Exploring Solutions for the Development Methodology of the Video Game DABABAT

Al-Doori Rami
Technical University of Cluj-Napoca
Cluj-Napoca, Romania
Rami@uob.edu.iq

Bianca-Cerasela-Zelia Blaga
Technical University of Cluj-Napoca
Cluj-Napoca, Romania
zelia.blaga@cs.utcluj.ro

Dorian Gorgan
Technical University of Cluj-Napoca
Cluj-Napoca, Romania
dorian.gorgan@cs.utcluj.ro

ABSTRACT

This paper proposes an implementation of a multiplayer war game using Unity Game Engine and explores all the basic steps of development, starting from planning until getting a product, and evaluating by usability heuristics. The development steps contain a method to build a functional avatar of tank module with a synchronized movement of the turret and the gun.

Author Keywords

Blender 3D; Heuristics Evaluation; Tanks; Unity.

INTRODUCTION

A multiplayer game over a network is basically a video game where more than one human player shares the same game at the same time with a unique view. Each player is able to see the same game but from different angles. All players share some resources and each one has some items, like weapons, characters etc. Some of these resources can be altered by the actions of other players. For example, if we take into consideration the health factor, for some players it can be changed by a shooting event performed by another player from the opposite team.

The main problem in an online multiplayer game is the optimization of network messages between the players, in other word data synchronization. Another problem is related to the network communication layer, for example, lack of service.

Dababat is an Arabic word that means tanks. Thus, the game is a tank fight between two teams over the network. It was inspired by another online game called World of Tanks [1]. A lot of people do not have knowledge about armored war vehicles and how they engage each other in battle. Therefore, the game's goal is to provide players with a battle environment governed by the law of physics and engaging rules which almost mimic the real world.

This paper is organized as follows: in section Related Work we present the description of other similar projects and evaluation methods. The next section contains the development of the game in detail and everything to know about it. We also have a section in which we present the heuristics evaluation method for the game, with the last two sections being the Results Analysis and Conclusions.

RELATED WORK

Games, in general, are a big subject to be explored due to the variety of the types and genres that exist. Digital games can be categorized according to the machine that runs the game, to the number of the players involved or to the end user's age. Shooter games are one of the most popular statistically. Most wargames are shooter type, and the target ages range from 10 – 50 for both genders.

Mobile devices can be used as terminals for games, as most of the current mobiles have the minimal requirements for it. 2D video games can work on most of today's phones. The development of a Tanks War Mobile Game based on Android System can be seen here [4].

A Massively Multiplayer Online Game (MMOG) is developed to evaluate and compare the peer-to-peer-based MMOG systems with scalability in mind. The game requires low network latency and creates frequent game state updates [5].

Another performance evaluation tested for peer-to-peer gaming overlays can be found here [6]. It consists of a 3D first person shooter game that is designed to run in a simulated network environment as well as on a real network. Simulation with autonomous players (bots) guarantees scalability, a controlled workload, and reproducible results.

As for other evaluation methods, [7] shows the creation of two heuristics, a specific and slandered one. The authors conduct an evaluation on two separate groups, and after that, matching between the two heuristics lists is done to have a relation to base the results upon. In [8], the authors used heuristics evaluation to detect playability problems in a short time for mobile games and proved the method to be efficient and effective.

The difficulties encountered by evaluators when they conduct a formative evaluation of both usability and fun in computer games for children is presented in [9]. It is also shown the problems when determining with which set of heuristics (usability vs. fun heuristics) identified problems should be explained. In the next sections, we will talk about the development requirements of a video game that we worked on and how we tackled the topics of design, implementation, and evaluation.

GAME DEVELOPMENT REQUIREMENTS

In order to create a game, we need a game engine that has enough capabilities for a war game over the network. Therefore, we chose Unity platform which is free and easy to use.

Unity framework offers a store for assets where a lot of 3D objects can be found. The code is written in C# and we can edit it using Visual Studio which is one of the most powerful IDEs that currently exist. Unity also supports 24 platforms including Oculus Rift, PlayStation 4 and Linux.

The multiplayer game depends on a communication layer that the game engine provides. The developer must design this layer efficiently without causing it to crash or overload.

Unity framework provides remote actions to manage communications that are basically functions that can be called at the server that provides the current game session. The server can be either a dedicated one or there can be a machine which acts as a server and client at the same time. It does not matter what or where the server is. The remote actions will be executed on that specific machine [2]. Figure 1 shows the illustration of remote actions.

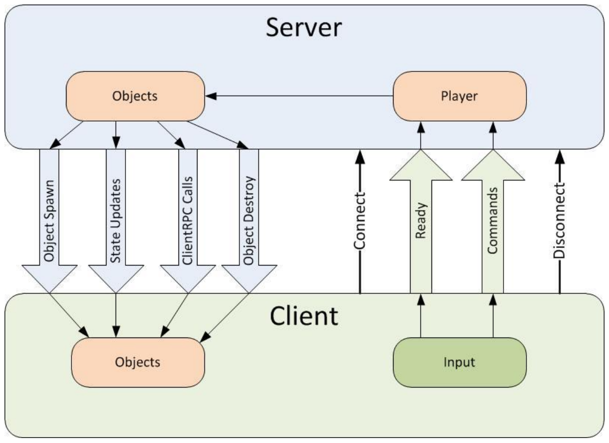

Figure 1. A diagram of the directions that remote actions take

Creating 3D games leads to dealing with 3D objects. In this case, customizing some objects to fit the requirements of the developed video game is required. A special tool is thus needed. Blender is a professional, free, and open-source 3D computer graphics tool, where it is possible to modify Unity Asset Store objects.

The designer uses Blender to create 3D objects from 2D images. In general, most of the avatars we see in games are started from a sketch on paper, then the 3D object is modeled using special software like Blender, and it is imported by a game engine to enable it to move around according to the end user's (player) commands [3].

DABABAT MULTIPLAYER GAME DEVELOPMENT

The planning phase precedes the system's development phase. In this case, the goal of planning was reaching a level where the game provides the following:

1. Environment to act as the battlefield (map).
2. Tank Avatar to act as the player.
3. Shell object to be the projectile from the gun.
4. Dynamic and Well-organized network handler.

Each aspect mentioned above has been tackled carefully during the development phase. This kind of game requires a team to build it, and it also takes a lot of time and patience. The minimal goals have been reached considering the time which has been given for the development of this particular genre of a video game, and the work has been done only by one person. Table 1 lists the names of the developers and the testers.

TABLE 1. The developer and the usability evaluators

Name	Specialization and year of study	Domain
Developer, Evaluator	Artificial Vision and Intelligence 1st year master's student	Software Developer; medium experience with video games
Evaluator	Artificial Vision and Intelligence 1st year master's student	Researcher in the image processing group; medium experience with video games

The Battle Field

The plan for the game scenes or the maps was to make a variety to choose from like:

- desert theme;
- city theme;
- green field theme.

But as mentioned before there was only enough time for one map which is the desert theme as shown in Figure 2.

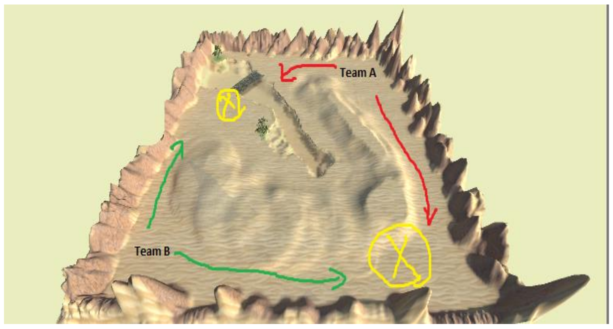

Figure 2. Desert Map

To make the theme more interesting for players, a river has been added in the middle to separate the battlefield into two halves. If any tank falls into the water, it will drown.

A terrain object has been used, as it provides the ability to shape any kind of grounds and then add textures to it. The whole object has been raised up to 15 measurement units, which means the river depth will be 15 measurement units below the surface of the map.

A series of mountains surround the map to act as a natural border, so no tank can drive off the map. Some hills have been added to separate the spawning points, so the players cannot hit each other from a long distance. Also, a bridge was installed at the far end of the river in case some of the players wish to flank the other team.

Tank Avatar

This is the most important element of the game that connects to all the other elements, as it took 60% of the development time. Unity has its own Asset Store from where the developer can get different types of useful objects, for example, 3D models, particles effects, animations etc. The main plan is to have 4 types of tanks:

- Heavy Tank (slow, well armored and good gun)
- Medium Tank (Faster, less armored and good gun)
- Light tank (fast, weak armored and bad gun)
- Tank Destroyer (slow, weak armored and great gun)

But with the time given, only the heavy tank has been made. From the variety of tanks that we found on the Asset Store, one of the most famous is the American heavy tank (T29). It has been selected for our game, and Figure 3 illustrates the model of the tank.

Figure 3. T29 Model [1][10]

After acquiring the tank's 3D Model from the store, the Blender application has been used to decompose it into 7 components. The tank will be reassembled in Unity. Figure 4 shows the tank after decomposition.

This procedure provides more flexibility to deal with the tank movements and the gun aiming. Camouflage texture is applied to the tank parts. Then two cameras and a fire effect are added to the tank avatar.

The biggest challenge with the tank object was making the gun follow the camera, so the player will be able to use the mouse to move the camera's view. For the sake of making the game more realistic, it is not preferable to hook the gun to the mouse movements, because real tank turrets can never move as fast as the mouse does. The only solution was making the turret follow the mouse with a slower movement speed. The hull (body of the tank) moves separately with the input buttons (W, A, S, and D) and this does not affect the turret horizontal movements or the gun vertical movements.

Figure 5 illustrates the diagram for the horizontal turret movements' algorithm. The same idea is applied to the gun's vertical movements.

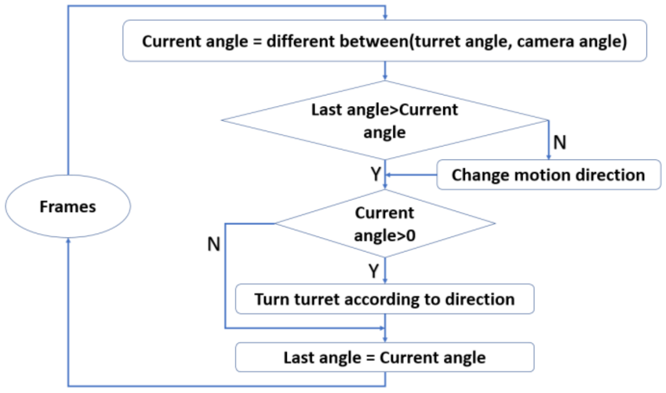
Current angle = different between(turret angle, camera angle)
Last angle>Current angle
N
Y
Change motion direction
Frames
Current angle>0
N
Y
Turn turret according to direction
Last angle = Current angle

Figure 5. Turret movements' diagram

To make the tank avatar feel realistic, some effects have been added to it:

- track sounds;

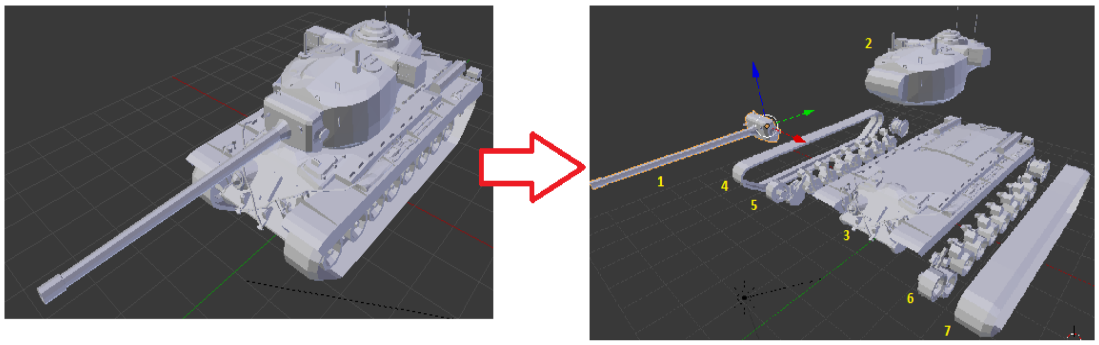

Figure 4 Model decomposition

- shell fire sound;
- muzzle fire flash;
- burning fire after the tank been destroyed;
- bubbles if the tank drowns in the river.

Everyone knows that guns have a recoil when they fire, so logically the big gun of the tank has to have some of its own and that is why we have used animations to make that effect to the gun. Basically, we designed the gun to be shorter for a little when it fires and it slowly returns back to its normal size just to make it feel more real as illustrated in figure 6.

Unity provides a great tool to do this kind of work by recording the movements and playing them from the C# code as needed, which in this case is the fire event.

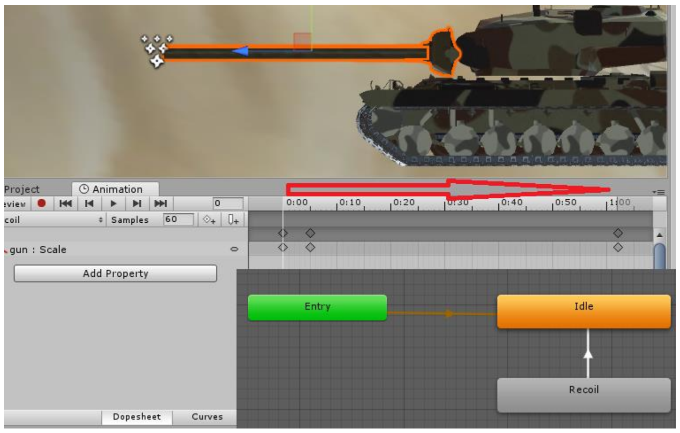

Figure 6. Recoil animation

Shell Object

Shells are what the tank fires toward enemies. In this game, the shell is an independent object, which is described by a position, and has velocity and mass, and can affect other objects physically by the laws of the game engine.

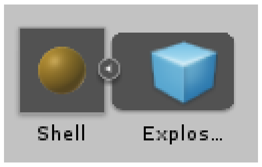

Figure 7. Shell object

Basically, the shell is a small ball which contains an explosion effect. It can be created from a point and travels toward another point, then hits whatever unfortunate thing in the way. Figure 7 shows the explosion effect connected to the shell object. To make the projectile feel real, some effects have been added to the shell:

- big explosion sounds;
- explosion particle effect on impact.

Tank Properties and Damage Handler

Each tank in the game has a data structure for the properties which contain the following:

- driving speed: integer
- tank turning speed: integer;
- turret turning speed: integer;
- gun vertical movement speed: integer;
- shell velocity: integer;
- health: integer;
- gun damage range: list[integer].

These properties will change the performance of the tank. When adding new tanks in the future, we should recalibrate the properties to fit the tank's descriptions and balance the gameplay to be as fair as possible.

The gun damage range is a one-dimensional array of integers. Each element represents how much health the shell can take from the other tank in case of a direct hit. These values should be proportional to the gun caliber.

When a tank fires a shell, a random number function will be invoked to get one of the elements from gun damage range array and this value will be subtracted from the health of the other tank in case the shell touches the other tank. In this game the array for the T29 gun damage range is [100, 200, 300, 400, 500] and the health property has been set to 1000. This means that it is possible to destroy a tank in 2 shots if you are lucky, or 10 shots if you miss your attacks. This mechanism made the gameplay much more enjoyable.

Network Handler

When everything is set (map, avatar, and shell) the only thing left to do is decide what to propagate over the communication layer. As mentioned before, Unity provides a tool to handle network operations, but this tool comes with more options than we actually need.

We had to create a custom network management class only to satisfy the game requirements. We used the existing network manager object and overridden functions to create a well-organized graphical interface to act as the game lobby as shown in Figure 8.

The interface requires the player to introduce the desired name and to select a team (red or blue). The team color represents the location on the map, where the player will spawn. The terminal can act as a server over some specific port or it can join another server in the game.

The network handler needs spawning locations to locate the players when the game starts, otherwise, the player will appear at the point (0, 0, 0) out of the map. Several spawning locations have been added to the game at the corners of the map above the terrain surface.

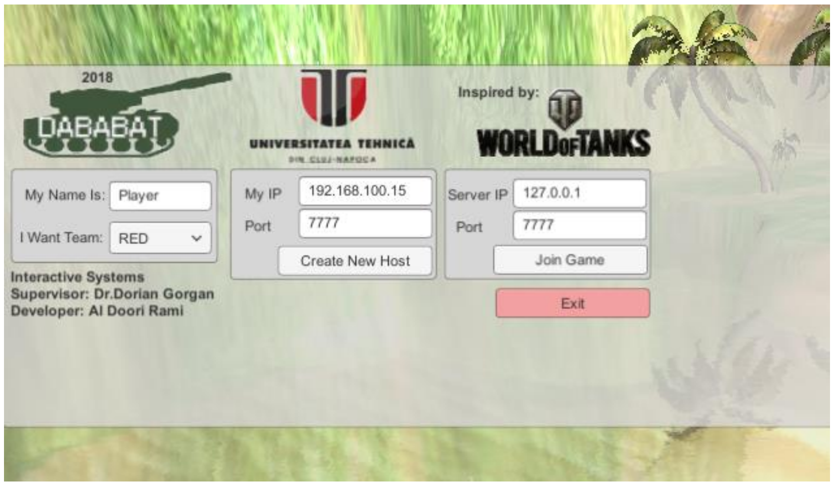

Figure 8. Game lobby

The following items can be spawned by the network handler:

- tank avatar;
- shell object.

The tank avatar uses the spawning locations when the game starts but the shell object will be spawned at the end of the gun.

We need to propagate the following data over the network:

1. game state (the number of players in the teams, who is still alive and who is dead, and what is the current health for each player and their names);
2. tanks locations on the map;
3. shell locations (contains the life cycle of the shell starting from the moment when it has been fired until it hits something and explodes or goes outside the map, after 5 seconds it will be disposed of automatically);
4. turret movements, so the players are able to see where the other tanks' turret is facing;
5. gun movements, so the players can see where the other tanks are aiming at.

The most important part of networking is not to overload the communication layer. Thus, a set of priorities have been established upon the network manager as follows:

1. the shell object movements have the highest priority because it is too fast and other tanks should be aware of the current location of the shell;
2. the tank movements;
3. the turret and gun movements;
4. the game's state;
5. the particle effect activation (this is a Boolean value that represents the state of the corresponding tank is the following cases:
    - Hit point = 0 and the tank is not drowning; activate the fire effect.
    - if the tank is drowning; activate the bubble particle effect.

Game Status Information

In this part of the project, several things need to be taken care of, like what information the player needs to know. According to our observations, the following list has been created:

- current health;
- gun reload time;
- time left to complete reload;
- the player's name;
- the names of the other players;
- the remaining health for other players;
- the number of players in the player's team;
- the number of players who died in the player's team;
- the number of players on the other team;
- the number of players who died in the other team;
- instructions about how to use the game.

It is more convenient to have the other player info above their tanks like the health bar and the player's name. The rest of the information is displayed on the screen in a way that will not affect the gameplay as seen in figure 9.

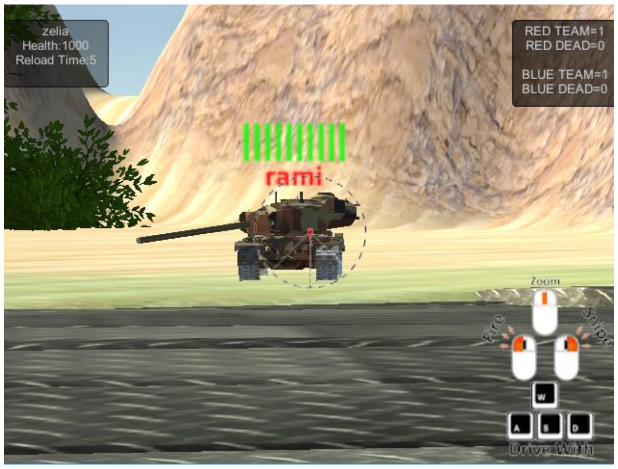

Figure 9. Game view shows the info provided to the user

Camera Handler

The game has three cameras. The main camera which views the game lobby is positioned in front of the waterfall which is the most beautiful scenery on the map. When the tank object has been spawned, it takes the view from the lobby camera and sets it to one of the two cameras which belong to it. The player can switch using the mouse's right click between the following cameras:

- commander view camera as shown in Figure 10;
- outside view camera as shown in Figure 11.

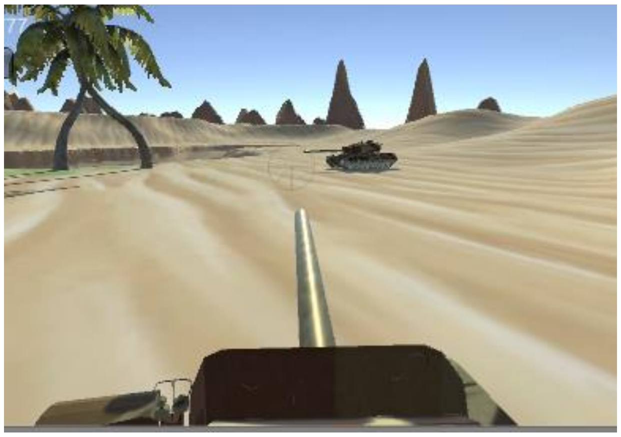
Figure 10. Commander view in sniper mode

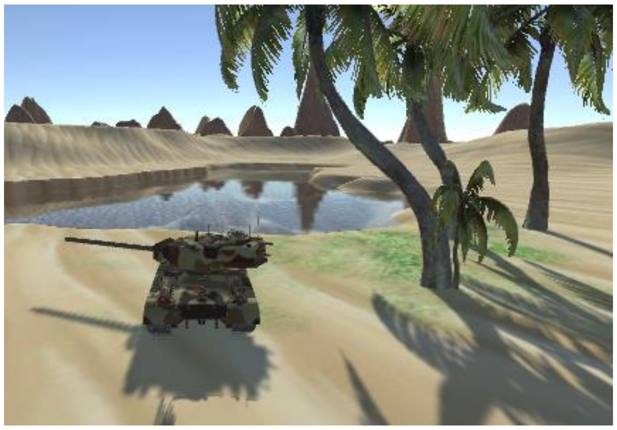
Figure 11. Outside view

USABILITY HEURISTICS FOR EVALUATION

When the game development phase is finished, we end up with a working copy of the game that is ready for testing. Usability heuristics methods have been used for testing because they are fast and do not require many resources. In order to perform such a method, we have to create some suitable heuristics for the project which can be detected by the testers. The testers have been informed about the game in general and the goals of the project.

TABLE 2. The mapping between game heuristics and Nielsen's heuristics

Game Heuristics		Nielsen's Heuristics	
ID	**Definition**	**ID**	**Definition**
H1	OS Compatibility	N4	Consistency and standards
H6	Consistency		
H2	Clarity	N2	Match between system and the real world
H3	Metaphors		
H10	Reality		
H4	Simplicity	N8	Aesthetic and minimalist design
H5	Feedback	N1	Visibility of system status
H7	Error prevention	N5	Error prevention
H8	Help and documentation	N10	Help and documentation
H9	Network Efficiency	N7	Flexibility and efficiency of use
H11	Enjoyment	N3	User control and freedom

Matching has been performed between the heuristics we created and the standard Nielsen's Heuristics, so the game heuristics will be as the following:

H1: OS Compatibility - the game should be easy to install and compatible, without additional software support. If needed, the package should contain the extra software support.

H2: Clarity - the interface should be easy to understand, using clear graphic elements, text, and language.

H3: Metaphors - use appropriate metaphors, making the possible actions easy to understand, through images and familiar objects.

H4: Simplicity - provide the necessary information in order to complete a task.

H5: Feedback - keep users informed on the games' progress, indicating both the global and the detailed state of the system. The application should deliver appropriate feedback on users' actions.

H6: Consistency - consistent in using language and concepts. The forms of data entry and visualization of the results should be consistent.

H7: Error prevention - prevent users from performing actions that could lead to errors and should avoid confusions that could lead to mistakes.

H8: Help and documentation - provide an easy to find, easy to understand, and complete online documentation. It should provide contextual help and glossary of terms for novice users [7].

H9: Network Efficiency - the end user must not feel any slow motion in the game or glitching.

H10: Reality - the end user must feel the laws of physics applied in the game the same way in real life, as well as the objects inside the game, have to behave as they are in real life.

H11: Enjoyment - the end user must have fun while playing the game.

Each game heuristics matches one of the Nielsen's Heuristics by meaning and goal as shown in Table 2. For conducting an experiment, we have two testers: the first will have the Game Heuristics and the second will have Nielsen's Heuristics and both of them have tested and evaluated the game according to the heuristics they been given. Each tester wrote a description of the problems that they have identified while testing as shown in Table 3.

TABLE 3. Number of usability problems identified by testers

Tester 1 using Game Heuristics		**Tester 2 using Nielsen's Heuristics**	
ID	**Number of problems**	**ID**	**Number of problems**
H1	0	N4	2
H6	3		
H2	0	N2	2
H3	0		
H10	1		
H4	1	N8	1
H5	0	N1	0
H7	0	N5	0
H8	0	N10	1
H9	2	N7	1
H11	2	N3	3
Total	**9**	**Total**	**10**

As a result, Nielsen's Heuristics tester detected the problems in the system in a more general point of view, but the Game Heuristics tester has described the problem specifically.

RESULTS ANALYSIS

Now we have enough data to prioritize the modifications on the game and fix the problems accordingly, so we sorted all the problems that have been detected by the testers starting from the highest problematics in the tester point of view. Table 4 illustrates the problems.

The two evaluation methods almost have similar results regarding the situation of this project and the available resources. Both testers got to the same main problems and described them independently.

TABLE 4. Results analysis

Description	**Value**
Problems detected by both testers	70%
Problems detected only by using Game Heuristics	10%
Problems detected only by using Nielsen's Heuristics	20%

The following problems came up while testing:

a) Watching out for the mass parameter of the objects when applying physics' laws to the game, because they can push each other away.

b) When creating a network game, it is recommended to start with the player's avatar before developing the network class.

c) Carefully deciding what to propagate over the network so the communication layer is not overloaded.

d) Setting a priority for the data needed to be sent over the network and acting accordingly, for example (the shell location while it been fired and travel through air to the target have a high priority over the current tanks' locations because they are slower).

e) The camera object within each client has it is own point of view, so if we have a 3D text, it should always face the right direction that is toward the current player.

f) The particle system (effects) cannot be transferred via the network, so synchronizing some Boolean values to activate them to be visible to the other clients is a good and a fast way to do it.

g) Using layers and tags to distinguish between objects in the scene like plyers and trees in case of a collision has been invoked.

CONCLUSIONS

Using a heuristics evaluation method is very effective in detecting problems with usability. It is important for a game to be easy to use and bug-free, so it can interact with humans as it supposed to do. Defining special heuristics for the game that we developed helped to produce a better feedback on the technical level and made it easier for the developer to make changes. The work would have been better if the project's team consisted of more than 5 people. This would also help with addressing each problem to the person responsible for it.

ACKNOWLEDGMENTS

The authors would like to thank the team behind the online game World of Tanks [10] for being the inspiration behind this project and to all the participants involved in the experiments that the present study required.

None of this would have been possible without the support from the Computer Science Department form the Technical University of Cluj-Napoca, Romania.

REFERENCES

[1] Wargaming Cyprus Corporate Registry WARGAMING GROUP LIMITED Archived October 11, 2016, worldoftanks.eu

[2] Anis Zarrad “Game Engine Solutions” INTECH 2018

[3] Tihomir Dovramadjiev Technical University of Varna, MTF, Department - Industrial “SPECIALISED ARCHITECTURAL FEATURES IN BLENDER 3D” Volume VI, 2016, Number 4: TECHNICAL STUDIES

[4] Yi Ping SHI , College of Electrical & Electronic Engineering, Shanghai University of Engineering Science, Shanghai, China “The Development of Tanks War Mobile Game based on Android System” MATEC Web of Conferences 63 6301035 MMME 2016

[5] Tonio Triebel , Benjamin Guthier, Richard Süselbeck, Gregor Schiele,Wolfgang Effelsberg “Peer-to-peer Infrastructures for Games”DBLP January 2008

[6] Max Lehn, Tonio Triebel, Christof Lengm, Alejandro Buchmann, Wolfgang Effelsberg, “Performance Evaluation of Peer-to-Peer Gaming Overlays” IEEE Xplore September 2010

[7] Rusu C., Roncagliolo S., Figueroa A., Rusu V., Gorgan D. , Evaluating the Usability and the Communicability of Grid Computing Applications. The Fifth International Conference on Advances in Computer-Human Interactions. ACHI 2012, pp.204-207, (2012).

[8] Sarmad Soomro, Wan Fatimah Wan Ahmad, Suziah Sulaiman Department of Computer & Information Science, University Teknologi PETRONAS ”Evaluation of Mobile Games with Playability Heuristic Evaluation System Sarmad Soomro1” 978-1-4799-0059-6/13/$31.00 ©2014 IEEE

[9] Wolmet Barendregt, Mathilde M. Bekker & Mathilde Speerstra Faculty of Industrial Design, TU Eindhoven “Empirical evaluation of usability and fun in computer games for children” Human-Computer Interaction - INTERACT'03 M. Rauterberg et al. (Eds.) Published by IOS Press, (c) IFIP, 2003, pp. 705-708

[10] TurboSquid, Inc. 935 Gravier St., Suite 1600, New Orleans, LA 70112, https://free3d.com/3d-model/t29-heavy-tank-83399.html