Design of a multiplayer mode for a virtual reality application for lower limb rehabilitation

A. Gordon Cabello de los Cobos

Departamento de Ingeniería Biomédica, Universidad CEU San Pablo, Madrid, España, a.gordon1@usp.ceu.es

Abstract

Gamification is a technique that uses gaming mechanics to improve another process. It can be applied to therapy, to increase the commitment and the enjoyment of the rehabilitation sessions by the patient. This project adds gamification strategies through a multiplayer mode to a Virtual Reality (VR) application designed for lower limb rehabilitation that uses an Inertial Measurement Unit (IMU). With this improvement, the resulting game will allow for group therapies and enable the interaction between players, which is hoped to increase the adherence to treatment of the patients.

1. Introduction

During these past years, the implementation of gamification, a technique which applies game mechanics to non-gaming environments such as healthcare [1], at rehabilitation has garnered interest from the scientific community [2]. Gamification can encourage patients to attend their rehabilitation sessions [3, 4] and increase their enjoyment of such sessions; the reward system that this technique offers produces a natural incentive to achieve the objectives that lead to the patient's recovery.

Another interesting development is the implementation of digital techniques on health, also called e-Health. An example of this trend is the use of Virtual Reality (VR). An example is its use in mental health therapies for different kinds of disorders, such as phobias or autism spectrum disorder [5]. However, it is also used in different fields of rehabilitation [6] such as upper limb function, where the controllers included in these devices can be very useful for tracking the patient's movements.

For the lower limb rehabilitation, it has been proved that the use of exoskeletons and novel interfacing techniques lead to a general improvement of the motor functions and can help with the transition from the clinic to the home-based rehabilitation settings [7].

The aim of this project is to implement a gamification algorithm and a multiplayer mode for a previously developed Unity VR game. The original game relied on an Inertial Measurement Unit (IMU) sensor for detecting the movement of the leg, which was used to control the velocity and movement of a boat or plane (depending on the game level). The game was developed for the Oculus Quest 2 standalone headset and uses hand-tracking as a controller, to provide a more immersive experience.

The multiplayer game will allow the patients to connect to an online room where all the patients can see each other boats or planes, during the session the patients will be able to activate some boosters that can affect themselves or the other patients depending on the booster selected.

2. Original VR Game

The original Unity application, which was expanded for this work, named “PedaleoVR” has 4 different scenes: The first one, “InitialScene” (Figure 1), loads when the application is started. This scene allows for the configuration of the username. This name will be used for associating the patient and the results obtained during the games. This scene also enables for the selection of the game level, so the patient can start the session.

The other 3 scenes, the game levels, “WeriumStandAloneShip” (Figure 2), “WeriumStandAloneClouds” (Figure 3) and “WeriumStandAloneTerrain” (Figure 4), are the different game levels that will be loaded depending on the patient's choice. The 3 levels work the same way: There is a transport, a boat, or a plane, where the user will be located and a sphere that will rotate depending on the velocity registered by the sensor. Since the sphere has different objects on it that act as a landscape, the player will perceive that he is the one moving around the “world”. The changes that will be performed at the 3 levels will be the addition of an exit button that will load the “InitialScene”, to avoid closing the application every time a new patient finishes its session; the addition of a gamification system that will give a random booster to the patient so it can play with the other users in the multiplayer room; and the addition of a multiplayer game manager that will allow some light interactions between the patients and the use of the online room. After the selected scene is loaded, the user will need to configure the sensor used and the velocity and number of cycles that they wanted to reach.

The main difference between the games is that, on the first one, the player will be inside a sailing boat, and for the other two, the player will be inside a plane, during a high-altitude flight if it’s the “WeriumStandAloneClouds” level, or during a low-altitude flight if it’s the “WeriumStandAloneTerrain” level.

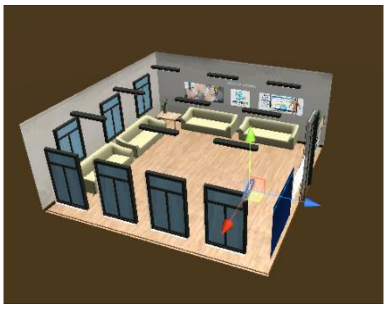
Figure 1. *Initial Scene overview with the player starting point at the axis indicator*

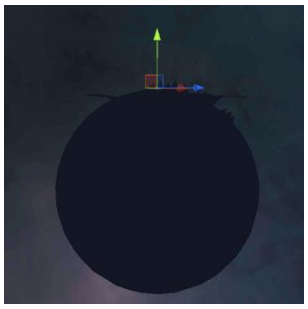
Figure 2. *WeriumStandAloneShip level overview with the boat at the axis indicator*

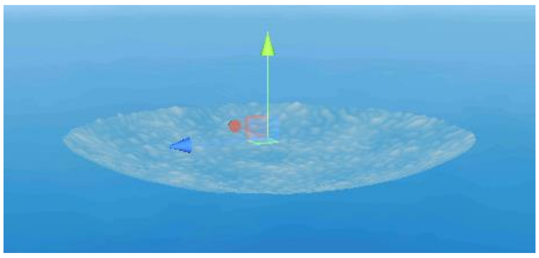
Figure 3. *WeriumStandAloneClouds level overview with the plane at the axis indicator*

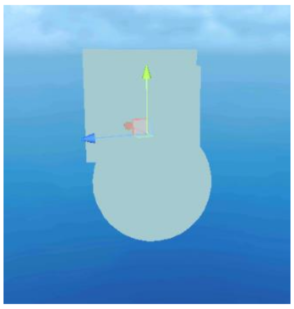
Figure 4. *WeriumStandAloneTerrain level overview with the plane at the axis indicator*

3. Project Development

The booster management and the exiting button incorporation were made by adding many methods and coroutines to the "GameManager.cs" script. Meanwhile the multiplayer was implemented by editing completely the "Launcher.cs" script and creating a multiplayer game manager that allows the pawn of the player in the scenes.

3.1. Exit button

The exit button (Figure 5) was developed following the same procedure as the buttons that were already used and located on the canvas "Nuevo_Canvas_Pedaleo" that appears after all the configuration of the sensor and the parameters are done.

Concerning the button's design, the prefab was imported from the Oculus Integration Toolkit Assets and the material used for distinguishing it from the rest of the buttons was "btn_mat_exit", whose base color is red. This material was applied to the ButtonMesh component, a child of the ButtonWiew, to apply the color only to the part of the button that are pressed.

For the actions of the button, the script "orientationButton.cs" was modified by adding a Boolean, (initially set to false), and the method that enables the scene change. The script works by detecting a collision with a child object called "sphere". If the collision is detected, the sphere's tag is "exit_btn", the button wasn't pressed, and the actual scene name is different from "InitialScene", then the scene "InitialScene" will call the GameManager's method ExitButton(), which loads the initial scene by using the SceneManager's method LoadScene, and the Boolean will be set to true for a short period, to avoid multiple collisions at the same time.

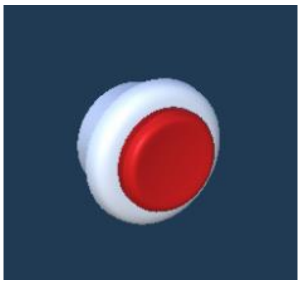
Figure 5. *Exit button model*

3.2. Multiplayer Mode

For the project, the multiplayer server system used was Photon, chosen because it allowed us to create a private server for free with a stable internet connection. The mode used was Photon Unity Networking (PUN), and the maximum amount of users that can share a room were set to 4. Also, a new canvas called "Sala_MP", that works the same way as the username canvas, was created.

The first thing that was needed for the implementation was to create the PUN app to have the server available and then import the PUN 2 asset from the Asset Store to the game. After the installation was finished, the PUN app ID was configured on the PUN wizard.

After the first step, the "Launch.cs" script that is included in the asset was edited. The most important variables declared were an integer for the maximum of players, two static strings for the player's name and for the room name, a static integer called "arena" that will be used for identifying the scene that the player wants to load and

Boolean that will allow to check if the room was successfully joined.

Two methods were also added: one for joining the room, JoinRoom, and another for loading the desired scene, LaunchArena.

The script "GoToGame.cs", that was already created on the project, will allow to configure all the parameters. The player's name will be the same as the one introduced for the registration process and the room name will be introduced on the new canvas "Sala_MP". When the Ok button from the canvas "Sala_MP", is pressed (works the same way as the exit button) the room will be joined and the master (the user to which all the rest will be connected) will be set to the first user that connects to the room. This is made automatically by Photon. After the user chooses the level and presses the Ok button from that panel, the arena will be loaded according to the variable "arena" which will be set to the value of the scene of the minigame selected.

The next part was the creation of the "MP_GameManager.cs" script. This script will allow to generate the prefabs for the plane or boat objects, depending on the scene, from the Resource folder on the indicated spawn points. The spawn points are empty object distanced ones from others in other to have a reference for the prefabs to appear. These instances will be created by a for loop that will create the prefabs only for the players loaded on the room.

Finally, the components Photon Transform View and Photon View were added to the prefabs, and the spawn points needed were defined.

3.3. Gamification Strategies Implementation

There will be three options for the booster to be. These options are "Gas", which will increase the speed for a time and play a particle system that will simulate the smoke from a motor; "Banco de peces", which will decrease the speed from the others from a limited time; and "Nube", which will activate a particle system that will interfere with the vision of the others. For the methods that affect other players, the attribute PunRPC is given. This attribute allows such methods to interact with the clients' that are in the same multiplayer room.

All the variables, methods and coroutines required for the gamification were stored on the "GameManager.cs" script.

For the buttons ReflexButton and BoosterButton the same model (Figure 6) was used.

Figure 6. *Reflex and Booster button model*

For showing the sprites a canvas was created (Figure 7) and depending on the booster and uses left the image will change.

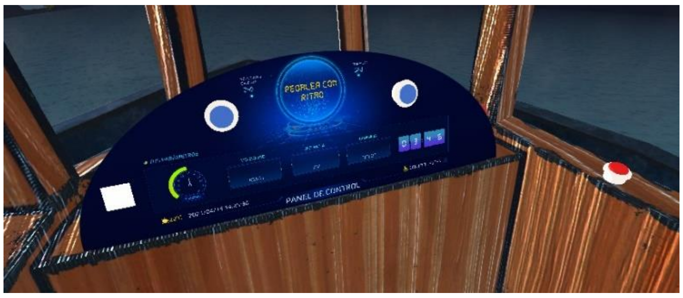

Figure 7. *Canvas "Nuevo_Canvas_Pedaleo", with the exit button (red), the booster button (right blue), reflex button (left blue) and the place where the sprites will appear (white square)*

The strategy developed works using two coroutines that implements different methods. The first coroutine, explained in Algorithm 1, allows the activation and deactivation of the ReflexButton and, if the button pressed on time, generates the uses, the booster and allows the player to use the booster.

Algorithm 1: Reflex button coroutine and collision

```
Input: Boolean reflejo
Input: Button reflex_button
Output: integers usos and i_temporal
Output: Boolean reflejo
Output: Activation of booster_button and start of it's coroutine
1    While reflejo is false
2        Activate reflex_button
3        Wait for a random number of seconds between 1 and 4
4        Deactivate reflex_button
5        Wait for a random number of seconds between 1 and 4
6    If reflex_button is collided while active
7        usos ← generarUsos()
8        generarBooster()
9        reflejo ← true
10       Deactivate reflex_button
11       Activate booster_button
12       Stop this coroutine
13       Start the booster coroutine
     end
```

The method generarUsos() generates a random integer between 1 and 3 if the number of uses is 0 and returns it. The method generarBooster() will also generate a random integer between 1 and 3 if the number of uses is 0, this integer will be stored in "i_temporal" and will allow to know which booster has been selected. If "i_temporal" it's 1 it's "Gas", if it's 2 then "Banco Peces" and if it's 3 then is "Nube".

The second coroutine, explained in Algorithm 2, maintains the booster button activated and when it's pressed then the action it's executed, and the uses left, and sprites updated.

```
Algorithm 2: Booster button coroutine

Input: Button booster_button
Input: integers usos and i_temporal
Input: Boolean boosterPressed and reflejo
Output: Action of the booster
Output: Boolean reflejo
1   While usos > 0
2       Wait for boosterPressed to be true
3       usos ← usos - 1
4       actualizarSprites()
5       boosterPressed ← false
6   Deactivate booster_button
7   reflejo ← false
8   Set the sprites to empty
9   Stop this coroutine
10  Start the reflex button coroutine
    end
```

3.4. Code

For downloading the project, a shared folder on OneDrive can be accessed by scanning a QR code (Figure 8).

Figure 8. *QR code for accessing the OneDrive folder*

4. Conclusions

Gamification is a technique that uses the reward system provided by the games for increasing the patient's enjoyment of the sessions and the commitment. Both outcomes are the result of a reward system that is included into the therapy. One way of implementing gamification in the rehabilitation is by using VR, since these technologies transport the patient to a more pleasant space where the rehabilitation can take part.

This work updates an already existing VR rehabilitation game. This update allows to go back to the lobby in case the session has finished, a multiplayer system that will allow up to 4 patients on same room and a gamification system so the patients can interact between them during the session. This update will allow to perform multiple sessions at the same time and to change the enjoyment of the patients during such sessions. These changes will result on a higher commitment by the patients to go to the sessions, accelerating their recovery.

Some future improvements that can be made to this work are: Increasing the maximum number of users per multiplayer room, creating a ranking system like the ones that appear on race games and incorporating more gamification strategies in the form of boosters.

Although the application basic functions are finished, there are some bugs that must be solved which came from the multiplayer implementation. The first of them is that the hands of the master client do not instantiate correctly and, instead of showing them normally, they appear fixed at a space. The second one is that the master is the only one that can see the effects of the booster "Nube".

At last, but not least, we want to show our gratitude to Ana Rojo Agustí, the developer of the original application, in appreciation for the help she has lent on us.

5. Bibliography

[1] Rodriguez, P. (2021, 10 March). *The Ultimate Definition of Gamification (With 6 Real World Examples)*. Growth Engineering. https://www.growthengineering.co.uk/definition-of-gamification/

[2] Hamari, J., Koivisto, J., & Sarsa, H. (2014, January). Does gamification work?--a literature review of empirical studies on gamification. In 2014 47th Hawaii international conference on system sciences (pp. 3025-3034). Ieee.

[3] Grant, S., & Betts, B. (2013, May). Encouraging user behavior with achievements: an empirical study. In 2013 10th Working Conference on Mining Software Repositories (MSR) (pp. 65-68). IEEE.

[4] Sardi, L., Idri, A., & Fernández-Alemán, J. L. (2017). A systematic review of gamification in e-Health. Journal of biomedical informatics, 71, 31-48.

[5] Emmelkamp, P. M., & Meyerbröker, K. (2021). Virtual reality therapy in mental health. Annual Review of Clinical Psychology, 17, 495-519.

[6] Rutkowski, S., Kiper, P., Cacciante, L., Cieślik, B., Mazurek, J., Turolla, A., & Szczepańska-Gieracha, J. (2020). Use of virtual reality-based training in different fields of rehabilitation: A systematic review and meta-analysis. Journal of Rehabilitation Medicine.

[7] Mubin, O., Alnajjar, F., Jishtu, N., Alsinglawi, B., & Al Mahmud, A. (2019). Exoskeletons with virtual reality, augmented reality, and gamification for stroke patients' rehabilitation: systematic review. JMIR rehabilitation and assistive technologies, 6(2), e12010.