) asymmetric, in which each player is assigned a unique combination of resources; ii) symmetric, in which resources are allocated equally, and victory depends purely upon execution. The former is often more interesting but tends to be more difficult to program, as goals need to be more finely balanced. Table 1 (below) shows the resources identified for *Overlord*. Given its historical setting, the asymmetric model was chosen so that players must 'strategise' in order to ensure the most effective use of resources (this is also a historically accurate modeling of resources allocation).

Resource	Increased by	Reduced by	Used for
Money ‡	Fief income; pillage/siege; borrowing	Pillage/siege; lending	Recruitment; NPCs; fief expenses; family expenses
Population	Population growth	Pillage/siege	Recruitment; fief productivity
Game days	Passage of time (seasonal update)	Negative skills	Performance of game actions
NPCs	Birth (family); re-spawning (non-family)	Death, hiring by another player	Assuming roles of responsibility
Troops ‡	Recruitment, transfer between players	Combat, transfer between players	Combat
Titles ‡	Gaining territories, transfer between players	Losing territories, transfer between players	In-game status

‡ *denotes resource is tradable between players*

Table 1: Game resources and their characteristics

Mechanics: Using Crawford's approach [36], an initial list of verbs was compiled in order to define core game mechanics for the *JominiEngine* which could then be translated more readily into object-oriented programming structures and functions. As suggested by Sicart [35], the core mechanics were subdivided into primary (those whose employment have a direct influence on the achievement of victory) and secondary (those that do not directly result in the achievement of victory but nevertheless are useful to the player). In addition, some mechanics are defined as compound; - i.e. several tertiary mechanics which comprise a core mechanic.

Character model: A strong character model is important, both for increased player immersion, and because it allows more realistic behavior modelling, particularly of NPCs, and assist the later implementation of AI. For the *JominiEngine*, the following attributes (simplified for the initial version) were assigned: Stature, Combat, Management, Virility, Maximum health. Additionally, in order to provide increased flexibility, a number of 'traits' were defined, each of which positively or negatively

influence particular game mechanics and character attributes. In the interests of simplicity, the pool of available traits was limited, and only 2-3 were allocated per character. It is hoped that future enhancements to the *JominiEngine*'s character model can be simplified via the introduction of new traits.

Rules: Game rules were also created so that they could be readily mapped to predicates and conditional checks in the game program. For this reason, it is important that even inherently obvious rules should be recorded (e.g. in the *JominiEngine,* the value of the fief tax rate must lie between 0-100). In an attempt to maintain historical accuracy, some rules were impacted by the need to ensure that the game ultimately remained fun to play; e.g. maintaining an army in the field is a straightforward matter of allocating the appropriate funds, rather than arranging for the purchase and delivery of different types of supplies.

Historical accuracy: The additional dimension of historical accuracy was reflected in the design of the *JominiEngine* game model in that: 1) goals (victory conditions) allow for the achievement of historical conditions; 2) game rules reflect the social practices of the times; e.g. the use of marriage as a tool to acquire stature; 3) game mechanics are restricted to those that would have been available in the target time period, albeit simplified. Where possible, the formulas underpinning the mechanics are based on historical research; e.g. the formula used to derive battlefield casualties [28]; 4) data for *Overlord* was taken from reliable sources [25] and includes personalities, titles and ranks, fief data, and troop types.

4 Performance Aspects

In order to assess the current implementation of the *JominiEngine*, in terms of basic performance and scalability, we assessed the performance of the underlying Riak database and ran a series of measurements, for insert, fetch and delete operations, into a simple but sizable database, performing a sequence of operations from within a varying number of threads up to 150 in total. The goal was to simulate high load on the database, and to compare the access times for each set of operations, with the access times on a standard MySQL server, as one representative of a main-stream SQL-based engine. The results in Fig 2 (below) show that insert latencies are considerable lower with Riak (4.8ms vs 27.8ms for a single thread, up to 140ms vs 300ms on 150 threads) compared to MySQL. This reflects differences in design, where an SQL-based engine is conservative, always providing exclusive write access to the database, which is expensive, Riak has looser guarantees, avoiding the necessity of locks at the expense of potential roll-backs. We observe a similar picture for the performance of delete operations: 4.4ms vs 26.9ms with a single thread, up to 310ms vs 220ms on 150 threads. In this case the cost for roll-back operations may kick in for the Riak implementation, making it more expensive on this scale. Finally, for the latencies of fetch operations we observe that MySQL is consistently faster than Riak: 0.3ms vs 1.5ms with a single thread, up to 30ms vs 68ms with 150 threads. This reflects that no locks are needed for read access, and the SQL engine can profit from a highly optimised handling of read requests. We note, however, that in terms of scalability, Riak catches up with MySQL for higher thread numbers (Fig 2 below).

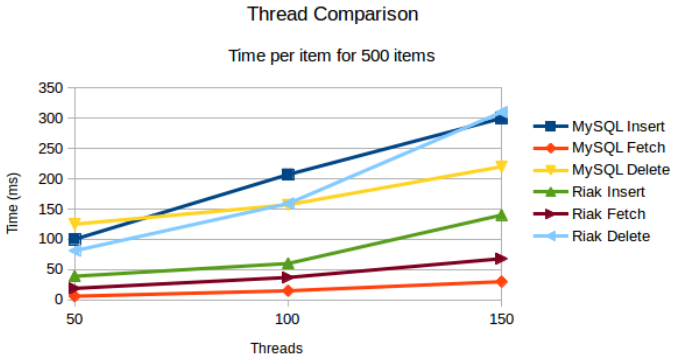

Fig 2: Database performance comparison with increasing thread numbers (50 to 150)

These preliminary performance results show a mostly positive picture in terms of the benefits of a NoSQL database. While we have gains for write access and modifications, the cross-over in performance for deletes for high thread numbers needs further investigation, and we do observe a performance penalty for read operations. In the bigger picture, however, Riak brings benefits in terms of distribution, which is wired into the database engine, and it has been designed with data redundancy and for massive concurrency, building on the Erlang system, which is known for its efficient handling of concurrency. We believe that these preliminary results strengthen our case for using Riak as the database in the *JominiEngine*.

5 Conclusion

We have presented the design and initial implementation of a game engine for historical massive-multiplayer on-line role-playing games. The *JominiEngine* has been designed for the educational application domain and to support this usage it allows for the accurate modeling of historical game data as an incentive metric to encourage team play and reflection on historical context. "*Overlord"* presents such data as one instance of the game engine. We envision to use such instances as educational tools to provide insights into dependencies of societal and economic forces in a specific time period and geographic area. Thus, our immediate application domain will be (interactive) history in education. This technical paper is concerned with the performance aspects of the game engine and its requirements for *scalability, modularity, fault tolerance* through data redundancy and the authoring of both primary and secondary game mechanisms. The *JominiEngine* is a generic game engine, which can be used, in different educational contexts. This technical article discusses key decisions made during the development of the engine in terms of scalability (the use of Riak as a NoSQL database to reduce lookup latency), and modularity (C# as a flexible implementation language, an API for key functionality in the game engine, usage of established data formats for data exchange). Reflecting on *Overlord*, the initial instance of the game, we observe that accurate content authoring is challenging and requires a high investment of research time.

The *JominiEngine* has been developed in order to design a scalable and extensible MMOPRG solution for serious game interventions. The scalability aspect discussed in this article is essential for large-scale deployment and a critical condition to be met in order to distribute educational MMORPGs simultaneously in a large number of institutions. Additionally, the *JominiEngine* has been designed as a generic and modular solution so as to extend its applicability to domains other than history and battle re-enactments. The *Overlord* game has been developed using the *JominiEngine* in order to illustrate the concrete implementation of these concepts discussed in this article and to provide a basis for future deployments A Unity-based client has recently been added, without requiring major restructuring of the core engine. The complete source code is available at https://github.com/libdab/hist_mmorpg/tree/nonMVC, (under the OSI approved MIT license) and a detailed discussion of the JominiEngine is given in Bond [32].

References

1. Adkins, S.: Keynotes Serious Play Conference 2013: The 2012-2017 Worldwide Game-based Learning and Simulation-based Markets (2013), http://bit.ly/1s42fDL (accessed: 07 July 2015).
2. Gee, J.P.: What Video Games Have to Teach Us About Learning and Literacy. Palgrave/Macmillan, New York (2003).
3. Csikszentmihalyi, M.: Flow: The Psychology of Optimal Experience. Harper & Row, New York (1990).
4. Gee, J.P.: Games, Learning, and 21st Century Survival Skills. J. for Virtual Worlds Research, 2(1) (2009), http://bit.ly/1NjPdZF (accessed: 18 Oct. 2015).
5. Gee, J.P.: Deep Learning Properties of Good Digital Games. In: Vorderer, P., Ritterfeld, U., Cody, M.J. (eds.) Serious Games: Mechanisms and Effects, pp. 63-80. Routledge, New York & London (2009).
6. Lee, M.J.W., Eustace, K., Fellows, G., Bytheway, A., Irving, L.: Rochester Castle MMORPG: Instructional gaming and collaborative learning at a Western Australian school. Australasian J. of Educational Technology, 21(4), 446-469 (2005).
7. Gikas, J., Van Eck, R.: Integrating video games in the classroom: Where to begin? Paper presented at the National Learning Infrastructure Initiative 2004 Annual Meeting, San Diego, CA, January 25-27 (2004), http://www.educause.edu/ir/library/pdf/NLI0431a.pdf (accessed: 01 June 2015).
8. Gagne, R., Briggs, L., Wager, W.: Principles of Instructional Design (4th ed.). HBJ College Publishers, Fort Worth (1992).
9. Bloom, B.S. (ed.): Taxonomy of educational objectives. Handbook I: Cognitive domain. David McKay, New York (1956).
10. De Freitas, S., Griffiths, M.: Online gaming as an educational tool in learning and training. British J. of Educational Technology, 38(3), 535-537 (2007).
11. Yu, T.W.: Learning in the virtual world: The pedagogical potentials of massively multiplayer online role playing games. International Education Studies, 2(1), 32-38 (2009).
12. Kolb, D.A.: Experiential learning, Prentice Hall, Englewood Cliffs (1984).
13. Childress, M.D., Braswell, R.: Using massively multiplayer online role-playing games for online learning. Distance Education, 27 (2) , 187-196 (2006).
14. Kongmee, I., Strachan, R., Montgomery, C., Pickard, A.: Using massively multiplayer online role playing games (MMORPGs) to support second language

learning: Action research in the real and virtual world. In: 2nd Annual IVERG Conference: Immersive technologies for learning (2011) http://bit.ly/1kiPHWc (accessed: 18 Oct. 2015).
15. Eustace, K., Fellows, G., Bytheway, A., Lee, M., Irving, L.: The application of massively multiplayer online role playing games to collaborative learning and teaching practice in schools. In: Atkinson, R., McBeath, C., Jonas-Dwyer, D., Phillips, R. (eds.) Beyond the comfort zone: Proceedings of the 21st ASCILITE Conference (2004) http://bit.ly/1hMNzEm (accessed: 18 Oct. 2015).
16. Kirschenbaum, M.: War; what is it good for? Learning from wargaming (2011), http://www.playthepast.org/?p=1819 (accessed: 18 Oct. 2015).
17. Von Reiswitz, G.H.R.: Instructions for the representation of tactical maneuvers under the guise of a wargame, s.l.: s.n. (1824).
18. Crusader Kings 2, http://www.crusaderkings.com/ (accessed: 18 Oct. 2015).
19. Clash of Clans, http://supercell.com/en/games/clashofclans/ (accessed: 18 Oct. 2015).
20. Mirani, L.: Why free games are increasingly the most profitable apps (2014), http://qz.com/172349 (accessed: 18 Oct. 2015).
21. Anderson, E. F., McLoughlin, L., Liarokapis, F., Peters, C., Petridis, P., de Freitas, S.: Developing serious games for cultural heritage: a state-of-the-art review. Virtual Reality, 14(4), 255-275 (2010).
22. Dunnigan, J.F.: Wargames handbook: How to play and design commercial and professional wargames (3rd ed.). Writers Club Press, Lincoln (2000).
23. Von Clausewitz, C.: On war. Wordsworth Editions Limited, Ware (1997).
24. Sumption, J.: The Hundred Years War. 3 vols. Faber and Faber, London (1990-2009).
25. Nofi, A.A., Dunnigan, J.F.: Medieval life & The Hundred Years War (1997), http://bit.ly/1M4hoNH (accessed: 18 Oct. 2015).
26. Oman, C.W.C.: The art of war in the Middle Ages: A.D. 378-1515. B.H. Blackwell, Oxford (1885).
27. Ross, S.J.: Medieval demographics made easy (2013), http://www222.pair.com/sjohn/blueroom/demog.htm (accessed: 5 July 2015).
28. Adams, E.: The Designer's Notebook: Kicking Butt by the Numbers: Lanchester's Laws (2004), http://ubm.io/1eD8N6v (accessed: 5 July 2015).
29. Emilsson, K.: Infinite space: An argument for for single-sharded architectures in MMOs (2014), http://ubm.io/1LWxaFh (accessed: 18 Oct. 2015).
30. Muhammad, Y.: Evaluation and Implementation of Distributed NoSQL Databases for MMO Gaming Environments. MSc dissertation, Uppsala Univ., unpublished (2011).
31. Li, Y., Manoharan, S.: A performance comparison of SQL and NoSQL databases. In: 2013 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM), pp.15-19. IEEE (2013).
32. Bond, D.A.: Design and implementation of a massively multi-player online historical role-playing game, MSc dissertation, Heriot-Watt University, unpublished (2015), http://bit.ly/1LXhj9k (accessed: 18 Oct. 2015).
33. Riak, http://basho.com/products/#riak (accessed: 18 Oct. 2015).
34. Lua, http://www.lua.org/ (accessed: 18 Oct. 2015).
35. Sicart, M.: Defining game mechanics. Game Studies: International J. of Computer Game Research 8(2) (2008), http://bit.ly/1Hv4UJH (accessed: 18 Oct. 2015).
36. Crawford, C.: The art of computer game design (1984), http://bit.ly/1Tjgv4X (accessed: 18 Oct. 2015).
37. Crawford, C.: The art of interactive design: A euphonious and illuminating guide to building successful software. No Starch Press, San Francisco (2002).