Usability evaluation methodology for scalable multiuser games

Bogdan-Ioan Oros
Technical University Of Cluj-Napoca
oros.bogdan.ioan@gmail.com

Victor Ioan Bâcu
Technical University Of Cluj-Napoca
victor.bacu@cs.utcluj.ro

Dorian Gorgan
Technical University Of Cluj-Napoca
dorian.gorgan@cs.utcluj.ro

ABSTRACT
The aim of this paper is to propose means for objective evaluation metrics in case of multiuser online games and to offer a viable solution with a scalable architecture for the coordination services for this specific type of games. This paper presents a covering analysis on the high usability of such a solution with justification based on objective usability metrics, reusable for different game designs and scenarios. The article justifies the need of switching to a set of proposed metrics as means to usability evaluation instead of relying strictly on user feedback based on empirical analysis of the game.

Author Keywords
Usability; HCI; human computer interaction; scalable; computer games; multiuser; multiclient;

ACM Classification Keywords
Human computer interaction (HCI)

General Terms
Human Factors; Design; Measurement.

DOI: 10.37789/rochi.2021.1.1.13

1. INTRODUCTION

It is widely known that computer games have seen a major increase in the number of users and studios preoccupied with the development of such applications. Among computer games, the ones that support online functionality with multiple users and coordination of a central system are the most promising in the aspects of continuous communication, interaction and socialization among the users due to its nature.

The development of such a solution encapsulates careful design decisions from multiple perspectives such as scalability concerns both on the client application as well as the server, performance and availability, lack of technological support and most importantly the ability to satisfy given usability constraints for a high adoption among the users.

The research performed in the writing of this paper has focused on exploring a possible solution that responds to the above mentioned development difficulties. The evaluation will feature several relevant metrics in order to justify its compliance to meet high usability thresholds.

This paper will be structured as following: The first section presents an introduction to the domain of online games together with current difficulties in the domain, the second section will present related work with focus on existing architectures for such games, techniques and methods used for implementation and evaluation, the third section will put emphasis on an overview of the evaluated game, the forth section will feature the proposed architecture for this specific scenario, the fifth section will list the experimental results and validation of the solution and finally a section with the conclusions and future improvements.

2. RELATED WORK

This section will contain existing work in the domain of development of large scale games with a central highly available architecture that interacts with multiple clients. It will cover findings related to the evaluation means for computer games based on user experience heuristics.

The paper [1] focuses on the application of known methods for usability evaluation to computer game design. It presents the concerns associated with development of computer games (the increasing game quality over time, the fact that costs have to be reduced and furthermore the fact that the game has to be adaptable to more genres) as a justification for incorporating usability evaluation in the development lifecycle.

The above mentioned paper targets key usability heuristics such as:

- Tasks should be taught appropriately, referring to the idea that actions that can be taken in a given game state should be intuitive and the user should be assisted with appropriate metaphors that he can recognize
- Tasks should be logical and consistent, highlighting the idea that the same action should be performed with the same user input independent of the rest of the environment of the game
- Aspects of the game world should be distinguishable, meaning that the user should be aware at all times what objects from the scene the user can interact with and which are static objects. The important idea of this heuristic is that dynamic objects in the scene should be observable over static ones.

The work highlighted in the papers [2], [3] and [4] reflects the methodology required for implementing a game following the main steps (topic selection, game specifications, prototyping, scenario and task description, implementation and evaluation). It can be observed that the paper [4] highlights the full implementation of a game and evaluation for it according to certain hardware specifications. It follows a functionality test, as well as a heuristic test. The visibility of the system status, the user control and freedom, design consistency and standards, error prevention and guidance for error resolution, as well as flexibility and efficiency of use.

The workshop [5] presents the problems in the evaluation of games, among which the lack of a standard for evaluating games, what factors of game experience should be measured and the methods for measuring them, common concepts and methods used in the industry, and was later detailed in [7]. As such, Bernhaupt identifies multiple key heuristics in for tabletop type games:

- Cognitive workload: the idea that the cognitive workload not connected to the game play should be minimal
- Challenge: the game should be designed in such a way that it satisfies the preconditions and the target group
- Reach: the game should be designed in such a way that the players can easily adapt and satisfy the requirements of the gameplay
- Examinability: The players should not be limited in the exploration of the area of the game play
- Adaptability: the system should adapt to various players and their setup
- Interaction: the interaction method should be intuitive, satisfy the expectations of the players and follow a consistent game logic
- Level of automation: All the actions relevant to the game should not be automated and the user should be able to perform them by themselves
- Collaboration and communication: interpersonal communication should be supported if possible throughout the game
- Feedback: The system should provide relevant feedback in case of possible errors and provide further guidance to performing a valid operation in a given context
- Comfort of the physical setup: the way of setting up the environment to be comfortable affect the user experience of the game directly

The work highlighted in [6] presents an example of an online game architecture similar to industry standard architecture for low latency real-time online games with a massive number of players. The advantage of this type of architecture is mainly the performance. In figure 1 it is presented a possible concrete architecture based on the design decisions according to [6]. In this type of architecture there is a server that communicates with the database. Game servers communicate with this server in order to perform periodical backup of the player state. The state of the player is kept in the memory of the process associated with the game server throughout the interaction period of the client with the game server, authentication server and player chat server. The authentication server is responsible for validating the identity of the player, it is decoupled from the rest of the game servers for security purposes, a player who is not authenticated cannot determine the network address of the game server. Lastly, there is a communication server for the players, the communication server does not require database access and it does not log messages across the network. It serves as a mean for players to communicate in the virtual world.

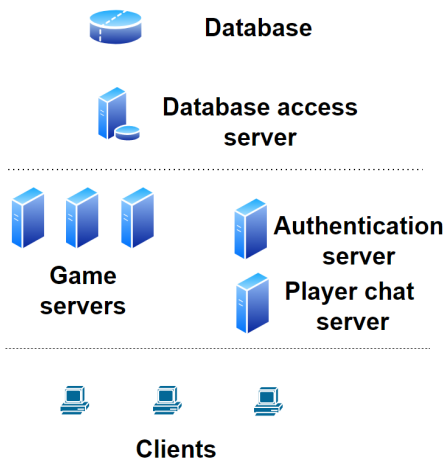

Figure 1: Low latency online game architecture

Networking latency issues that are concerns of [8], that is processing delay, transmission delay, queuing delay and propagation delay are addressed by this architecture in the sense that queuing time is minimal, there are multiple game servers supporting a specific region of the virtual world, the rest are implementation independent latency factors.

The work in paper [6] presents the fact that the game servers are responsible for coordination and keeping the state. In this sense the paper proposes region based locking and object locking. Locking has an important significance because it enables transactional operations.

As compared to [1] and [5] we plan to use measureable results over empirical analysis in order to prove the fact that the game satisfies minimal usability requirements. As such, the chosen metrics focus on multiple aspects of the game in order to obtain the final indicator for usability.

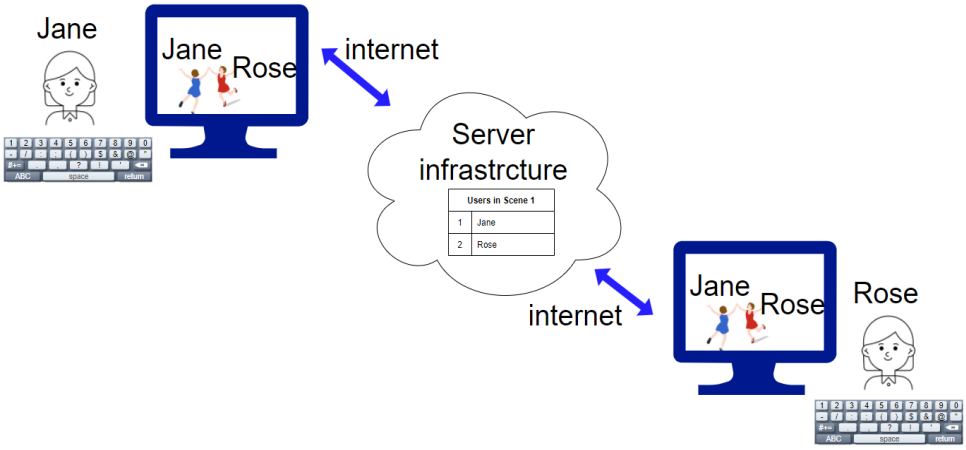

Figure 2: Conceptual representation of the multiuser game

3. GAME PRESENTATION

The setup of the proposed game (Figure 2) focuses on an adversarial multiplayer game with the rendering performed on the client machines. Each player will be represented by an avatar with a unique name in the virtual world. By issuing different commands the user can control the different actions for the avatar he or she controls. The scene contains different obstacles and boundaries in order to restrict movement and encourage strategic planning.

It encourages player communication, socialization and interaction by the means of allowing chat as well as cooperation in order to eliminate a common enemy, overall by means of human computer interaction we enable means of establishing a virtual human to human interaction.

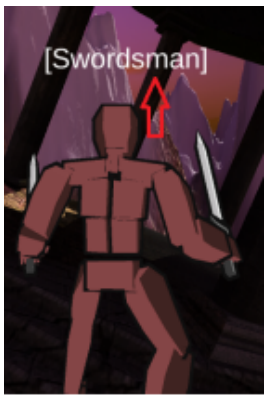

Figure 3: Representation of a user as an avatar in the virtual world

Figure 3 shows the representation of a user in the virtual world in the form of an avatar with a unique name. Other users can view the changes in animation of the respective avatar when the user who is controlling it triggers an action.

The user interacts with the game world by issuing commands via the mouse and keyboard, the mouse being used for controlling the position and zoom factor of the camera and the keyboard being used to trigger movement commands, entering chat messages and initialing attacks.

The communication between the client and the server takes place using binary packets over text encoded messages as to reduce the bandwidth required to the minimum. The communication language is defined in such a way as to satisfy the game requirements, but prevent irrelevant information from being sent. Each game client application has to be gathered by each user individually. This application contains the executable as well as the graphics contents required: the scenes of objects, the models for the avatars, the corresponding animations, fonts and particle effects, sounds, etc. The graphics content is not streamed during the gameplay session to other users nearby in order to avoid traffic, only a minimal sized packet identified by its type and additional information is sent.

Due to the fact that this type of games requires massive amounts of users in order for them to be enjoyable, usability is a primary concern, as the higher the usability of the game the higher the user base of the game will become.

4. MULTIUSER DEVELOPMENT SOLUTION

The main consideration when designing multiuser online games is the fact that the central architecture has to offer performance while scaling according to the number of connected users. The architecture focuses on being ACID (atomicity, consistency, isolation, durability) compliant in order to provide a more general scope for any kind of game that can implement it.

The client performs the graphical processing for the game from rendering objects to animation control of the characters in the scene. The avatars that are displayed require information coming from the server. The server side is responsible for broadcasting information for players within your radius and their movement and animation start or end actions. For this we require that the client can be notified asynchronously from the server.

The continuous bidirectional communication is assured by means of a keep-alive TCP connection. As known the TCP protocol is a networking protocol that assures ordering of packets, guaranteed delivery of packets, retransmission of data and error checking capabilities. Due to the fact that the connection can fail, the client maintains multiple connections similar to this, but only uses one of these at a time. In order to retrieve the list of server IP addresses to use for load-balancing the client can query a DNS server and get a list of associated game server servers. A domain name server (DNS) is a decentralized solution that offers domain name to IP address mapping services.

The game servers are responsible not only for the continuous communication with the players, but they are responsible for triggering periodical tasks in order to generate events, perform certain business logic and communicate the results to the players. As an example we could consider the use of AI controlled avatars. At a certain interval the agent would need to perform an action. In order to trigger this task we use task schedulers and once the task is performed connected users will be notified, as well as the rest of the game servers. Game servers communicate between each other in order to be aware of which server is handing what players.

Another layer of load balancing is then introduced for the handling of the business logic. The framework we propose is independent of the business logic of the game, as a result we should be able to allow any number of complex tasks to run smoothly on the architecture. This stateless automatically scalable layer is a layer consisting of microservices that scale according to the workload automatically. The fact that these units of work do not hold any state of the game object is relevant because they can be turned off or on at any moment. Elasticity is an important matter due to the high volumes of users that can be expected in such a game. As a result it is not cost efficient to run a large number of units at all times.

Finally, in the design we proposed for the server part the database consists of a relational database that is ACID compliant. In the design we favor consistency over availability. The relational database implements sharding capabilities in order to scale. It is accessed using a common interface by the stateless processing units.

5. EXPERIMENTAL RESULTS AND VALIDATION

In order to accurately validate the model we are required to integrate existing metrics with the multiuser game model and applying an aggregation of these metrics in order to obtain the final score for the usability of the multiuser game. The metrics featured will be the performance measured by means of average bandwidth, average throughput and average latency of the user input according to varying amounts of stateless processing units and varying amounts of users, the synchronization between the positions of the avatars at two clients and between a client and the server, the frame rate with a varying amount of users in the viewing volume for a single client instance, as well as the interactiveness and responsiveness measured by means of user feedback.

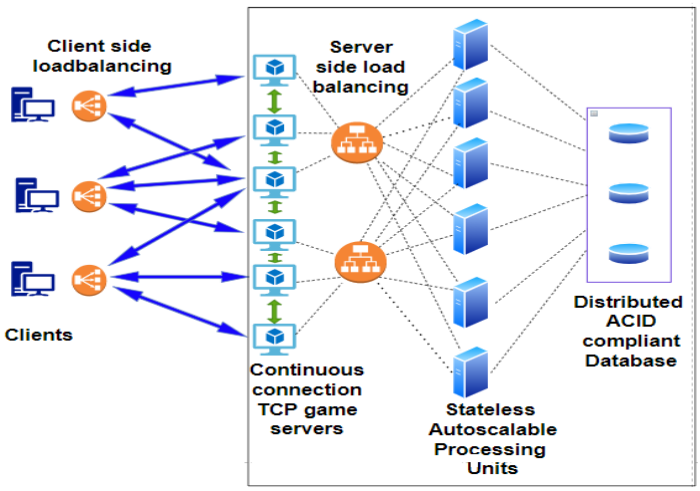

Figure 4: Architecture overview for multiuser online game with acid compliance for full transactional operation

5.1 Performance

Performance is one of the most important factors in online applications as it is mandatory for the game to be responsive. In order to assure high performance even at an increased amount of users in the same scene we have resorted to server replication.

In order to measure performance we have several key indicators for our communication language. We validate the performance by means of measuring latency, bandwidth, throughput in several scenarios.

The bandwidth shown in the following table is measured as an average between both the client to server and server to client communication during active game play.

In order to obtain relevant results we have used dummy clients without rendering capabilities such as not to include in our measurements the time required for graphical processing.

The latency measured refers to the latency for enqueuing a message at client side. Due to the fact latency is dependent on the location of the server and client it is irrelevant to measure the latency of a packet transmission.

The system throughput is measured as a sum of operation during the time unit of one second at all endpoints responsible for processing (stateless processing units). In terms of evaluation, the average throughput should scale linearly with the amount of users and with the amount of processing units.

The cumulative difference obtained by summing the deviation (measured in percentages) from the linearly scaled values represents the performance loss associated with the system. In our case, for each of the following: average bandwidth consumption, average throughput and average latency, we would sum up the differences between the estimated values for these metrics (the estimations are obtained by linearly scaling based on the first value) and the actual values obtained in the experiments. A cumulative difference of lower than 5% will be assigned a score of 10 for evaluation, while a cumulative difference of over 50% will be assigned a 1.

5.2 Synchronization

It is relevant that clients perceive the same state of the virtual world at the same point in time, the same would apply between the state of the game at server side and at a random client. As such we have defined an aggregated difference between the positions on the three axis of the Cartesian system.

$$\text{SyncFactor} = \sum_{k=0}^{n}(C1_k.\text{X-}C2_k.\text{X}) + (C1_k.\text{Y-}C2_k.\text{Y}) + (C1_k.\text{Z-}C2_k.\text{Z})$$

where $C1_k$ represents the position of avatar k on the client application of the first application instance considered for measurements, and $C2_k$ represents the position of avatar k on the client application of the second instance, similarly the formula can be applied for a client and the avatar instances on the server.

In order to perform these experiments we have created a remote query server that will query any of the two clients client, respective one client and the total synchronization factor for various numbers of avatars.

The data gathered indicates the synchronization factors between 2 random clients from the instance pool, as well as the comparison between the game state available at the server side and a random client. It is noticeable in figure 3 that the synchronization factors scale linearly with the amount of instances assuming similar latency to the server and a constant number of stateless processing units of three.

Table 2 outlines the fact that server validation for player state is more appropriate compared to clients cross validation due to a smaller error of position between avatar instances in the two game states.

Stateless processing units	Amount of users	Average bandwidth consumption	Average Throghput	Average Latency
1	10	412 bytes/sec	124op/sec	<1ms
1	50	394 bytes/sec	321op/sec	<1ms
1	100	390 bytes/sec	333op/sec	<1ms
3	10	427 bytes/sec	115op/sec	<1ms
3	50	399 bytes/sec	514op/sec	<1ms
3	100	388 bytes/sec	889op/sec	<1ms

Table 1. Performance indicators based on various number of users and processing units

In figure 3, it can be seen that the synchronization factor in the case of the measurement between the game state on the server and the game state on a client is not exactly twice as large as the client to client synchronization factor because the synchronization factor is dependent on multiple factors such as the load on the server at that point, the network latency which is not consistent, as well as the individual latency until the message is sent to the server from the client generated by the amount of resources allocated to the client application.

In figure 5, it can be seen that the synchronization factor in the case of the measurement between the game state on the server and the game state on a client is not exactly twice as large as the client to client synchronization factor because the synchronization factor is dependent on multiple factors such as the load on the server at that point, the network latency which is not consistent, as well as the individual latency until

the message is sent to the server from the client generated by the amount of resources allocated to the client application.

Amount of avatar instances	Client 1 average ping latency	Client 2 average ping latency	Server average ping latency	Syncronization factor (between clients)	Syncronization factor (between client and server)
10	2.3 ms	2.431 ms	2.11 ms	0.54	0.23
25	2.3 ms	2.431 ms	2.11 ms	0.74	0.33
50	2.3 ms	2.431 ms	2.11 ms	0.94	0.45
75	2.3 ms	2.431 ms	2.11 ms	1.15	0.61
100	2.3 ms	2.431 ms	2.11 ms	1.43	0.73
150	2.3 ms	2.431 ms	2.11 ms	1.67	0.9

Table 2. Syncronization factors between two clients and between a client and the server for a varying amount of avatar instances

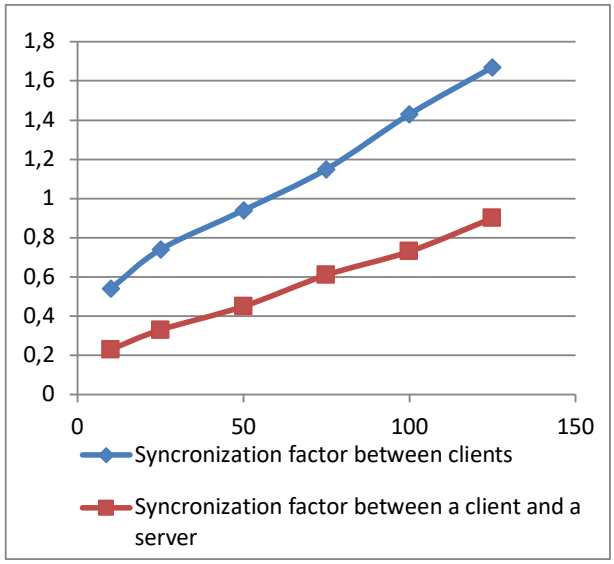

Figure 5. Syncronization factors of game world state at different targets

The theoretical upper bound for the synchronization factor is limited by the size of the scene, in our case a scene of 1024 by 1024 virtual meters (default measurement unit for space as considered in Unity [10]) and the theoretical lower bound is 0 in case of perfect synchronization between the two game states.

5.3 Frame rate

The experiments were performed using a client that has constant graphical processing and CPU capabilities. We have experimented by creating multiple scenarios with different number of dummy game clients in order to measure the frame rate at this client which is performing the rendering. The tests were perfomed with the following configuration: Nvidia GTX 1050 Ti graphics card, Ryzen 3600 CPU, 16 gigabytes of random access memory.

In table 3, the frame rate for the client scales inverse proportionally to the amount of avatars, as avatars are represented using three dimensional objects formed out of polygons, the number of polygons will increase as the number of instances increasing, resulting in higher computational needs to compute a frame, resulting in a lower frame rate. The matchmaking procedure can split the users into different instances of the same scene, with a maximum amount of avatar instances in order to satisfy a target frame rate, in our implementation we targeted 150 players within the same scene instance and had an output of approximately 30 frames per second, which is the industry standard for acceptable frame rate.

The values for the frame rate will be considered within the range 0 to 60. During evaluation, 60 frames per second will be assigned as score of 10 and a frame rate between 0 and 5 will be assigned a score of 1.

Amount of avatar instances	Rendering frame rate
10	60
25	60
50	60
75	58
100	49
150	29

Table 3. Frame rate based on amount of avatar instances present within the viewing range for a single rendering client

5.4 Interactiveness and responsiveness

The interactivity of the game is increased as players can communicate with each other, observe each other in the

visual world and attack each other or collaborate towards eliminating a common perceived enemy. These actions translate to a set of commands that form the communication language. The communication language is complete, consistent and flexible. It holds commands that allow for:

- Broadcasting rotation and scaling changes for an object in world space coordinates
- Broadcasting movements as a set of two points with the current target and destination, used on the client side for linear interpolation in order to obtain a smooth movement
- Broadcasting movements relative to other identifiable objects in the scene
- Broadcasting movements relative to other avatars in the scene
- Broadcasting chat messages
- Broadcasting attacks messages
- Broadcasting system signals (such as the start of a round, the end of a round, a player being eliminated)

The completeness of the language refers to offering full support for translations, rotations and scaling in absolute world coordinates as well as relative to other objects or avatars in the scene. The language contains collaborative support as well as adversarial actions in the form of chat and attack messages respectively. In brief, the language allows for full transformation control of the avatars as well as allowing communication support and support for collaborative or adversarial actions in order to change the state of the avatar and the game world.

Responsiveness is highly depended on the frequency of movement corrections on the server side. Since the interpolation takes place on the client side, we can assume different duration for the movement to complete on the client side based on the rendering capabilities and on the server side. Considering the rendering delay and the network delay, we can assume that once the movement has completed and the user has reached the destination point a cross validation is performed on the server side. In case the avatar lies outside the area that is defined by a certain margin of error, the user will be forcefully teleported (figure 6) to the destination point calculated on the server side. Given high network latency, the frequency of the teleports or movement corrections will be directly proportional with the network characteristics and it will in turn be increased, resulting in jittering and reduced responsiveness.

The responsiveness will be evaluated based on questionnaires offered to a different group of users that can assign a grade from 1 to 10 for the responsiveness and different interactivity elements.

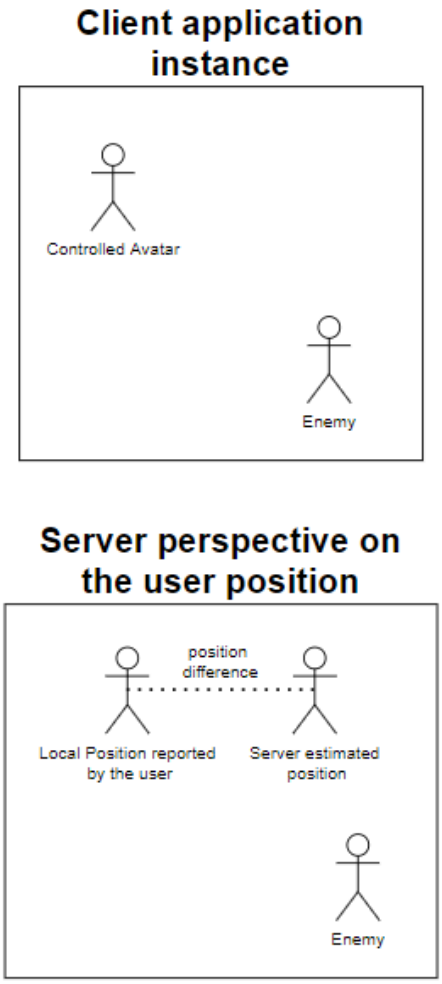

Figure 6. Server side position validation and correction

5.5 Usability

We define the usability as the weighted average (with the weights assigned accordingly in relation to the focus of the game design, if one metric is more important it should be assigned a higher weight) of the above presented metrics in order to obtain a final evaluation for this type of game. It is important to consider that the network latency plays an important factor in the responsiveness of the application. Even though the application was designed in such way as not to block the execution of the game awaiting a server reply, an action that awaits a response tends to confuse the user. Increased performance by means of scalability is a secondary factor as limitations on the number of instances the server and a client instance can hold significantly affects the playability of the game. Furthermore, in abstraction the networking, the client the 2D interface available for the user, as well as the metaphors available in the 3D space to operate the avatar in the scene were designed by keeping in mind the usability engineering principles under [9].

$$Usability = \frac{PerformaceScore * w1 + SyncFactor * w2}{w1 + w2 + w3 + w4} + \frac{Responsiveness * w3 + FramerateScore * w4}{w1 + w2 + w3 + w4}$$

6. CONCLUSION AND FUTURE IMPROVEMENTS

The usability evaluation methodology proposed n this paper puts emphasis on the importance of having quantifiable metrics for evaluating and validating a software design for a larger system where several users globally distributed interact by means of coordination from a central server and it presents a general architecture that enables any number of implementations regardless of the theme of the game.

The contribution that this paper brings to the field consists of proposing an original solution for handling large volumes of users and processing tasks over a distributed system, as well as validating the solution by means of an original set of usability metrics.

Future work involved would imply expanding the set of metrics and providing a more general framework to evaluate any possible reference game in order to compare and contrast different implementation with the intent of improving usability for human users of said computer games. Development of automated agents that operate the game client application could be developed and a server side solution to identify this software agents would prove useful as the model evolves and the game would gain popularity.

REFERENCES

1. Brown, Michael. (2008). Evaluating Computer Game Usability: Developing Heuristics Based on User Experience.
2. Al-Doori Rami, Blaga B.C.Z., Gorgan D., Exploring Solutions for the Development Methodology of the Video Game DABABAT, Proceedings of the RoCHI 2018 Conference, ISSN 2501-9422, pp. 119-126, (2018).
3. Morar A.G., Gorgan D., Experiments on Computer Game Development Methodology, Proceedings of the RoCHI 2018 Conference, ISSN 2501-9422, pp. 127-134, (2018).
4. Blaga B.C.Z, Gorgan D., Game Development and Evaluation of the EvoGlimpse Video Game, Romanian Journal of Human-Computer Interaction, Vol.11(1), ISSN 1843-4460, pp.40-62, (2018).
5. Bernhaupt, Regina & Eckschlager, Manfred & Tscheligi, Manfred. (2007). Methods for evaluating games: How to measure usability and user experience in games?. 309-310. 10.1145/1255047.1255142.
6. Marios Assiotis & Velin Tzanov.(2006). A Distributed Architecture for MMORPG
7. Bernhaupt, Regina .(2010).Evaluating User Experience in Games
8. Josh Glazer, Sanjay Madhav. (2015). Multiplayer Game Programming: Architecting Networked Games.
9. J. Nielsen, Usability Engineering: Morgan Kaufmann Publishers Inc., 1993.
10. Unity game engine.(2021). https://unity.com/