David López Gasparin

FRAMEWORK PER A JOC DE CARTES ONLINE

TREBALL DE FI DE GRAU

dirigit per María Montserrat García Famoso

Grau d'Enginyeria Informàtica

UNIVERSITAT ROVIRA I VIRGILI

Tarragona

2021

Resum

Aquest Treball de Fi de Grau consta d'una implementació d'un joc de cartes multijugador i multiplataforma. Aquest ha estat dissenyat per oferir la possibilitat de jugar a múltiples jocs de cartes en un de sol.

Per aconseguir aquest objectiu s'ofereixen diferents opcions de configuració bàsiques i una interfície intuïtiva per simular el joc com si estiguessis jugant en la taula de casa teva.

L'aplicació està formada per un fons que simula una taula i al damunt es crea una baralla de cartes la qual pots agafar, triar i moure les cartes al teu gust, tot sincronitzat en el mateix instant amb els altres jugadors. Per facilitar la jugabilitat s'han afegit opcions per mesclar i repartir aquesta baralla de cartes.

Finalment s'han fabricat funcions de control de les cartes per si en un futur es projecta ampliar les opcions que el joc ofereix.

El joc ha estat implementat amb el motor de joc Unity, aquest permet crear i desenvolupar jocs amb totes les opcions possibles i permet compilar en múltiples plataformes. Per a l'apartat multijugador s'ha utilitzat el paquet de Photon en la seva versió gratuïta i per a la incrustació del joc en una pàgina web l'eina WebGL.

Resumen

Este Trabajo de Fin de Grado consta de una implementación de un juego de cartas multijugador y multiplataforma. Este ha estado diseñado para ofrecer la posibilidad de jugar a múltiples juegos de cartas en uno solo.

Para conseguir este objetivo se ofrecen diferentes opciones de configuración básicas y una interficie intuitiva para simular el juego como si estuvieras jugando tú mismo en la mesa de tu casa.

La aplicación está formada por un fondo que simula una mesa y encima de esta se crea una baraja de cartas donde puedes coger, elegir y mover las cartas a tu gusto, todo sincronizado en el mismo instante con los otros jugadores. Para facilitar la jugabilidad se han añadido opciones para mezclar y repartir esta baraja de cartas.

Finalmente se han fabricado funciones de control de las cartas por si en un futuro se proyecta ampliar las opciones que el juego ofrece.

El juego ha estado implementado con el motor de juego Unity, este permite crear y desarrollar juegos con todas las opciones posibles y permite compilar en múltiples plataformas. Para el apartado multijugador se ha utilizado el paquete de Photon en su versión gratuita y para la incrustación del juego en una página web se ha utilizado la herramienta WebGL.

Abstract

This Final Degree Project consists of an implementation of a multiplayer and multiplatform card game. It has been designed to offer the ability to play multiple card games in one.

To achieve this objective, different basic configuration options and an intuitive interface have been implemented to simulate the game as if you were playing yourself at your home table.

The application is made up of a background that simulates a table and on it a deck of cards is created where you can take, choose, and move the cards to your liking, all synchronized at the same time with other players. To facilitate the gameplay, options have been added to mix and distribute this deck of cards.

Finally, card control functions have also been manufactured in case it is planned to expand the options that the game offers in the future.

The game has been implemented with the Unity game engine, it allows you to create and develop games with all possible options and compile on multiple platforms. For the multiplayer section, the Photon package has been used in its free version. WebGL tool has been used for the incrustation of the game on a web page.

Índex

1 INTRODUCCIÓ 10
2 DESCRIPCIÓ DEL PROJECTE 10
3 REQUISITS 11
3.1 FUNCIONALS 11
3.1.1 Lobby 11
3.1.2 Dintre del joc 11
3.2 NO FUNCIONALS 11
4 ANÀLISI DELS REQUISITS FUNCIONALS 12
4.1 DIAGRAMA DE CLASSES 12
4.1.1 Classes del Lobby 12
4.1.2 Classes dintre del joc 13
4.2 DIAGRAMA DE SEQÜÈNCIA DELS CASOS D'ÚS 14
4.2.1 Especificació textual dels casos d'ús 15
5 DISSENY 20
5.1 ARQUITECTURA DE L'APLICACIÓ 20
5.2 DISSENY INTERFÍCIE GRÀFICA 21
5.2.1 Entrada Nick 21
5.2.2 Seleccionar opció 21
5.2.3 Disseny de la sala 22
5.2.4 Selecció de Sala 22
5.2.5 Dintre de la Sala 23
5.3 DISSENY DEL JOC 23
6 IMPLEMENTACIÓ 25
6.1 CONFIGURACIÓ DE L'ENTORN DE DESENVOLUPAMENT 26
6.1.1 Configuració Unity 26
6.1.2 Configuració Photon 26
6.2 CREACIÓ DEL LOBBY 28
6.2.1 Implementació entrada Nick 28
6.2.2 Implementació Selecciona opcions 29
6.2.3 Implementació Creació de la Sala 30
6.2.4 Implementació Selecció de la Sala 32
6.2.5 Implementació Dintre de la Sala 33
6.3 IMPLEMENTACIÓ DEL JOC 34
6.3.1 Punt de partida 34
6.3.2 Implementació de la Sincronització 36
6.3.3 Creació de la Baralla 38
6.3.4 Funcions de mesclar i repartir 39

6.3.5 Privadesa de les Cartes *42*
6.3.6 Mostrar la teva carta *43*
6.3.7 Menú d'opcions dintre del joc *44*
6.3.8 Control del Joc *44*
6.3.9 Transferència d'informació *46*
6.3.10 Implementació del joc en una pàgina web *46*
7 TEST DEL PROGRAMA 47
7.1 CONNEXIÓ A PHOTON 47
7.2 CREAR I CONFIGURAR SALA 48
7.3 CERCA DE SALES 48
7.4 JUGADORS DINTRE DE LA SALA 49
7.5 MOURE CARTA 50
7.6 PRIVADESA CARTA 50
7.7 MOSTRAR CARTA 51
7.8 MESCLAR CARTES 52
7.9 REPARTIR CARTES 53
8 CONCLUSIONS 54
9 RECURSOS UTILITZATS 55
10 ANNEX 56
10.1 INSTAL·LACIÓ 56
10.2 REQUISITS 56
10.3 ERRORS CONEGUTS 56

Índex Figures

Figura 1. Diagrama de classes del Lobby 12
Figura 2. Diagrama de classes del joc 13
Figura 3. Diagrama de seqüència dels casos d'ús 14
Figura 4. Interficie principal d'entrada del Nick 21
Figura 5. Interfície on ens deixa crear o buscar sales 21
Figura 6. Descripció visual de la configuració de sala 22
Figura 7. Descripció visual de la selecció de sala 22
Figura 8. Descripció visual de la sala de jugadors 23
Figura 9. Representació del sistema Drag & Drop 24
Figura 10. Logotip d'Unity 25
Figura 11. Logotip de Photon 25
Figura 12. Logotip del llenguatge C# 25
Figura 13. Logotip de Microsoft Visual Studio 25
Figura 14. Photon en l'Asset Store 26
Figura 15. "Demos" de Photon 26
Figura 16. Creació de l'aplicació en la pàgina web de Photon 27
Figura 17. Dashboard en la pàgina web 27
Figura 18. Referència a entradaUsuari dintre l'script 28
Figura 19. Referència a la funció que crida el botó al ser premut 28
Figura 20. Resultat de la implementació de l'entrada de Nick 29
Figura 21. S'activa un panell i a la vegada es desactiva l'altre 29
Figura 22. Referència del botó a la funció per buscar sala 29
Figura 23. Implementació de la selecció d'opcions 30
Figura 24. Referencia d'un dels botons a la funció per triar baralla 30
Figura 25. Visió de la selecció de baralla 30
Figura 26. Visió de les entrades personalitzades 31
Figura 27. Visió dels toggle creats en un grid layout group 31
Figura 28. Visió de l'entrada del nombre de jugadors i com s'ordenaran aquests 32
Figura 29. Implementació final de la configuració de sala 32
Figura 30. Implementació selecció de sala 33
Figura 31. Implementació dintre de la sala 33
Figura 32. Referència a la imatge del sprite 34
Figura 33. Sense correcció 35
Figura 34. Amb correcció 35
Figura 35. Script referenciat a dintre de l'objecte 35
Figura 36. Llistat de components de Photon que es poden afegir 36
Figura 37. Visualització de la col·lecció de cartes en format .png 38
Figura 38. Array de referències als sprites de les cartes 38
Figura 39. Botó utilitzat per al menú 44
Figura 40. Mostra les opcions al obrir el menú 44
Figura 41. Prova de la introducció de Nick i connexió 47
Figura 42. Resultat positiu de la connexió 47
Figura 43. Mostra de la llista de sales 48
Figura 44. Mostra de la interacció dels jugadors a dintre de la sala 49
Figura 45. Prova de sincronització de moviment de les cartes 50
Figura 46. Prova de privacitat de la carta 50
Figura 47. Prova per mostrar el contingut de la carta 51
Figura 48. Primera mescla de cartes 52
Figura 49. Segona mescla de cartes 52
Figura 50. Configuració i prova de repartir cartes 53

Índex Codis

Codi 1. Extracció del text introduït a partir de la referència 28
Codi 2. Funció OnConnectedToMaster 29
Codi 3. Funció OnRoomListButtonClicked 32
Codi 4. Variables elementals 34
Codi 5. Funció de referència OnMouseDown 34
Codi 6. Funció de referència OnMouseUp 35
Codi 7. Funció de referència Update 35
Codi 8. Sincronització de la posició i rotació de les cartes 37
Codi 9. Actualització de la posició i rotació en els jugadors no propietaris 37
Codi 10. Exemple de funcions RPC 38
Codi 11. Exemple de crida a funció RPC 38
Codi 12. Array que defineix quines cartes formen part de la baralla 39
Codi 13. Declaració de baraja, la llista que conté les cartes, dintre de la classe Baraja 39
Codi 14. Funció que inicialitza la baralla 39
Codi 15. Funció que ens ofereix Photon per instanciar objectes en la xarxa 40
Codi 16. Variables per a la creació de cartes 40
Codi 17. Implementació de l'algoritme Durstenfeld 40
Codi 18. Implementació de la funció per mesclar la baralla 41
Codi 19. Implementació de la funció per repartir les cartes 42
Codi 20. Variable que ens diu si la carta està assignada 42
Codi 21. Comprovació d'assignació de la carta i posterior requisit de propietari 42
Codi 22. Crida realitzada si volem assignar cartes 43
Codi 23. Control dels dobles click 43
Codi 24. Canvi dels elements necessaris per mostrar la carta 43
Codi 25. Variables extretes de la configuració de partida 46
Codi 26. Evita la destrucció de les dades en el canvi d'escena 46

Índex Taules

Taula 1. Joc de proves connexió al servidor 47
Taula 2. Joc de proves creació i configuració de sala 48
Taula 3. Joc de proves cerca de sala 49
Taula 4. Joc de proves interacció a dins de la sala 49
Taula 5. Joc de proves moviment de cartes 50
Taula 6. Joc de proves privadesa de les cartes 51
Taula 7. Joc de proves mostrar contingut de la carta 51
Taula 8. Joc de proves mescla de les cartes 52
Taula 9. Joc de proves repartiment de les cartes 53

1 Introducció

Els videojocs de cartes online normalment estan destinats a un únic tipus de joc, aquests no poden variar i el seu únic objectiu es poder jugar al joc per al que han estat creats.

Amb la idea de canviar aquest comportament s'ha pensat en desenvolupar un framework[1] que sigui multijugador per a jugar a cartes online.

Per a canviar de joc nomes s'hauria de triar entre les opcions que estiguin disponibles en el menú i així canviar el comportament, posteriorment els mateixos jugadors decidirien les noves normes a seguir tal i com es fa en un joc de taula en temps real.

El projecte esta enfocat a tots aquells jugadors que no troben una forma de jugar el seu joc de cartes online, aquesta plataforma permetrà que els jugadors que no disposen d'una plataforma on jugar el seu joc de cartes favorit online puguin disposar d'una plataforma on jugar-hi.

Aquesta característica també donarà lloc a la creació de noves formes de jugar a cartes online degut a la seva flexibilitat.

2 Descripció del projecte

Per al desenvolupament d'aquest framework s'utilitzarà la plataforma de desenvolupament Unity. Aquesta porta un motor de joc molt flexible ja integrat. El llenguatge de programació utilitzat és el **C#**, aquest està per defecte incorporat en la plataforma i el que s'utilitza per escriure els scripts.

Unity conté la seva pròpia tenda d'eines i llibreries, algunes són gratuïtes i altres de pagament, aquesta s'anomena "Asset Store".

Per a la plataforma online s'utilitzarà l'asset "Photon", aquesta eina incorpora una llibreria anomenada "Photon Network" que simplifica de forma significativa la implementació online del projecte. També simplifica la seva possible escalabilitat amb la utilització dels servidors de la plataforma.

Per a l'elaboració dels scripts en C# s'ha utilitzat la plataforma de programació Visual Studio creada per Microsoft.

Aquest projecte no ha requerit d'una coordinació específica, s'han marcat uns objectius específics i a partir de l'aprenentatge de les eines utilitzades i el seu funcionament s'han anat implementant les característiques del projecte.

Des del punt de vista formatiu aquest projecte ajudarà a entendre el funcionament dels videojocs en línia, la complexitat d'aquests augmenta davant dels videojocs offline[2] ja que es requereix la sincronització dels jugadors en cadascuna de les possibles opcions del joc.

[1] Conjunt d'eines que serveixen de base per al desenvolupament de nou software.
[2] Fora de línia.

3 Requisits

3.1 Funcionals

3.1.1 Lobby[3]

- Pàgina web on s'incrustarà el joc
- Entrada de nom de usuari
- Opció de creació de sala
- Possibilitat de configuració de la sala quan es crea una sala
- Opció per buscar una sala ja creada
- Possibilitat de triar una sala dintre de la cerca de sales
- Opció per triar si estàs llest per començar el joc
- Confirmació de començar el joc només per al creador de la sala

3.1.2 Dintre del joc

- Possibilitat de moure les cartes
- Sincronització del moviment de les cartes entre els jugadors
- Les cartes assignades a un jugador no poden ser vistes per altres jugadors
- Possibilitat d'ensenyar les cartes als altres jugadors
- Possibilitat de mesclar les cartes
- Possibilitat de repartir les cartes

3.2 No funcionals

- Escalabilitat – Que el sistema pugui admetre mes usuaris en un futur si el joc progressa en el seu desenvolupament.
- Els usuaris hauran de poder aprendre com funciona el joc en menys de 10 minuts
- L'aplicació ha de tenir un disseny responsive[4] amb l'objectiu de que es pugui adaptar a les múltiples resolucions i maquinaris que hi ha.

[3] Sala d'espera
[4] Disseny adaptable a tots els dispositius

4 Anàlisi dels requisits funcionals

4.1 Diagrama de classes

4.1.1 Classes del Lobby

La classe **LobbyMainPanel** conté tots els camps i mètodes necessaris per al correcte funcionament de la interfície principal.

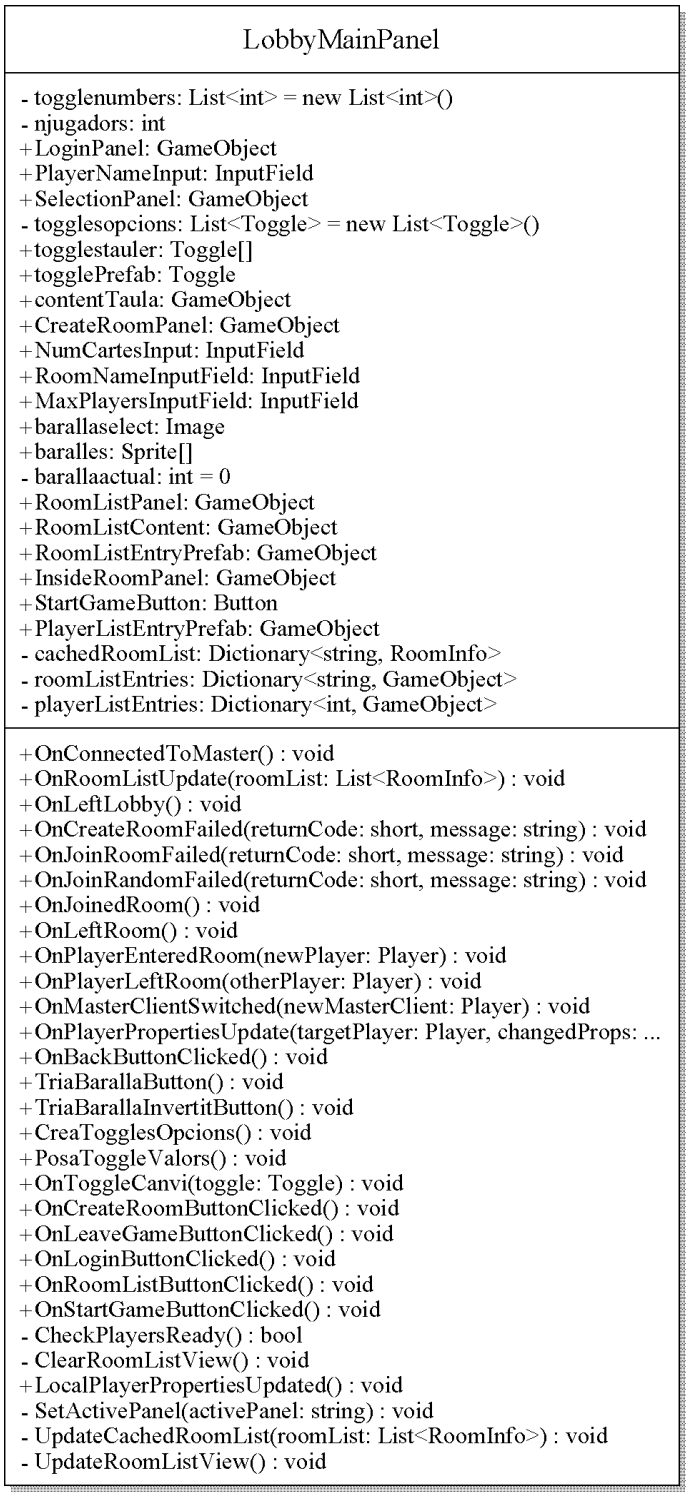

Figura 1. Diagrama de classes del Lobby

4.1.2 Classes dintre del joc

Les classes dintre del joc s'encarreguen d'implementar totes les funcionalitats que han d'estar disponibles per als jugadors quan entren al joc.

La classe **Jugador** ja és implementada juntament amb l'extensió de Photon per a Unity, s'han inclòs únicament les classes utilitzades per al funcionament de l'aplicació.

Per al jugador creador de la sala hi ha una **Baraja** quan aquest inicia el joc, aquesta **Baraja** permet identificar i utilitzar amb posterioritat tots els elements **DragDrop**, els quals son els scripts que implementen tots els **GameObject Carta** instanciats.

JuegoCartasGameManager s'encarrega d'implementar totes les funcions i mètodes per a la correcta interacció dels usuaris amb els mètodes de les demés classes.

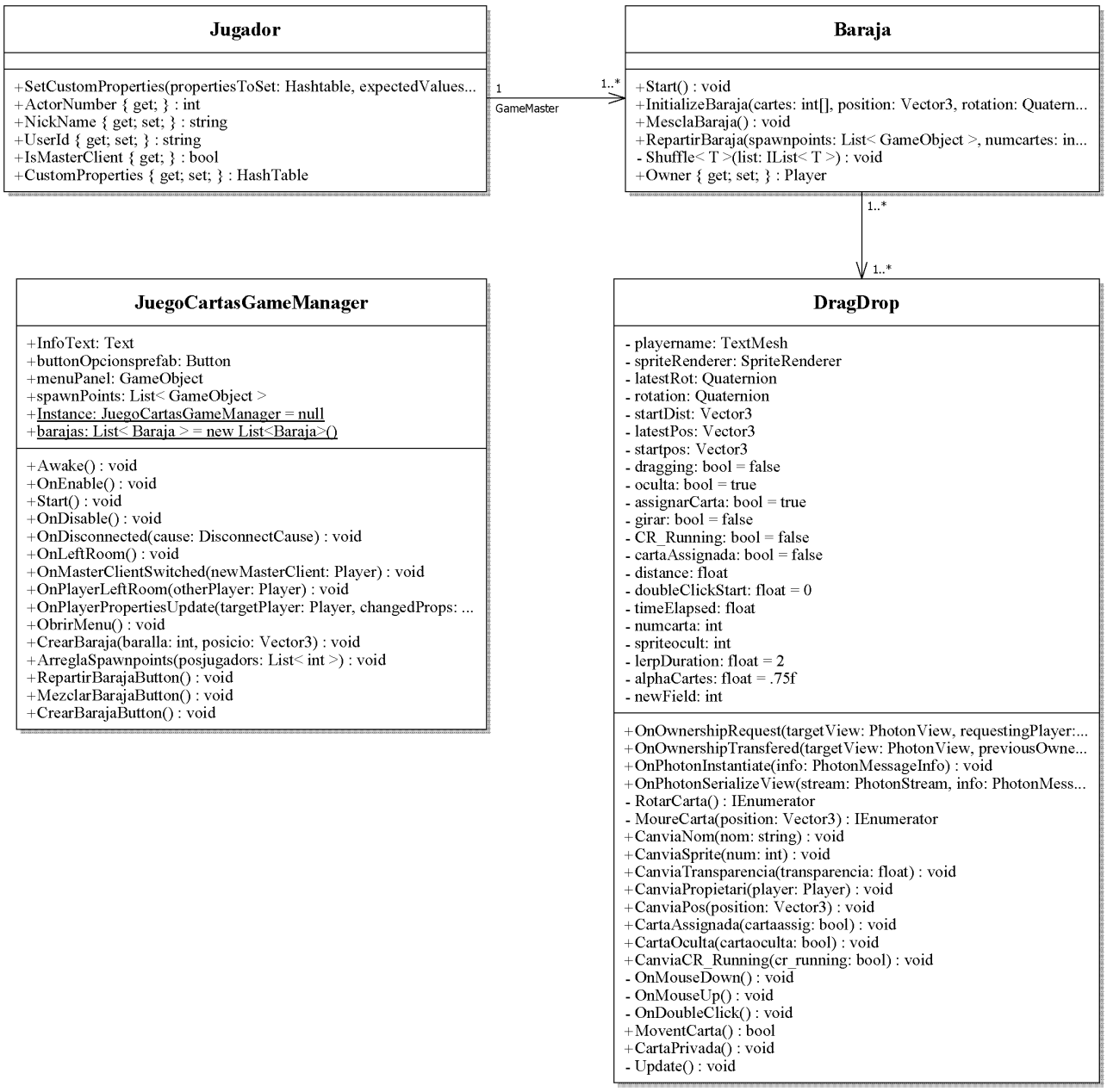

Figura 2. Diagrama de classes del joc

4.2 Diagrama de seqüència dels casos d'ús

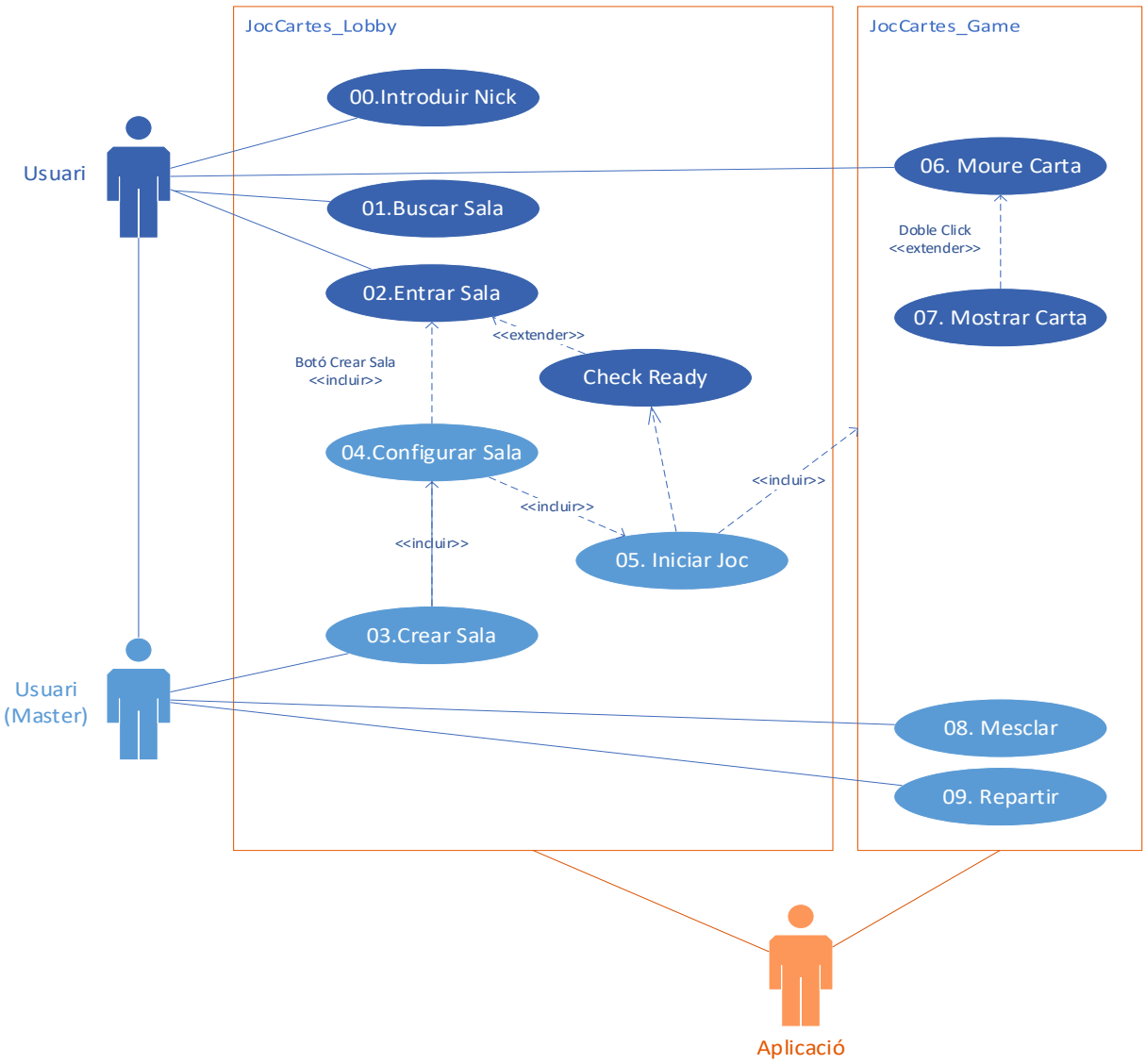

Figura 3. Diagrama de seqüència dels casos d'ús

4.2.1 Especificació textual dels casos d'ús

Els casos d'ús s'han dividit en dos sistemes diferents, *JocCartes_Lobby*, el qual fa referència als casos d'ús que es duen a terme dintre del Lobby, interfície que permet al usuari identificar-se, crear o unir-se a una sala. *JocCartes_Game* fa referència al sistema que implementa el joc i que permet jugar als usuaris que anteriorment han passat pel Lobby.

4.2.1.1 Cas d'ús 00. Introduir Nick[5]

Resum de la funcionalitat: Permet decidir el nick de l'usuari

Paràmetres d'entrada: Nick del usuari

Paràmetres de sortida: Cap

Actors: Usuari, Usuari(Màster)

Precondició: Cap

Postcondició: Connexió al servidor

Procés normal principal:

1. L'aplicació mostra en pantalla la interfície d'entrada de Nick
2. L'usuari introdueix el seu Nick en l'entrada de text
3. L'usuari prem el botó d'entrar
4. L'aplicació guarda el nom de l'usuari dintre de les seves dades
5. L'aplicació connecta l'usuari al servidor

Alternatives de procés

4a. L'usuari no ha introduït un Nick
    4a1. L'aplicació genera un Nick aleatori per al usuari

4.2.1.2 Cas d'us 01. Buscar Sala

Resum de la funcionalitat: Deixa escollir una sala al usuari

Paràmetres d'entrada: Cap

Paràmetres de sortida: Cap

Actors: Usuari, Usuari(Màster)

Precondició: Cap

Postcondició: Cap

Procés normal principal:

1. L'aplicació mostra en pantalla la llista de sales disponibles
2. L'usuari decideix quina sala és la que vol escollir

[5] Àlies o pseudònim.

Alternatives de procés

1a. No hi ha cap sala per mostrar
    1a1. L'usuari prem el botó de tornar enrere
    1a2. L'aplicació mostra al usuari la interfície de crear o buscar sala

4.2.1.3 Cas d'ús 02. Entrar Sala

Resum de la funcionalitat: Permet entrar a l'usuari a dins d'una sala

Paràmetres d'entrada: Sala escollida per l'usuari

Paràmetres de sortida: Cap

Actors: Usuari, Usuari(Màster)

Precondició: Cap

Postcondició: Connexió a la sala

Procés normal principal:

1. L'usuari prem damunt de la sala per entrar
2. L'aplicació connecta al usuari amb la sala escollida
3. L'aplicació mostra al usuari la llista d'usuaris a dintre de la sala

Alternatives de procés

2a. La sala seleccionada ja no està disponible o està plena
    3a1. L'aplicació actualitza el llistat de sales disponibles
    3a2. Torna al pas 1

4.2.1.4 Cas d'ús 03. Crear Sala

Resum de la funcionalitat: Permet a l'usuari crear una sala

Paràmetres d'entrada: Cap

Paràmetres de sortida: Cap

Actors: Usuari(Màster)

Precondició: Cap

Postcondició: Creació de la sala

Procés normal principal:

1. L'aplicació carrega la interfície de configuració de sala
2. Cas d'ús 04. Configurar Sala
3. S'afegeix la sala en la llista de sales disponibles

4.2.1.5 Cas d'ús 04. Configurar Sala

Resum de la funcionalitat: Permet a l'usuari configurar una sala

Paràmetres d'entrada: Baralla triada, nom de la sala, nombre de jugadors, nombre de cartes per repartir, posició dels jugadors

Paràmetres de sortida: Cap

Actors: Usuari(Màster)

Precondició: Cap

Postcondició: Configuració de la sala

Procés normal principal:

1. L'aplicació mostra les opcions de configuració
2. L'usuari entra totes les dades necessàries per configurar la sala
3. L'usuari prem el botó de Crear Sala
4. L'aplicació guarda les dades introduïdes en el sistema

Alternatives de procés

3a. L'usuari prem el botó de tornar enrere
    3a1. L'aplicació carrega la interfície per crear o buscar sala

4a. L'aplicació detecta que alguna dada falta o ha estat mal introduïda
    4a1. L'aplicació genera les dades per defecte

4.2.1.6 Cas d'ús 05. Iniciar Joc

Resum de la funcionalitat: Permet a l'usuari iniciar el joc

Paràmetres d'entrada: Cap

Paràmetres de sortida: Cap

Actors: Usuari(Màster)

Precondició: Tots els usuaris estan llestos

Postcondició: S'inicia el joc

Procés normal principal:

1. Els usuaris premen el botó de "Llest" per indicar que estan llestos
2. L'aplicació comprova que tots els usuaris estan llestos
3. L'aplicació inicia el joc

Alternatives de procés

2a. No tots els usuaris estan llestos
    2a1. Tornar al pas 2

4.2.1.7 Cas d'ús 06. Moure Carta

Resum de la funcionalitat: Permet a l'usuari moure una carta

Paràmetres d'entrada: Cap

Paràmetres de sortida: Cap

Actors: Usuari, Usuari(Màster)

Precondició: Cap

Postcondició: Creació de la sala

Procés normal principal:

1. L'usuari prem al damunt d'una carta
2. L'usuari arrastra la carta cap a la seva destinació
3. Es comprova si l'usuari és el propietari de la carta o no ha estat assignada
4. L'aplicació actualitza la posició de la carta per a tots els usuaris

4.2.1.8 Cas d'ús 07. Mostrar Carta

Resum de la funcionalitat: Permet a l'usuari mostrar la seva carta

Paràmetres d'entrada: Cap

Paràmetres de sortida: Cap

Actors: Usuari, Usuari(Màster)

Precondició: Cap

Postcondició: El contingut de la carta es vist pels demés usuaris

Procés normal principal:

1. L'usuari fa doble click[6] a sobre d'una carta
2. L'aplicació mostra el contingut de la carta als demés usuaris

4.2.1.9 Cas d'ús 08. Mesclar

Resum de la funcionalitat: Permet a l'usuari mesclar la baralla de cartes

Paràmetres d'entrada: Cap

Paràmetres de sortida: Cap

Actors: Usuari(Màster)

Precondició: Baralla existent

Postcondició: Mescla de la baralla

[6] Acció de prémer un botó del dispositiu apuntador.

Procés normal principal:

1. L'usuari prem el botó del menú
2. L'aplicació mostra el menú d'opcions al usuari
3. L'usuari prem el botó de mesclar baralla
4. L'aplicació executa les funcions necessàries per mesclar la baralla

4.2.1.10 Cas d'ús 09. Repartir

Resum de la funcionalitat: Permet a l'usuari repartir les cartes

Paràmetres d'entrada: Cap

Paràmetres de sortida: Cap

Actors: Usuari(Màster)

Precondició: Cap

Postcondició: Es reparteix la baralla

Procés normal principal:

1. L'usuari prem el botó del menú
2. L'aplicació mostra el menú d'opcions al usuari
3. L'usuari prem el botó de repartir baralla
4. L'aplicació executa les funcions necessàries per repartir la baralla

5 Disseny

S'expliquen els diferents aspectes del disseny del projecte, inclou una descripció detallada de com s'ha elaborat el projecte, els seus objectius, la forma com s'han aconseguit aquests objectius i el procés d'elaboració.

Com a aplicació per a aquest projecte implementarem les característiques bàsiques del framework, bàsicament aplanarem el camí per a que en un futur aquest serveixi com a plataforma per a múltiples implementacions.

5.1 Arquitectura de l'aplicació

- Pàgina web programada amb HTML5
    - Joc de cartes incrustat a dintre de la pàgina web
        - Interfície gràfica inicial
            - Introduir un Nick de jugador i entrar al joc
            - Triar crear sala o escollir sala
                - Possibilitat de configuració en la creació de sala
                - Possibilitat d'entrar en múltiples sales al escollir sala
            - Es mostra un botó on podem elegir si estem preparats per començar el joc
                - Podem veure si els demés jugadors estan preparats per començar el joc
            - Quan tots estan preparats es mostra un botó en el jugador que ha creat la partida per començar el joc
        - Interfície gràfica Joc
            - Fons de pantalla que simula un tauler
            - Objectes que representaran cartes
                - Possibilitat de moure les cartes
                - Privacitat del contingut de les cartes
                - Opció d'ensenyar les cartes als altres jugadors
            - Menú d'opcions
                - Botó per mesclar les cartes
                - Botó per repartir les cartes

5.2 Disseny interfície gràfica

5.2.1 Entrada Nick

El primer que es mostrarà al iniciar el joc serà el Lobby, la primera pantalla del lobby té que deixar triar un nick al usuari. Aquest es mostrarà quan entre dins una sala per poder ser identificat per les demés persones, també es mostrarà en el moviment de les cartes.

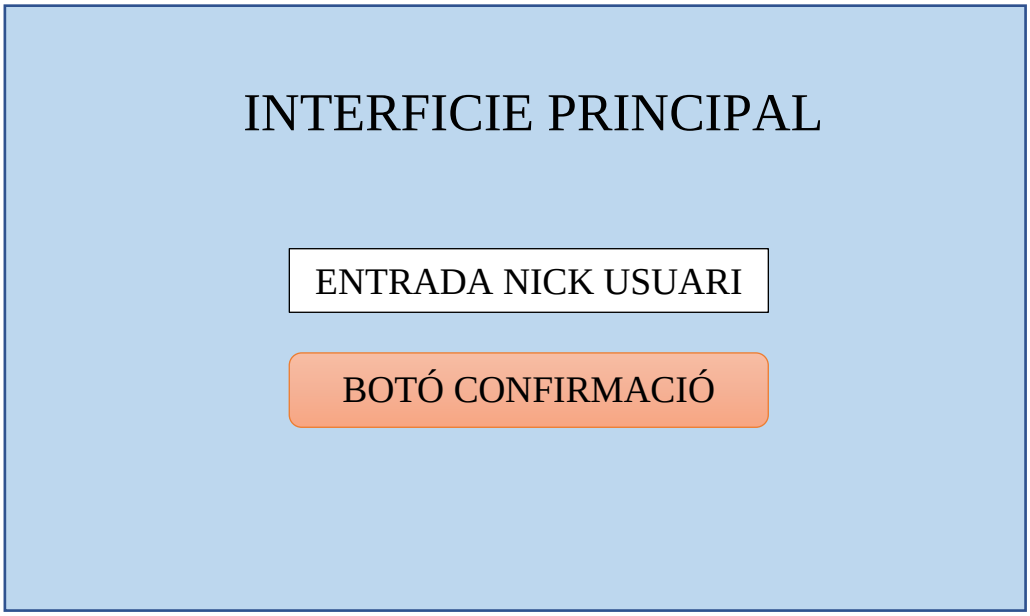

Figura 4. Interficie principal d'entrada del Nick

5.2.2 Seleccionar opció

Aquí el jugador decidirà si vol crear o elegir una sala, normalment el jugador triarà crear una sala si vol ser el game màster o accedirà a la llista de sales si aquest vol elegir una sala ja creada per un altre jugador.

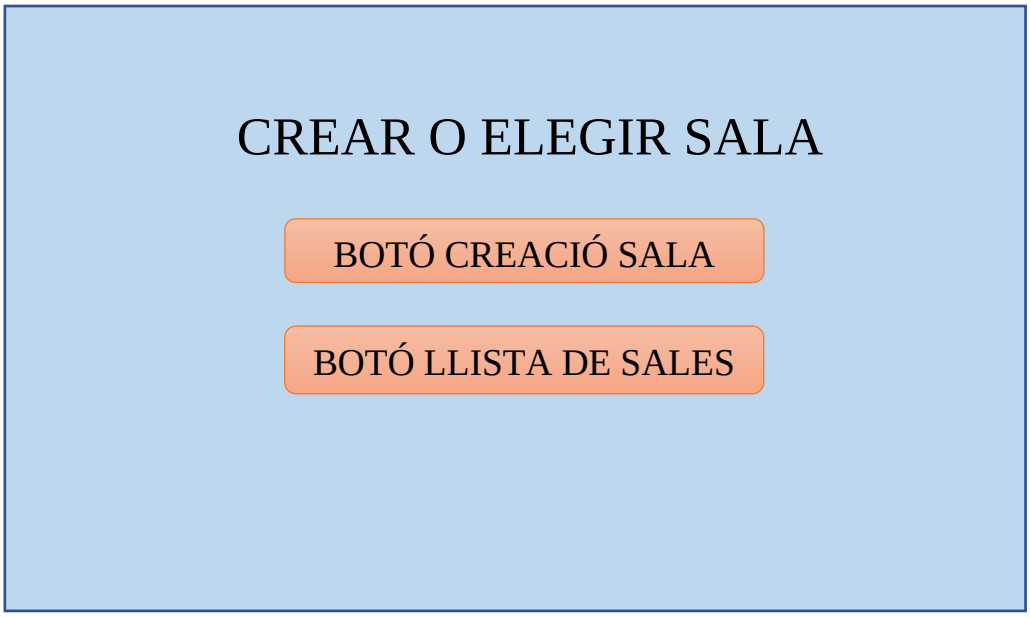

Figura 5. Interfície on ens deixa crear o buscar sales

5.2.3 Disseny de la sala

En el disseny de la sala el game màster podrà elegir entre les diferents opcions que es proposen.

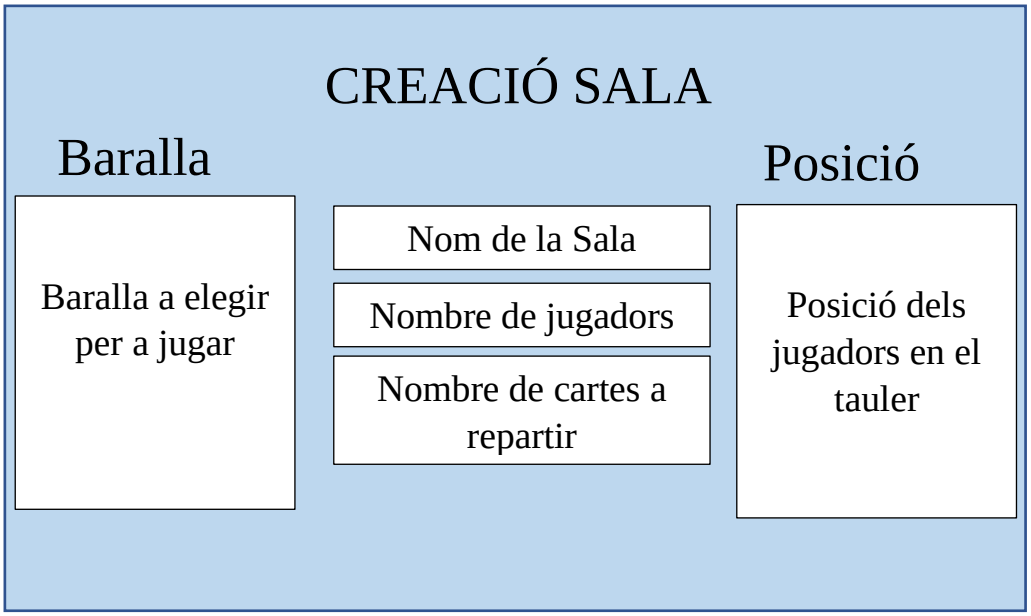

Figura 6. Descripció visual de la configuració de sala

5.2.4 Selecció de Sala

Si l'usuari no ha creat una sala i elegeix buscar sala, aquest podrà triar una de les sales on hi hagin espais buits, en cas de que no hi hagi sales disponibles, hi ha d'haver una opció per a que pugui tornar enrere i crear ell una sala.

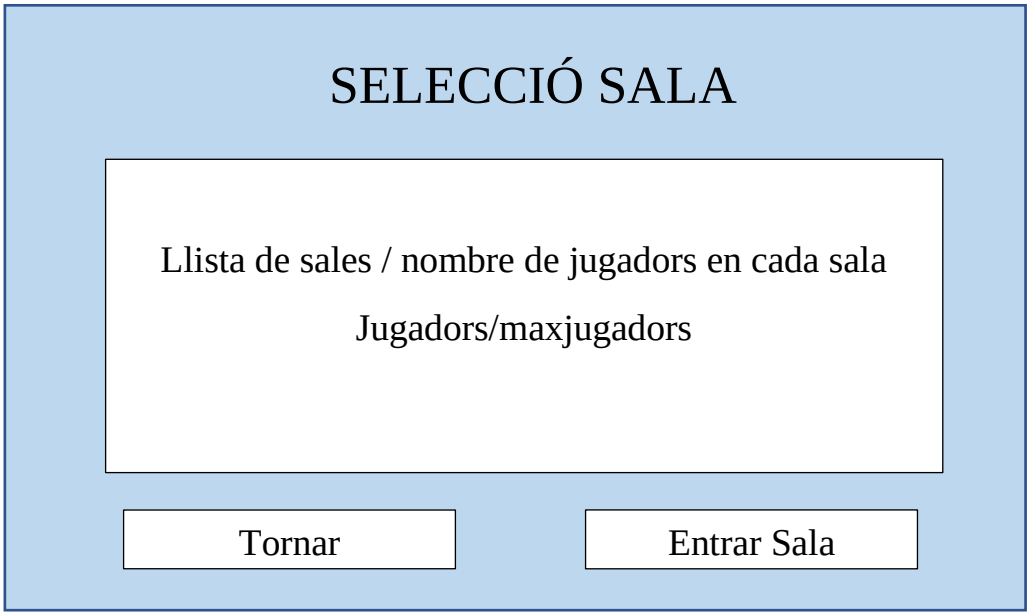

Figura 7. Descripció visual de la selecció de sala

5.2.5 Dintre de la Sala

Un cop dintre de la sala es mostraran tots els jugadors que hi hagi a dins de la sala, aquests podran tenir la opció de tornar si no és la sala que estaven buscant. Un cop estiguin llestos per a jugar hauran de prémer un botó per a confirmar que estan llestos.

Només el game màster podrà prémer el botó jugar per començar el joc un cop estiguin tots llestos.

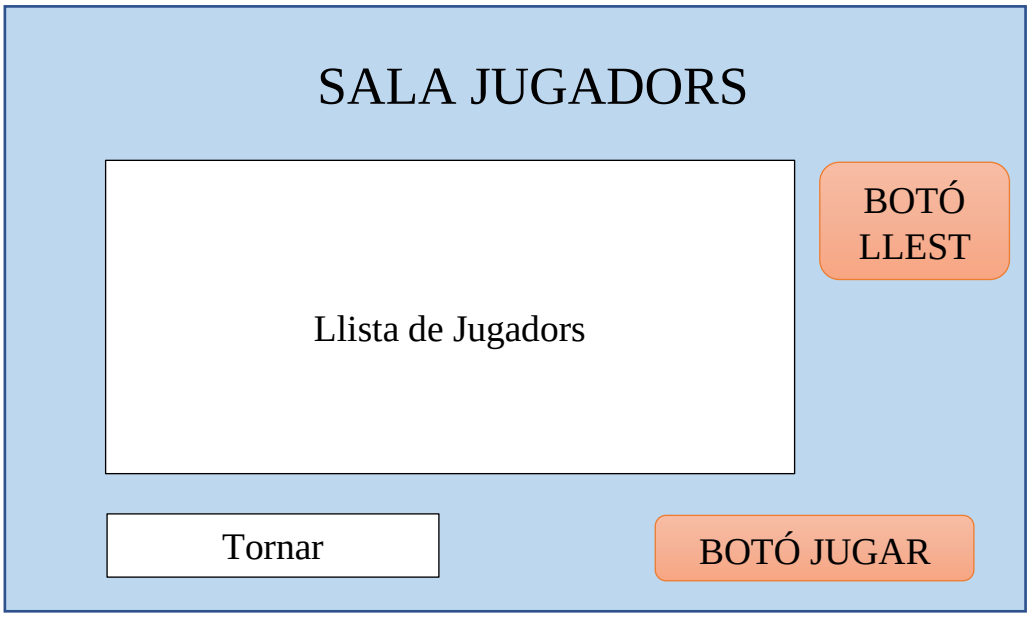

Figura 8. Descripció visual de la sala de jugadors

5.3 Disseny del joc

En el disseny del joc decidirem quines característiques bàsiques volem que siguin implementades per a proporcionar la adaptabilitat necessària en implementacions futures.

En un estudi de les accions principals en els jocs de cartes, hem deduït que les més bàsiques són les següents:

- Moure la carta
- Mostrar la carta
- Agafar cartes
- Mesclar cartes
- Repartir cartes

El moviment de cartes es farà mitjançant el punter del ratolí, aquestes cartes es trobaran a sobre d'un fons de pantalla que simularà una taula o tapet on es solen jugar partides de cartes en la realitat.

Aquest moviment s'implementarà amb un sistema "Drag & Drop" (arrastra i solta), aquest consistirà en fer-hi click damunt de la carta que volem moure i mentre mantenim pressionat el botó del ratolí arrastrar la carta fins a la seva destinació final.

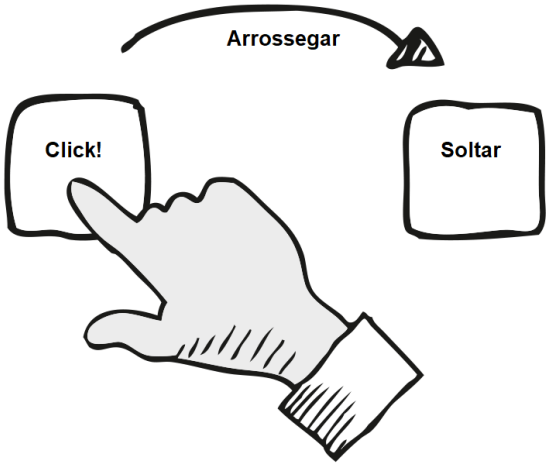

Figura 9. Representació del sistema Drag **&** Drop

Les cartes estaran organitzades en baralles, les baralles es crearan en el centre del tauler, aquestes baralles es podran mesclar i repartir. Les accions indicades s'han de poder fer a partir d'un menú desplegable on es mostraran els respectius botons. Només podran ser realitzades per l'usuari que ha creat la partida.

Els propietaris de les cartes són els únics que podran veure el seu contingut, aquest s'ha de poder mostrar als altres jugadors fent doble click a sobre la carta.

S'han de poder agafar cartes de la baralla de cartes, els usuaris que agafen una carta passaran a ser els propietaris d'aquesta.

Només els usuaris propietaris de la carta podran moure-les i mostrar el seu contingut.

6 Implementació

Per a la implementació de l'aplicació es tindrà en compte el disseny fet anteriorment, d'aquesta forma, es podrà elaborar un projecte amb una directiva clara que permetrà assolir els objectius plantejats.

Com a motor de joc principal s'utilitzarà Unity, aquest conté un entorn de desenvolupament gratuït que ens permetrà crear i desenvolupar el projecte amb totes les característiques que es requereixen.

El llenguatge de programació utilitzat normalment per Unity és **C#**, aquest és un llenguatge de programació desenvolupat i estandarditzat per l'empresa de Microsoft, és un llenguatge modern, basat en objectes i amb seguretat de tipus. Permet als desenvolupadors crear molts tipus d'aplicacions segures i sòlides que s'executen en .NET.

C# també conté el seu propi entorn de desenvolupament, **Microsoft Visual Studio**, aquest té les seves versions gratuïtes però hem optat per la seva versió completa per estudiants proporcionada per la universitat, aquest és completament compatible amb Unity i proporciona opcions fàcils d'implementació del codi.

Unity també disposa d'una llibreria d' "**Assets**" o eines que normalment provenen de fora, aquestes normalment son creades per altres usuaris o fins i tot empreses que es dediquen a crear paquets d'aquestes eines expressament per poder vendre-les dintre de la tenda d'Unity.

En el cas que ens ocupa hem buscat un paquet multijugador que ens permeti crear sales de jugadors on aquests es puguin unir en una partida individual i que ens permeti sincronitzar els moviments dels jugadors de la sala.

El paquet seleccionat finalment ha sigut Photon, aquest ens dona la possibilitat d'implementar les característiques que tenim en ment per a elaborar el joc multijugador i que hem requerit anteriorment.

Figura 10. Logotip d'Unity

Figura 11. Logotip de Photon

Figura 12. Logotip del llenguatge C#

Figura 13. Logotip de Microsoft Visual Studio

6.1 Configuració de l'entorn de desenvolupament

6.1.1 Configuració Unity

Un cop descarregat i instal·lat Unity només haurem de procedir a utilitzar la seva interfície gràfica per obtenir el complement de Photon i configurar-lo.

Ens registrem a Unity, accedim a la Unity Asset Store i descarreguem el complement en la seva versió gratuïta.

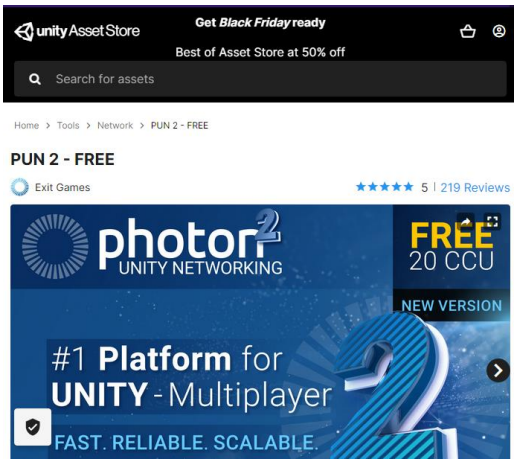

Figura 14. Photon en l'Asset Store

Un cop importada l'eina ens fixem en que aquesta inclou diverses Demos[7], farem ús d'aquestes per tenir un exemple i aprendre com programar el nostre joc multijugador.

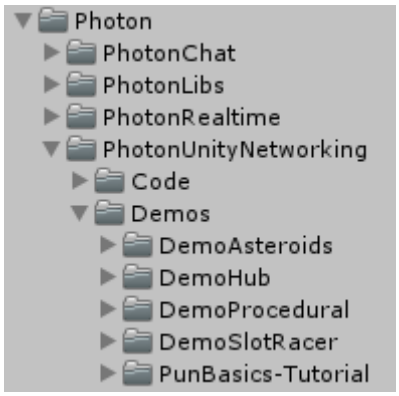

Figura 15. "Demos" de Photon

6.1.2 Configuració Photon

Per a poder implementar la sincronització el primer que hem de fer és registrar-nos a la pàgina web de Photon, un cop registrats accedim a la Dashboard[8] que ens és proporcionada i creem una nova aplicació utilitzant la seva interfície gràfica.

[7] Programa informàtic de demostració.
[8] Quadre de comandaments.

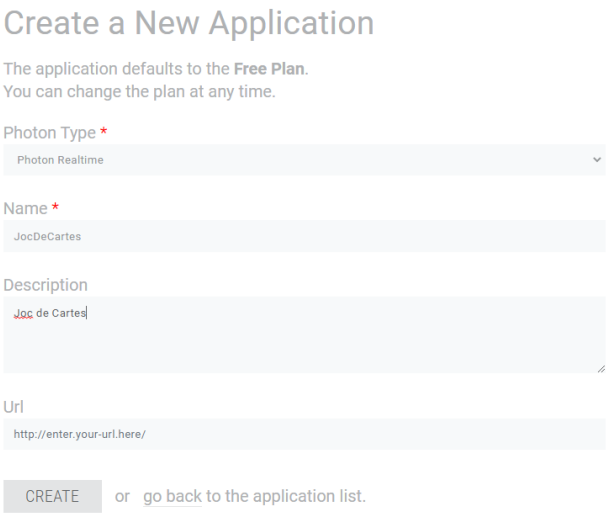

Figura 16. Creació de l'aplicació en la pàgina web de Photon

Un cop creada l'aplicació tornem a la Dashboard i notem que ens ha aparegut un quadre on se'ns mostra les estadístiques de la nostra aplicació i una AppID, aquest AppID l'enregistrarem al nostre paquet de Photon i permetrà a l'aplicatiu identificar per quin servidor es te que realitzar la connexió.

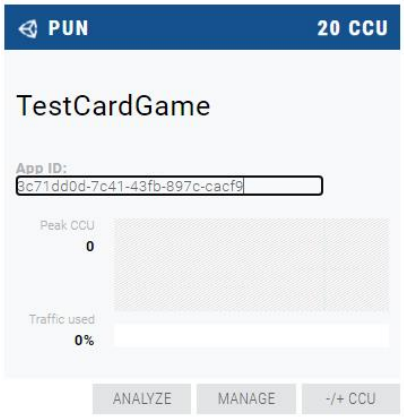

Figura 17. Dashboard en la pàgina web

De moment veiem que l'aplicació de la pàgina web ens mostra que tenim 20 CCU, significa que com a màxim podem tenir 20 jugadors connectats utilitzant la nostra aplicació en un moment puntual.

Aquest element és el que ens permetrà en un futur analitzar la quantitat de jugadors que hi ha jugant en la nostra aplicació. Un cop realitzat un anàlisi sobre la utilització dels servidors podrem decidir si ampliar la quantitat d'usuaris que poden utilitzar la nostra aplicació.

D'aquesta forma complim amb els objectius d'escalabilitat de l'aplicació, ja que aquesta pot augmentar de forma significativa els seus usuaris.

6.2 Creació del Lobby

Tenint en compte el disseny de la interfície gràfica feta anteriorment es passa a crear-la utilitzant Unity com a eina gràfica principal.

6.2.1 Implementació entrada Nick

Per a l'entrada del Nick s'utilitza un **InputField**[9], aquest **GameObject** d'Unity permet la entrada per teclat d'un text el qual es pot obtenir posteriorment si hi introduïm una referència al nostre script.

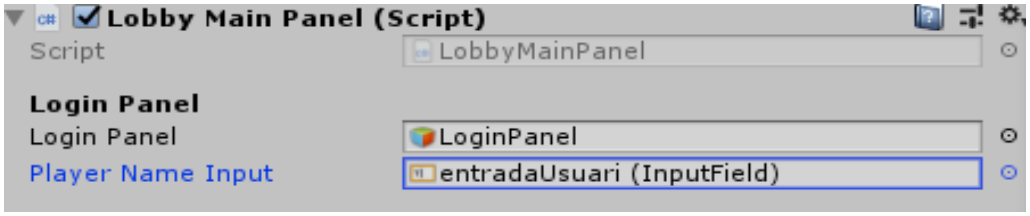

Figura 18. Referència a entradaUsuari dintre l'script

```
public void OnLoginButtonClicked()
        {
            string playerName = PlayerNameInput.text;
```

Codi 1. Extracció del text introduït a partir de la referència

Per al botó d'entrada s'utilitza un **Button**[10], el qual també és un **GameObject**[11] d'Unity, aquest permet cridar una funció d'un script referenciat anteriorment quan es prem.

Figura 19. Referència a la funció que crida el botó al ser premut

En aquest cas s'utilitza l'script **LobbyMainPanel** i la funció **OnLoginButtonClicked** la qual s'encarrega de guardar el nom del jugador escrit dintre de l'entrada de text a dintre de la variable ja implementada **PhotonNetwork.LocalPlayer.NickName**, aquesta s'encarrega de guardar el Nick del jugador durant el temps que aquest estigui connectat al servidor.

L'script també s'encarrega de cridar la funció **PhotonNetwork.ConnectUsingSettings**, aquesta s'encarrega de connectar-se als servidors de Photon utilitzant la configuració establerta prèviament.

L'script implementa la classe **MonoBehaviourPunCallbacks**, un cop la connexió ha sigut establerta es crida la funció **OnConnectedToMaster**, la qual s'encarrega de posar activa la següent pantalla de selecció d'opcions.

[9] Entrada de text
[10] Botó.
[11] Objectes fonamentals en Unity, contenen components que implementen la funcionalitat.

```
public override void OnConnectedToMaster()
        {
            this.SetActivePanel(SelectionPanel.name);
        }
```

Codi 2. Funció OnConnectedToMaster

Figura 20. Resultat de la implementació de l'entrada de Nick

6.2.2 Implementació Selecciona opcions

Aquesta interfície ens deixa triar si volem crear una sala o bé accedir a la llista de sales ja creades, els botons provenen dels **GameObject** d'Unity ja implementats els quals contenen la funció **OnClick**, aquesta ens permet especificar el que volem fer quan es prem el botó, en el cas de creació de sala, desactivem el panel actual i activem el següent.

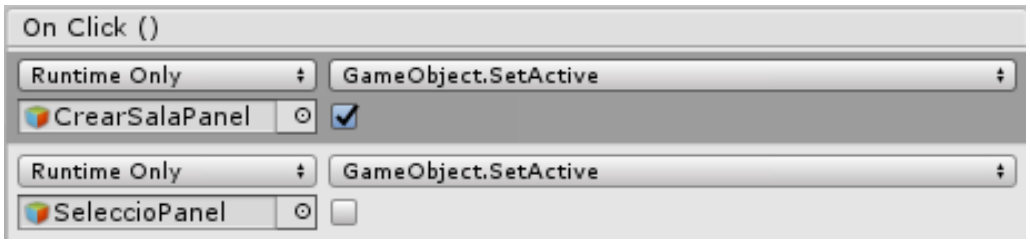

Figura 21. S'activa un panell i a la vegada es desactiva l'altre

En el cas de Llista de sales, cridem a la funció **OnRoomListButtonClicked**, funció que s'encarrega d'obtenir i mostrar la llista de sales disponibles.

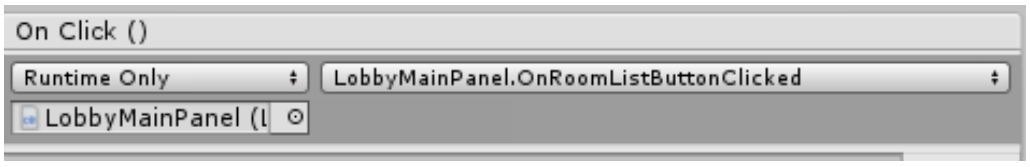

Figura 22. Referència del botó a la funció per buscar sala

Figura 23. Implementació de la selecció d'opcions

6.2.3 Implementació Creació de la Sala

6.2.3.1 Selecció de la baralla

En la creació de la sala hi ha el requeriment de poder triar una baralla, es creen dos botons per recórrer les baralles i un sprite[12] que serà l'encarregat de mostrar una de les cartes principals, així podrem identificar de quina baralla estem parlant.

Es crea una variable global dintre de **LobbyMainPanel** anomenada **barallaactual** i s'inicialitza a 0, s'utilitzarà posteriorment per identificar la baralla seleccionada.

```
private int barallaactual = 0;
```

Als dos botons creats anteriorment per triar la baralla els assignarem les seves respectives funcions per canviar el sprite i actualitzar la variable **barallaactual**.

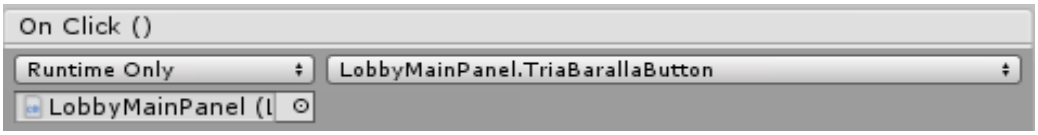

Figura 24. Referencia d'un dels botons a la funció per triar baralla

Figura 25. Visió de la selecció de baralla

[12] Objecte gràfic 2D.

6.2.3.2 Característiques de la Sala

Es deixarà als jugadors triar el nom de la sala que es mostrarà en la llista de sales, d'aquesta forma facilitem que els jugadors puguin identificar partides la sala que han creat els seus amics i poder jugar amb ells.

Es decideix que aquests també puguin triar el nombre màxim de jugadors de la sala i el nombre de cartes a repartir; es creen els **InputField** necessaris per a que els usuaris puguin introduir aquestes dades i les seves referencies corresponents en el codi.

```
public InputField NumCartesInput;
public InputField RoomNameInputField;
public InputField MaxPlayersInputField;
```

Quan es decideix prémer el botó de **Crear Sala** es crida a la funció **OnCreateRoomButtonClicked** que s'encarrega de comprovar si s'ha introduït un nom de sala, si no s'ha introduït se'n crea un, aquesta funció també s'encarrega de limitar el nombre de jugadors que s'han introduït entre 2 i 8 depenent del valor que s'ha introduït.

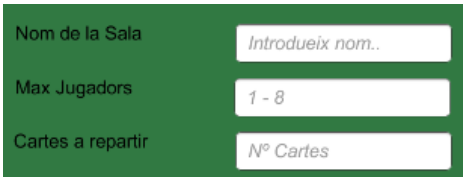

Figura 26. Visió de les entrades personalitzades

6.2.3.3 Posició dels Jugadors

Per a que l'usuari creador de la sala pugui decidir on es situaran els jugadors depenent de l'ordre d'entrada a la sala es decideix utilitzar els objectes d'unity anomenats **Toggle**, aquests objectes permeten canviar el seu contingut en el moment que es premen, normalment s'utilitzen com a tick[13] , però en aquest cas els modificarem per a que continguin un nombre establert.

Per a poder ordenar tots els **Toggle**[14], utilitzarem una característica d'unity anomenada **Grid Layout Group**[15], la qual ens permetrà ordenar-los sense tenir que fer-ho manualment. Un sol GameObject buit contindrà l'script de grid layout group i els Toggle seran descendents d'aquest. Posteriorment decidirem la distància de separació entre aquests.

Figura 27. Visió dels toggle creats en un grid layout group

Inicialitzarem el tauler de toggles cada cop que entrem a crear sala o cada cop que canviem el nombre màxim de jugadors de la partida. Per aconseguir-ho crearem la funció

[13] Marcador
[14] Alternador, pot estar activat o desactivat
[15] Grup de disseny de quadrícula.

PosaToggleValors. Aquesta canvia el text de cada un dels Toggle pel valor corresponent, si s'ha arribat al màxim de jugadors simplement introdueix un text buit.

Tal i com va col·locant els valors a les caselles aquesta funció introdueix els valors dintre d'una llista anomenada **togglenumbers** per si posteriorment es decideix reordenar aquests valors.

Figura 28. Visió de l'entrada del nombre de jugadors i com s'ordenaran aquests

Per a que sigui possible reordenar els jugadors depenent de les preferències de l'usuari es crea la funció **OnToggleCanvi**, quan siguin pressionats, tots els toggle cridaran aquesta funció que s'encarregarà d'assignar-los el seu valor corresponent.

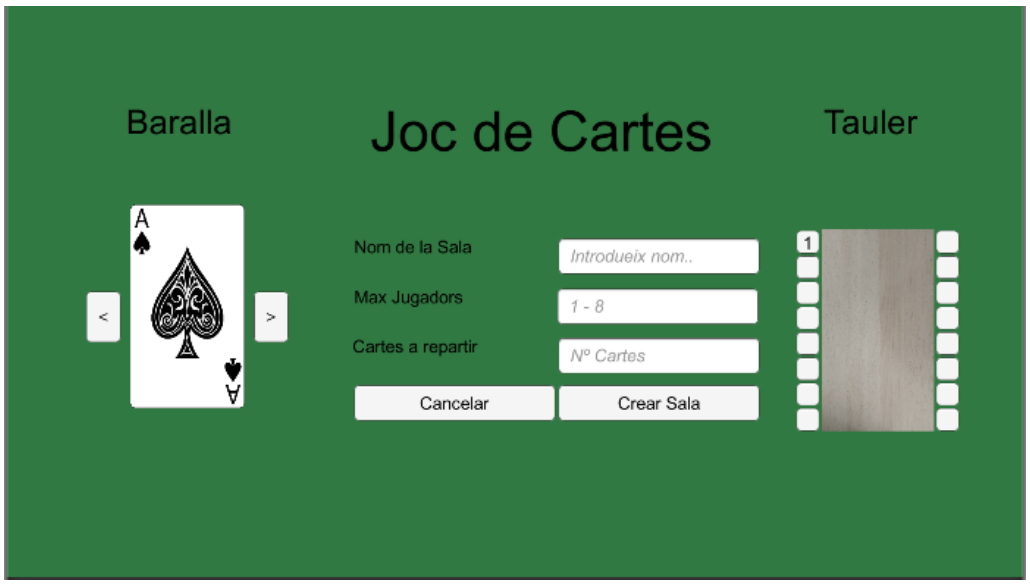

Figura 29. Implementació final de la configuració de sala

6.2.4 Implementació Selecció de la Sala

La funció **OnRoomListButtonClicked** s'assegura que el jugador entri al Lobby i activa el panell on es mostra la llista de sales que hi ha actives. Per a entrar al Lobby s'utilitza la funció ja implementada de Photon anomenada **JoinLobby**.

```
public void OnRoomListButtonClicked()
{
    if (!PhotonNetwork.InLobby)
    {
        PhotonNetwork.JoinLobby();
    }

    SetActivePanel(RoomListPanel.name);
}
```

Codi 3. Funció OnRoomListButtonClicked

Figura 30. Implementació selecció de sala

6.2.5 Implementació Dintre de la Sala

Dintre de la sala simplement se'ns permetrà decidir si estem preparats i se'ns mostrarà els altres jugadors, fem la implementació agafant com a exemple la "Demo" que incorpora Photon en el seu Asset.

Aquesta ja porta definida un prefab[16] anomenat **PlayerListEntry** que conté el nom del jugador i un botó per decidir si estàs preparat, l'utilitzarem per a la nostra implementació.

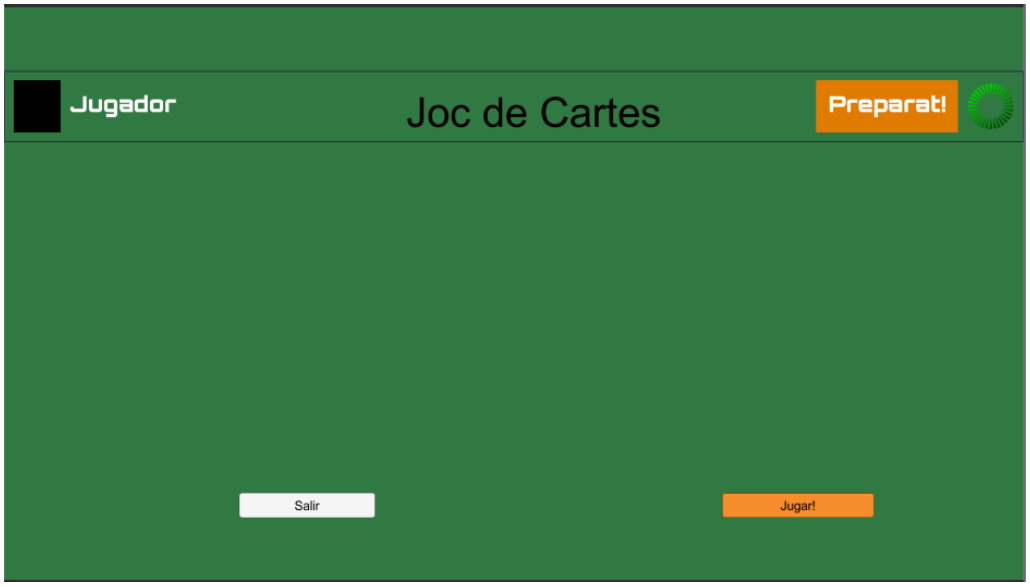

Figura 31. Implementació dintre de la sala

[16] Objectes reutilitzables i ja creats prèviament amb una sèrie de característiques.

6.3 Implementació del Joc

6.3.1 Punt de partida

Agafem com a punt de partida un sprite que simularà una carta el qual hauríem de poder moure utilitzant el ratolí.

Creem un **Sprite** a dintre d'Unity, aquest serà el component que simularà la nostra carta, en aquest cas només es necessari canviar la imatge de l'sprite.

Figura 32. Referència a la imatge del sprite

Per a implementar la característica de moviment amb el ratolí agafem com a exemple un script que ens permeti fer Drag & Drop, aquest conté un codi simple amb crides a OnMouseDown, OnMouseUp i la funció **Update**[17] per a actualitzar la posició.

Comencem declarant la variable **dragging**[18], la qual indicarà a la funció **Update** si te que actualitzar el moviment de l'objecte o no i **distància**, en la qual guardarem la distància de l'objecte a la càmera principal.

La variable **startDist** serà utilitzada per a que la nova posició de l'objecte no sigui exactament la posició del ratolí i que aquesta sigui relativa a la posició des d'on agafem l'objecte.

```
private bool dragging = false;
private float distance;
private Vector3 startDist;
```

Codi 4. Variables elementals

Quan es prem el ratolí es crida a la funció **OnMouseDown**, aquesta guarda la distància **distance** que hi ha entre l'objecte clicat i la càmera principal (transform es refereix a la posició, rotació i escala d'un objecte).

Es posa el booleà **dragging** a verdader per a que es comenci a actualitzar la posició dintre de la funció **Update** explicada mes endavant.

```
void OnMouseDown()
    {
        distance = Vector3.Distance(transform.position,
Camera.main.transform.position);
        dragging = true;
        Ray ray = Camera.main.ScreenPointToRay(Input.mousePosition);
        Vector3 rayPoint = ray.GetPoint(distance);
        startDist = transform.position - rayPoint;
    }
```

Codi 5. Funció de referència OnMouseDown

[17] Funció que actualitza els valors en l'objecte, s'executa un cop per fotograma.
[18] Efecte d'arrossegar.

La funció **OnMouseUp** detectarà quan s'ha deixat de prémer el ratolí i simplement canviarà el booleà **dragging** a fals per a que deixi d'actualitzar-se la posició.

```
void OnMouseUp()
    {
        dragging = false;
    }
```

Codi 6. Funció de referència OnMouseUp

Es genera un vector (Ray) des de la posició on es troba la càmera i en direcció a la posició on es troba el ratolí.

Obtenim el punt de destí utilitzant la funció integrada **ray.GetPoint**, utilitzada per obtenir punts en la direcció del vector introduint nosaltres la distància a la que es troba el punt.

El punt final obtingut **rayPoint** és el destí on anirà el nostre objecte, en aquest cas, la carta.

La variable **startDist** es refereix a la posició calculada anteriorment en la funció **OnMouseDown** de la mateixa forma que hem explicat el càlcul de **rayPoint**, aquesta és una correcció del punt on s'ha de moure la carta, per simular que l'estem agafant amb el ratolí.

Figura 33. Sense correcció

Figura 34. Amb correcció

```
void Update()
    {
        if (dragging)
        {
            Ray ray = Camera.main.ScreenPointToRay(Input.mousePosition);
            Vector3 rayPoint = ray.GetPoint(distance);
            transform.position = rayPoint + startDist;
        }
    }
```

Codi 7. Funció de referència Update

Finalment, introduïm la referència al script a dintre de l'objecte que hem creat

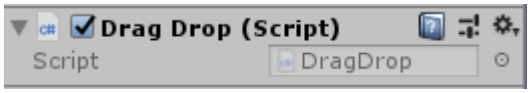

Figura 35. Script referenciat a dintre de l'objecte

6.3.2 Implementació de la Sincronització

6.3.2.1 Sincronització del moviment de les cartes

La documentació de Photon ens diu que per sincronitzar objectes a través de la xarxa és necessari assignar un component **PhotonView** per a poder sincronitzar la posició, rotació i altres valors que tinguin en comú amb els duplicats dels altres jugadors.

Un cop instal·lats els paquets necessaris ens apareixerà com a paquet disponible dintre de l'apartat **Photon Networking**, afegim el component mencionat PhotonView.

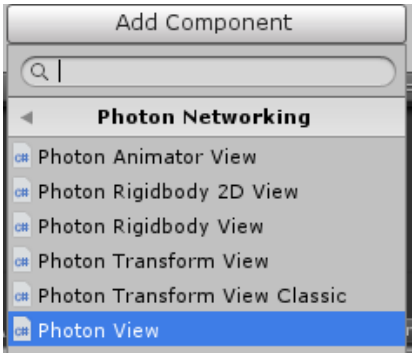

Figura 36. Llistat de components de Photon que es poden afegir

Comencem a editar l'script **DragDrop** el qual hem creat anteriorment per a que funcioni la sincronització entre els jugadors que hi ha a la sala.

Primer de tot, l'script te que canviar de la implementació **MonoBehaviour** per defecte d'unity cap a **MonoBehaviourPun**, que és la classe de Photon de la qual han d'heretar els scripts[19] dels objectes que s'han de sincronitzar.

Per a que els jugadors puguin veure el moviment de les cartes que fan els altres jugadors és necessària la sincronització de la posició dels objectes que estaran en moviment, en aquest cas les cartes.

Ens es proporcionada una funció en la llibreria anomenada **OnPhotonSerializeView**, incloem les variables a sincronitzar **transform.position** i **transform.rotation**, les quals contenen la posició i la rotació a sincronitzar de l'objecte.

La funció es cridada varies vegades per segon, ens permet triar les dades que volem que es sincronitzen regularment, el client que és propietari de l'objecte estarà en mode ‘‘escriure’’, mentre que els altres clients estaran en mode ‘‘lectura’’.

```
public void OnPhotonSerializeView(PhotonStream stream, PhotonMessageInfo info)
{
    if (stream.IsWriting)
    {
        //We own this player: send the others our data
        stream.SendNext(transform.position);
        stream.SendNext(transform.rotation);
    }
    else
    {
```

[19] Arxiu que conté una seqüència de comandaments.

```
        //Network player, receive data
        latestPos = (Vector3)stream.ReceiveNext();
        latestRot = (Quaternion)stream.ReceiveNext();
    }
}
```

Codi 8. Sincronització de la posició i rotació de les cartes

Per a que els demés clients sincronitzin el moviment de les cartes és necessari que actualitzin els valors en la funció Update, utilitzem el booleà **photonView.IsMine** per a saber quan la carta no està en la nostra propietat i actualitzar la posició.

Mitjançant la funció **Vector3.Lerp** provoquem que el moviment de les cartes sigui "suau" ja que la posició que ens arriba per part dels altres jugadors no es constant.

Un augment dels enviaments per part de la funció **OnPhotonSerializeView** pot provocar problemes de rendiment.

La rotació no la utilitzarem de moment per a la versió actual del projecte però possiblement s'utilitzi en versions posteriors.

```
private void Update()
{
    if (!photonView.IsMine)
    {
        transform.position = Vector3.Lerp(transform.position, latestPos,
Time.deltaTime * 5);

        transform.rotation = Quaternion.RotateTowards(transform.rotation,
latestRot, Time.deltaTime * 50);

    }
}
```

Codi 9. Actualització de la posició i rotació en els jugadors no propietaris

6.3.2.2 Sincronització de variables per al funcionament del joc

Per a canvis de variables que no sigui necessària una sincronització continuada, se'ns ofereix la possibilitat d'enviar RPC(Remote Procedure Call) o crides de procediments remots, s'han creat crides per a les següents variables que seran sincronitzades únicament en el moment que sigui necessari:

- Canvi de nom
- Canvi de sprite
- Canvi de transparència
- Canvi de Propietari
- Canvi de Posició
- Canvi de la variable per saber si la carta esta assignada
- Carta en moviment (CR_Running)

La funció per a la crida de canvi de variable del nom serà utilitzada per les funcions OnMouseDown i OnMouseUp, ja que ens interessa que es mostri el nom del jugador que esta movent la carta en el text superior ancorat a aquesta.

Algunes funcions seran utilitzades per la classe de Baraja, ja que aquesta necessita de funcions especifiques per a poder mesclar i repartir les cartes.

```
[PunRPC]
public void CanviaSprite(int num)
{
    spriteRenderer.sprite = DataManager.instance.cartaSprites[num];
}

[PunRPC]
public void CanviaTransparencia(float transparencia)
{
    spriteRenderer.color = new Color(1f, 1f, 1f, transparencia);
}

[PunRPC]
public void CanviaPropietari(Player player)
{
    photonView.TransferOwnership(player);
}
```

Codi 10. Exemple de funcions RPC

```
photonView.RPC("CanviaSprite", RpcTarget.Others, numcarta);
```

Codi 11. Exemple de crida a funció RPC

6.3.3 Creació de la Baralla

Primer de tot necessitem les cartes de la baralla, obtenim les baralles de repositoris online on ja es troben en format **.png** que és ideal per a la seva utilització en unity

Figura 37. Visualització de la col·lecció de cartes en format .png

Es crea una instància en la qual emmagatzemarem la informació que necessitem per al joc anomenada **DataManager**, Dintre d'aquesta instància emmagatzemarem tots els sprites referents a les cartes en una variable pública anomenada **cartaSprites**.

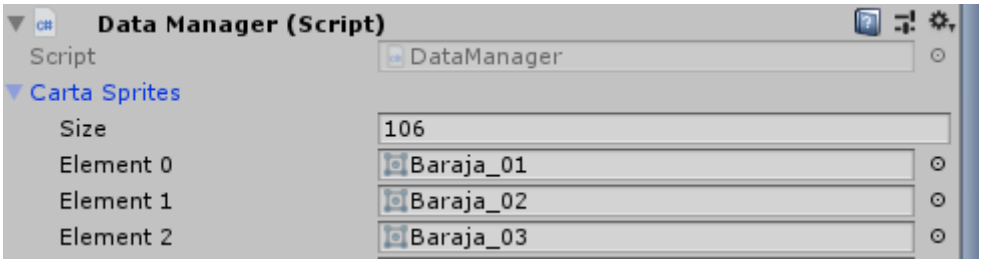

Figura 38. Array de referències als sprites de les cartes

També crearem un Array d'Arrays constant on hi definirem les cartes dintre de **cartaSprites** que formen part de cada baralla.

```
public readonly int[][] defbaralles = new int[2][]
{
    new int [] { 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, . . . }
    new int [] { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, . . . }
};
```

Codi 12. Array que defineix quines cartes formen part de la baralla

Anteriorment hem decidit que la baralla triada es guardarà dintre de la variable **barallaactual** aquesta s'encarrega d'identificar quin Array dintre de **defbaralles** s'utilitza per a crear la baralla.

6.3.4 Funcions de mesclar i repartir

Per a millorar la jugabilitat dels jugadors i que no tinguin que fer tants de moviments amb el ratolí s'han implementat les funcionalitats de mesclar i repartir la baralla, primer de tot creem la classe **Baraja**, que ens ajudarà a implementar aquestes funcions més fàcilment.

6.3.4.1 Inicialització de la Baralla

Aquesta classe contindrà una **Llista** de cartes, com que l'script que conté les característiques i funcions de les cartes és el de la classe **DragDrop**, podríem dir que específicament serà una **Llista** de tipus **DragDrop**.

```
private List<DragDrop> baraja = new List<DragDrop>();
```

Codi 13. Declaració de baraja**,** la llista que conté les cartes**,** dintre de la classe Baraja

Per inicialitzar la baralla, seran necessaris els següents elements:

- Array d'enters que ens digui quines cartes formaran part de la baralla
- Posició on es trobarà la baralla
- Rotació de la baralla
- ID de la baralla per identificar-la en un futur

Crearem una funció la qual li puguem passar tots aquests paràmetres i inicialitzar-los

```
public void InitializeBaraja(int[] cartes, Vector3 position, Quaternion rotation,
int idbaraja)
```

Codi 14. Funció que inicialitza la baralla

Per a inicialitzar les cartes se'ns requereix que utilitzem la funció inclosa en el paquet anomenada **PhotonNetwork.Instantiate**, aquesta ens permetrà crear objectes en la xarxa, normalment s'utilitzaria la funció d'Unity **Object.Instantiate**, però aquesta només ens crearia objectes en l'àmbit local del usuari.

PhotonNetwork.Instantiate obté el prefab que se li passa per paràmetre i li passa les dades necessàries que li hem introduït, aquesta com a paràmetres principals només ens accepta la posició i la rotació dels objectes a instanciar.

```
public static GameObject Instantiate(string prefabName, Vector3 position,
Quaternion rotation, byte group = 0, object[] data = null)
```

Codi 15. Funció que ens ofereix Photon per instanciar objectes en la xarxa

Per a passar-li la resta de dades necessàries hem de crear un array d'**objectes** que posteriorment obtindrà el nostre script DragDrop al ser instanciat.

Com a dades passem l'sprite que la carta hauria de tenir al ser destapada, l'sprite que la carta tindrà al estar oculta i l'ID de la baralla que l'ha creat.

```
object[] data = new object[3];
data[0] = i; //Sprite de la carta
data[1] = cartes.Last(); //Sprite ocult de la carta
data[2] = idbaraja; //Baralla a la que pertany la carta
```

Codi 16. Variables per a la creació de cartes

6.3.4.2 Mesclar la Baralla

Per a mesclar la baralla el primer que necessitem és un algoritme de mescla que ens doni resultats diferents en cada mescla. Hem decidit basar el nostre algoritme en l'algoritme **Fisher-Yates**, aquest és el que s'utilitza típicament per a barallar en els jocs d'atzar.

Aquest algoritme és un algoritme de permutacions que tècnicament encaixa en la categoria dels algoritmes d'ordenament, encara que, en aquest cas, l'objectiu perseguit és l'oposat, desordenar els elements que conté.

Hem decidit crear una funció que anomenarem **Shuffle**(barallar) , el pseudocodi de l'algoritme que hem trobat es diu algoritme de Durstenfeld, aquest és una variant de l'algoritme de Fisher-Yates per optimitzar memòria.

Passem a implementar l'algoritme en la nostra funció:

```
public static System.Random rng = new System.Random();

private void Shuffle<T>(IList<T> list)
{
    for (int k = list.Count - 1; k > 1; k--)
    {
        int az = rng.Next(k);
        T tmp = list[az];
        list[az] = list[k];
        list[k] = tmp;
    }
}
```

Codi 17. Implementació de l'algoritme Durstenfeld

On la funció **Next(valor)** de la classe implementada **Random** ens retorna un valor positiu i com a valor màxim el valor que li passem per paràmetre.

Tenint ja la funció que ens mesclarà la baralla creada, creem la funció que implementarà totes les funcions necessàries per mesclar la baralla, **MesclaBaraja()**.

En aquesta funció necessitem que es canviï la **coordenada z** de la carta, aquesta és l'encarregada de dir-nos quina carta estarà mes amunt i quina estarà mes avall.

També necessitem que les cartes tornen al punt d'origen de la baralla i que aquestes estiguin ocultes de nou (sinó no serviria de res mesclar-les).

Finalment hem d'encarregar-nos que la transparència que hagi pogut tenir la carta torni a ser opaca i canviar les variables **CartaOculta** i **CartaAssignada**.

Utilitzant les funcions que hem implementat a dintre de l'algoritme **DragDrop** i la que hem creat anteriorment per mesclar la baralla, acabem d'implementar la funció.

```
public void MesclaBaraja()
{
    int i = 1;

    Shuffle(baraja);

    foreach (DragDrop carta in baraja)
    {
        carta.photonView.RPC("CanviaPos", RpcTarget.All, (new Vector3(startpos.x,
startpos.y, (float)(-0.01 * i))));
        carta.photonView.RPC("CanviaSprite", RpcTarget.All, spriteocult);
        carta.photonView.RPC("CanviaTransparencia", RpcTarget.All, 1f);
        carta.photonView.RPC("CartaOculta", RpcTarget.All, true);
        carta.photonView.RPC("CartaAssignada", RpcTarget.All, false);
        i++;
    }

}
```

Codi 18. Implementació de la funció per mesclar la baralla

6.3.4.3 Repartir la Baralla

Per repartir la baralla el primer que necessitem seran les posicions on deurien anar les cartes en cada cas, també necessitarem del nombre de cartes a repartir en cada posició. Com a posicions on deurien anar les cartes podríem obtenir per paràmetre els punts com a variable **Vector3** o extreure aquestes posicions directament des d'uns **GameObject**s, com a mes simple optem per la segona opció.

Recorrem la llista de jugadors i situem les cartes en la posició que pertoca amb una lleugera diferència per a que els usuaris no tinguin que moure les seves cartes per veure quina hi ha baix d'un altra. Utilitzarem la variable **z** (que tenim com a comptador) per aplicar aquesta distància.

Com que les cartes que volem repartir son les que es troben a sobre del tot agafarem la que estigui mes a dalt i anirem baixant el nombre, en aquest cas la de mes a dalt serà **baraja.Count-1** ja que l'array comença en 0.

Utilitzant les funcions creades a dintre de **DragDrop** per controlar les nostres cartes implementem la funció que les repartirà.

```
public void RepartirBaraja(List<GameObject> spawnpoints, int numcartes)
{
    int i = baraja.Count-1; //Variable que diu quina carta es reparteix
    int z = 0; //Variable utilitzada per repartir el numcartes que toca
    int k = 0; //Variable que diu a quin spawnpoint es reparteix la carta, l'array
de jugador i de spawnpoint coincideix

    foreach (var player in PhotonNetwork.PlayerList)
    {
        Debug.Log(player.NickName);
        while (z < numcartes)
        {
            baraja[i].photonView.RPC("CanviaPropietari", RpcTarget.All, player);
            baraja[i].photonView.RPC("CanviaPos", RpcTarget.All, new
Vector3(spawnpoints[k].transform.position.x + z,
spawnpoints[k].transform.position.y, (float)(-0.01 * i)));
            i--;
            z++;
        }
        k++;
        z = 0;
    }

}
```

Codi 19. Implementació de la funció per repartir les cartes

6.3.5 Privadesa de les Cartes

Un joc de cartes on tots els jugadors puguin veure les cartes dels demés jugadors no seria un joc de cartes just, com a solució a aquest problema s'ha decidit implementar un sistema per a que els jugadors no puguin veure ni moure les cartes dels altres jugadors, i que, en cas de que sigui necessari, els mateixos jugadors si puguin mostrar les seves cartes als altres jugadors.

Primer de tot hem decidit crear un booleà dintre de l'script **DragDrop** que es el que s'encarrega de controlar les cartes, anomenarem a aquest booleà **cartaAssignada** i per defecte el seu valor serà fals, ja que la carta en un principi no està assignada a ningun jugador.

```
private bool cartaAssignada = false;
```

Codi 20. Variable que ens diu si la carta està assignada

Dintre de la funció **OnMouseDown** si la carta no ha estat assignada a ningú i ningú la esta movent demanarem ser els propietaris de la carta.

```
if (!cartaAssignada && !CR_Running)
{
    photonView.RequestOwnership();
    . . .
}
```

Codi 21. Comprovació d'assignació de la carta i posterior requisit de propietari

Un cop s'ha transferit la propietat de la carta, cridem a la funció **CartaPrivada** dintre de la funció **OnOwnershipTransfered**, comprovem si realment volem la opció d'assignar cartes

amb un booleà **assignarCarta** que de moment canviarem manualment. En un futur es podria afegir a les opcions de joc per crear la sala.

```
If (assignarCarta) CartaPrivada();
```

Codi 22. Crida realitzada si volem assignar cartes

6.3.6 Mostrar la teva carta

S'ha implementat la possibilitat de poder mostrar la carta als altres jugadors simplement fent un doble click, per aconseguir-ho primer es compta el temps entre les crides a la funció **OnMouseUp**, si la diferència entre crides es de menys de 0,3 segons significa que és un doble click.

```
void OnMouseUp()
{
    if ((Time.time - doubleClickStart) < 0.3f)
    {
        this.OnDoubleClick();
        doubleClickStart = -1;
    }
    else
    {
        doubleClickStart = Time.time;
    }
```

Codi 23. Control dels dobles click

La funció que s'executa en cas de que hi hagi un doble click **OnDoubleClick** s'encarrega de cridar les funcions en forma d'RPC que ja hem creat per a que els canvis especificats tinguin lloc.

En el cas donat de que la carta estigui oculta i s'ha realitzat un doble click, comprova si l'assignació de cartes està activada, si ho està simplement treu la transparència de la carta i mostra el contingut de la carta mitjançant la funció RPC **CanviaSprite**.

```
if (oculta) //si esta oculta es canvia a no oculta, sino a normal
{
    if (assignarCarta)
    {
        spriteRenderer.color = new Color(1f, 1f, 1f, 1f); // 0% transparent
        photonView.RPC("CanviaSprite", RpcTarget.Others, numcarta);
    }
    else
    {
        photonView.RPC("CanviaSprite", RpcTarget.All, numcarta);
    }
    photonView.RPC("CartaOculta", RpcTarget.All, false);
}
```

Codi 24. Canvi dels elements necessaris per mostrar la carta

6.3.7 Menú d'opcions dintre del joc

S'ha decidit implementar un menú a dintre del joc on podrem seleccionar si volem barallar o repartir les cartes, és bastant simple aquest es mostra quan premem el botó que hem creat i ens deixa triar entre les dues opcions que hem mencionat anteriorment.

Figura 39. Botó utilitzat per al menú

Aquest menú s'ha creat amb transparència per a que es pugui veure el joc de fons mentre es tria quina opció seleccionar.

Figura 40. Mostra les opcions al obrir el menú

6.3.8 Control del Joc

Totes les funcions i eines que hem creat anteriorment necessiten d'algun element que les administri, per a complir aquest objectiu es crea l'script **JuegoCartasGameManager**, aquest serà l'encarregat d'inicialitzar el joc per a tots els jugadors, donar-los la informació que s'ha introduït en la interfície principal al configurar la sala i d'administrar les accions que els jugadors vulguin fer referents a la baralla.

6.3.8.1 Començament de la partida

El sistema de començament de la partida ja venia implementat en la "Demo" de Photon, aquest bàsicament espera a que tots els jugadors hagin carregat el nivell i comença un compte enrere, en aquest ordre:

1. OnPlayerPropertiesUpdate
   Es crida quan detecta que un element de les propietats dels jugadors s'ha canviat, en aquest cas "PlayerLoadedLevel".

2. CheckAllPlayerLoadedLevel
   Es cridada per l'anterior funció, aquesta s'encarrega de comprovar que tots els jugadors estan llestos, accedeix a les propietats de cadascun dels jugadors i comprova

el valor de la taula de hash “PlayerLoadedLevel”, aquest l’extreu com un booleà. Si el booleà resulta ser verdader continua comprovant la resta de jugadors, en cas de que hagi pogut recórrer tots els jugadors retorna verdader.

3. SetStartTime
   Si tots els jugadors estaven llestos i no s’havia començat cap compte enrere, la primera funció crida a aquesta funció de la classe **CountdownTimer**.

4. OnCountDownTimerIsExpired
   Quan el temporitzador de la classe **CountDownTimer** s’acaba crida a aquesta funció mitjançant delegació. Quan es crida a aquesta funció aquesta finalment crida a **StartGame**, el que fa que comenci el joc.

5. StartGame
   La única funció que hem modificat, en aquest cas si el jugador és el creador de la partida és ell qui crea la baralla.
   Aprofitem aquesta funció per activar el botó de menú només per al creador de la partida, ja que és ell qui té totes les variables que necessitem.
   També es crida a la funció **ArreglaSpawnpoints** que elimina els llocs de jugadors que no son utilitzats i així no repartir cartes en aquests llocs posteriorment.

6.3.8.2 Crear, Mesclar i Repartir la baralla

S’han implementat les funcions de mesclar i repartir cartes, aquestes reben la informació a partir de les dades introduïdes en el menú principal i criden les funcions creades dintre de la classe **Baraja** per a assolir el seu objectiu.

- CrearBaraja
  Inicialitza la baralla amb les dades que se li passen per paràmetre i afegeix aquesta a dins d’una llista de baralles per a futures implementacions

- RepartirBarajaButton
  S’activa quan premem el botó del menú per a repartir la baralla, crida a la funció de repartir baralla que es troba a dins de la classe **Baraja**, tot passant-li els llocs on repartir i el nombre de cartes que s’han de repartir.

- MesclarBarajaButton
  S’activa quan premem el botó del menú per a mesclar la baralla, crida a la funció per mesclar la baralla, aquesta es troba dins de la classe **Baraja**, no rep res per paràmetre ja que la funció de mesclar baralla està completament implementada a dins de la classe.

6.3.9 Transferència d'informació

S'ha creat una instància anomenada **DataManager**, aquesta s'encarrega de guardar la informació que introduïm en el menú principal, a part d'algunes constants que ja hem definit anteriorment.

- Llista d'enters togglesfinal
  Aquesta és la variable on guardarem les posicions dels jugadors introduïdes en el tauler de configuració del menú principal.
- Enter barallatriada
  En aquesta variables es guardarà la baralla que hem triat en el menú principal
- Enter numcartesrepartir
  Aqui es guarda el nombre de cartes que volem repartir, aquesta també l'hem introduït en el menú principal.

```
public List<int> togglesfinal = new List<int>();
public int barallatriada;
public int numcartesrepartir;
```

Codi 25. Variables extretes de la configuració de partida

Per a que aquesta instància pugui transferir les dades entre escenes s'ha activat la opció **DontDestroyOnLoad** en la funció **Awake** que s'executa en el moment en el que aquesta és creada.

```
// Do not destroy this object when we load a new scene.
DontDestroyOnLoad(gameObject);
```

Codi 26. Evita la destrucció de les dades en el canvi d'escena

6.3.10 Implementació del joc en una pàgina web

Finalment, per incrustar el joc en una pàgina web aquest ha sigut compilat amb l'eina WebGL que es pot trobar a dintre de l'Asset Store d'Unity.

Aquest s'incrusta a dintre d'una pàgina web programada amb HTML5 utilitzant un wrapper[20] i un iframe[21], tot definint nosaltres la resolució que preferim.

```
<html>
      <body>
       <h1>Joc de Cartes</h1>
       </p> Joc de Cartes</p>
       <div id="wrapper">
       <iframe src="JuegoCartas/index.html" style="border:0px #000000 none;"
      name="joctest" scrolling="no" frameborder="1" marginheight="px"
      marginwidth="320px" height="720px" width="1080px"></iframe>
       </div>
      </body>
<html>
```

Aquesta pot ser una futura implementació de l'aplicació, ja que no requereix cap instal·lació i proporciona facilitat per ser distribuïda.

[20] Embolcall.
[21] Permet inserir un document HTML a dintre d'un altre document HTML

7 Test del Programa

Per a tots els testos s'ha duplicat el codi del programa i s'ha executat un altra instància d'Unity, d'aquesta forma es poden dur a terme les proves i a la vegada comprovar la sincronia de tots els elements del joc.

7.1 Connexió a Photon

La connexió a Photon es realitza quan es prem el botó d'Entrar després d'haver introduït el Nick, introduïm el Nick "Jugador" i esperem que en la consola ens aparegui que s'ha connectat correctament.

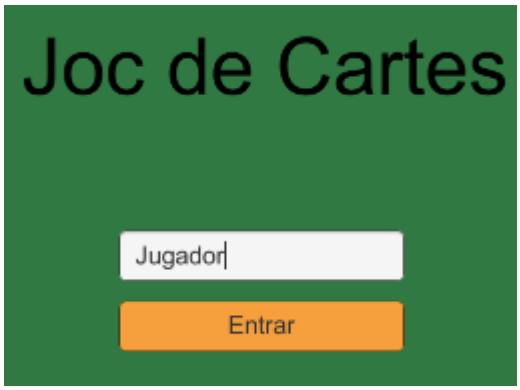

Figura 41. Prova de la introducció de Nick i connexió

Ens apareix la pantalla de selecció d'opcions i es mostra el text de sortida que hem programat en cas de realitzar-se correctament la connexió.

Figura 42. Resultat positiu de la connexió

Requisit	Resultat	Valoració
L'usuari ha de tenir un Nick per ser identificat	L'usuari té el seu propi Nick.	OK
L'usuari s'ha de poder connectar al servidor de Photon	L'usuari es pot connectar al servidor de Photon.	OK

Taula 1. Joc de proves connexió al servidor

7.2 Crear i configurar sala

Per comprovar que la sala es creada correctament, primer l'haurem de configurar, triem les opcions que volem i les entrem a les entrades corresponents, comprovem que la configuració que hem introduït de moment s'ha introduït correctament a la sala.

Requisit	Resultat	Valoració
L'usuari màster ha de poder crear la seva pròpia sala	L'usuari pot crear la seva pròpia sala.	OK
L'usuari màster ha de poder configurar la sala abans de crear-la.	L'usuari pot introduir les opcions de configuració de la sala que vol crear.	OK

Taula 2. Joc de proves creació i configuració de sala

7.3 Cerca de sales

Per a comprovar si la llista de sales es mostra correctament farem crear una sala a un dels jugadors amb un nom específic i intentarem accedir-hi des de l'altra instància.

El jugador de l'esquerra es el creador de la sala, ha creat la sala amb un màxim de 5 jugadors i el nom "Nom d'exemple". Ell veu com està a dintre de la sala a l'espera d'altres jugadors per entrar.

De mentre, el jugador de la dreta ha triat entrar a dintre de la Llista de sales, aquest, en canvi, pot veure totes les sales creades, en aquest cas només hi ha la sala del jugador principal que hem creat nosaltres.

En les característiques de la sala a la llista, es pot veure el nom de la sala "Nom d'exemple", i també la quantitat de jugadors que hi ha a dins, a més de la quantitat màxima de jugadors. En cas de que aquest vulgui entrar a la sala només ha de prémer el botó "Entrar" que es mostra.

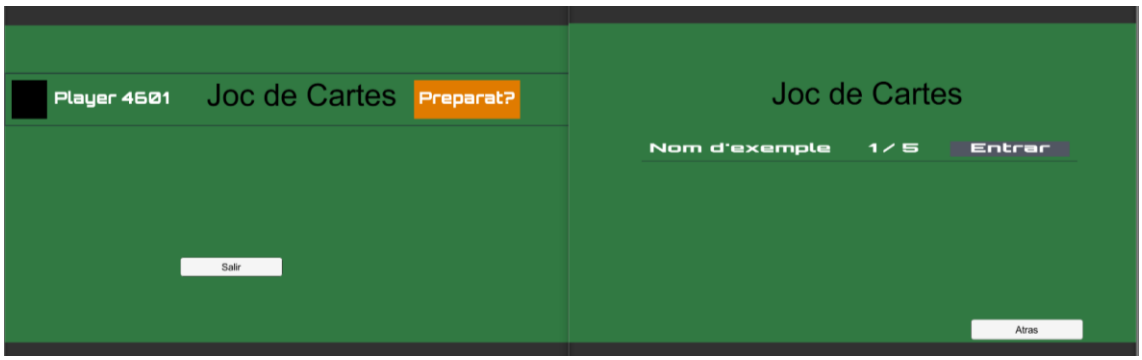

Figura 43. Mostra de la llista de sales

Requisit	Resultat	Valoració
L'usuari ha de poder cercar una sala	L'usuari pot cercar una sala a dintre de la llista de sales disponibles.	OK
L'usuari ha de poder configurar la sala abans de crear-la.	L'usuari pot introduir les opcions de configuració de la sala que vol crear.	OK

Taula 3. Joc de proves cerca de sala

7.4 Jugadors dintre de la Sala

Comprovarem la interacció entre els jugadors a dintre de la sala, simplement hem de prémer els botons de **Preparat** en cadascuna de les instancies dels jugadors, aquest canvien del text "Preparat?" al text "Preparat!" un cop es premen els botons. A la vegada apareix una icona de confirmació a la dreta.

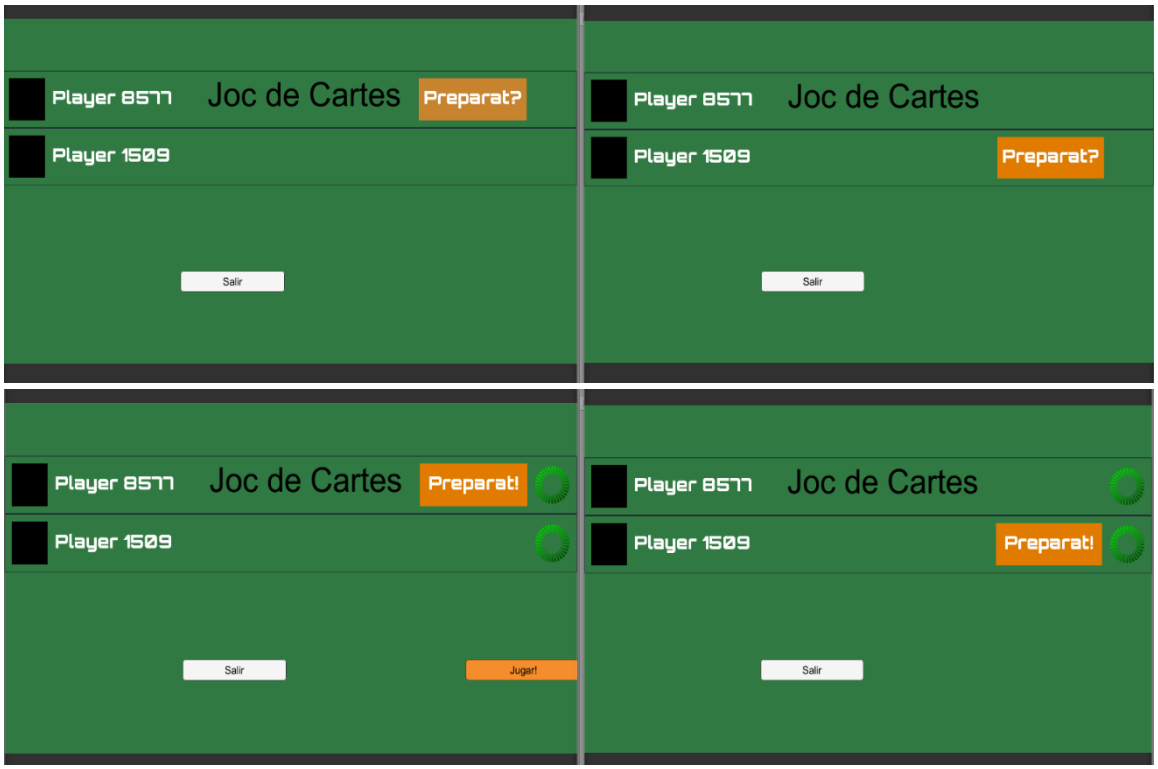

Figura 44. Mostra de la interacció dels jugadors a dintre de la sala

Requisit	Resultat	Valoració
L'usuari ha de poder decidir si està preparat per iniciar la partida	L'usuari pot prémer un botó per decidir si està preparat	OK
L'usuari màster ha de poder iniciar la partida quan tots els jugadors estiguin llestos	L'usuari màster pot iniciar la partida un cop tots els jugadors han premut el botó per indicar que estan llestos	OK

Taula 4. Joc de proves interacció a dins de la sala

7.5 Moure Carta

El mes important en el moviment de les cartes ha de ser la sincronització dels moviments que facin entre els jugadors, per comprovar-ho movem les cartes en una de les instàncies i ens assegurem que estigui sincronitzat en l'altra instància.

Amb el jugador de l'esquerra hem mogut una carta, aquesta s'ha sincronitzat amb el jugador de la dreta i s'ha mostrat el moviment en curs mentre aquest movia la carta.

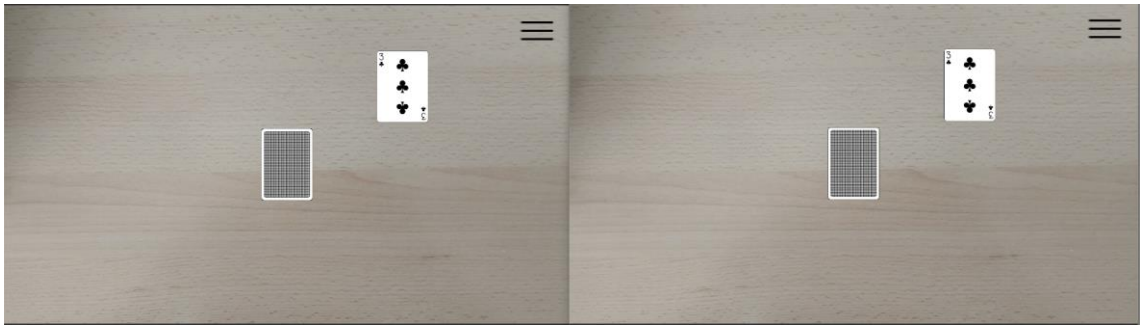

Figura 45. Prova de sincronització de moviment de les cartes

Requisit	Resultat	Valoració
L'usuari ha de poder moure les cartes.	L'usuari pot moure les cartes mitjançant el sistema Drag & Drop.	OK
El moviment de les cartes s'ha de sincronitzar amb els demés jugadors de la sala.	El moviment es sincronitza instantàniament quan es mou una carta.	OK

Taula 5. Joc de proves moviment de cartes

7.6 Privadesa Carta

La funcionalitat de mostrar carta és referent a la implementació de la privadesa de les cartes en quant una carta és assignada a un jugador, aquest jugador ha de poder veure la carta mentre que els altres jugadors no, es mostra amb una semi-transparència.

Per tant, agafem una carta amb un dels jugadors i ens assegurem que l'altre jugador no pot veure la carta en la serva instància.

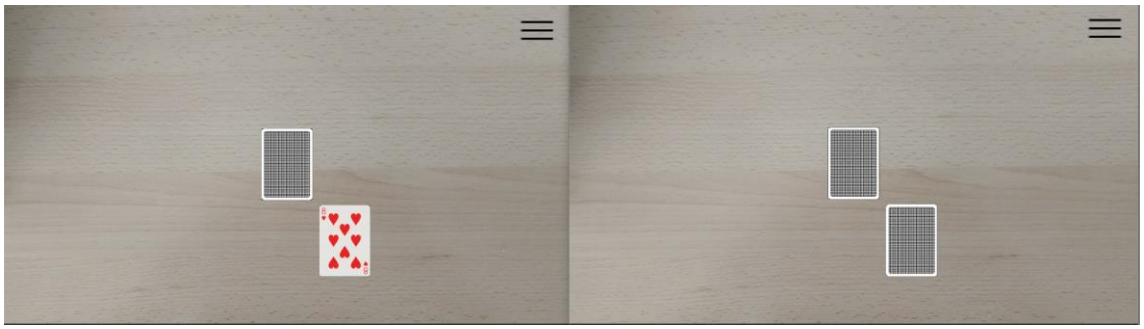

Figura 46. Prova de privacitat de la carta

Hem agafat una de les cartes de la baralla amb un jugador, aquest, pot veure el seu contingut, mentrestant, l'altre jugador no pot veure el contingut d'aquesta carta.

Requisit	Resultat	Valoració
L'usuari ha de poder veure el contingut de la carta en la seva propietat.	L'usuari pot veure el contingut de la seva carta mitjançant transparència.	OK
L'usuari no ha de poder veure el contingut de les cartes alienes.	L'usuari no pot veure el contingut de les cartes que no siguin de la seva propietat.	OK

Taula 6. Joc de proves privadesa de les cartes

7.7 Mostrar Carta

Per a aquesta prova utilitzarem la prova de la classe anterior, simplement haurem de fer doble click al damunt d'una carta i comprovar que aquesta s'ha mostrat a l'altre jugador.

Figura 47. Prova per mostrar el contingut de la carta

Comprovem que l'altre jugador pot veure la carta des del moment en que l'altre jugador ha decidit fer doble click per mostrar el seu contingut.

Requisit	Resultat	Valoració
L'usuari ha de poder mostrar el contingut de la seva carta.	L'usuari pot mostrar el contingut de la seva carta fent doble-click al damunt.	OK
L'usuari ha de poder veure el contingut de les cartes que es mostren	L'usuari pot veure el contingut de les cartes que es volen mostrar gràcies a la sincronització.	OK

Taula 7. Joc de proves mostrar contingut de la carta

7.8 Mesclar Cartes

Per a comprovar si les cartes es mesclen bé amb el nostre algoritme procedirem a mesclar les cartes a partir del botó que hem implementat en el menú que crida la funció que s'encarrega de mesclar les cartes.

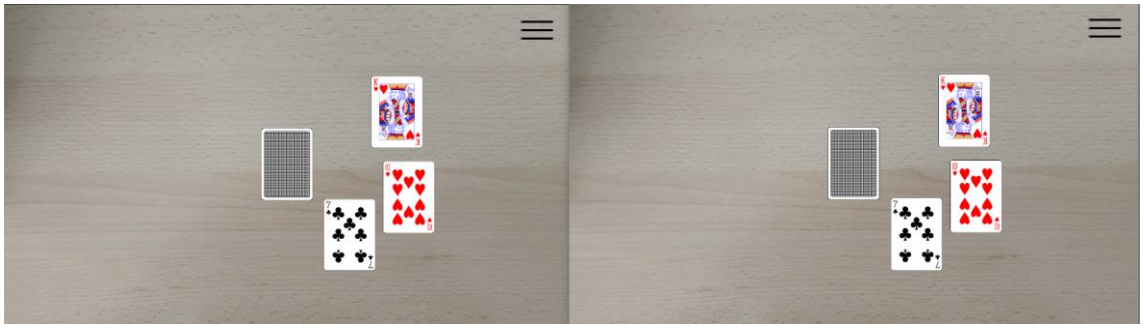

Figura 48. Primera mescla de cartes

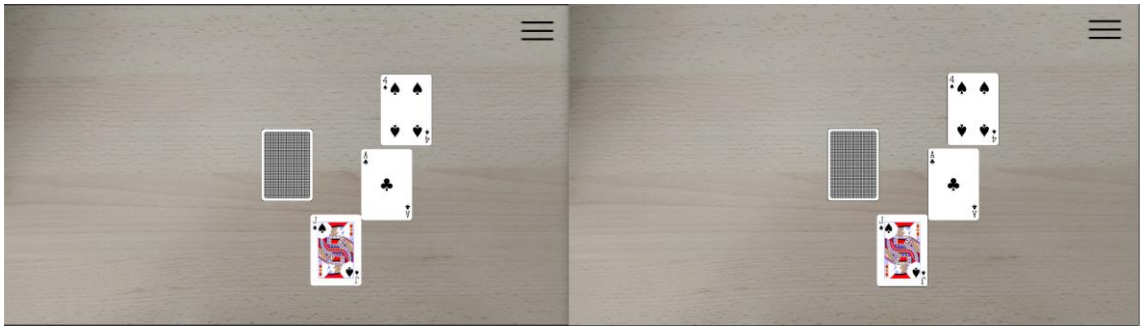

Figura 49. Segona mescla de cartes

Comprovem que les cartes s'han mesclat correctament, és important que vegem que ninguna s'ha repetit el que significa en principi que la mescla de cartes és òptima.

Requisit	Resultat	Valoració
L'usuari màster ha de poder mesclar les cartes.	L'usuari màster pot mesclar les cartes utilitzant el menú d'opcions implementat.	OK
Les cartes mesclades s'han de sincronitzar amb tots els jugadors.	La posició de les cartes és sincronitzada amb tots els jugadors.	OK

Taula 8. Joc de proves mescla de les cartes

7.9 Repartir Cartes

Per a que la repartició de cartes sigui correcta és necessari que aquestes es reparteixen just en la posició del jugador que ha triat el creador de la sala, primer de tot configurarem el tauler per a que els jugadors es trobin en extrems oposats i així comprovar-ho visualment.

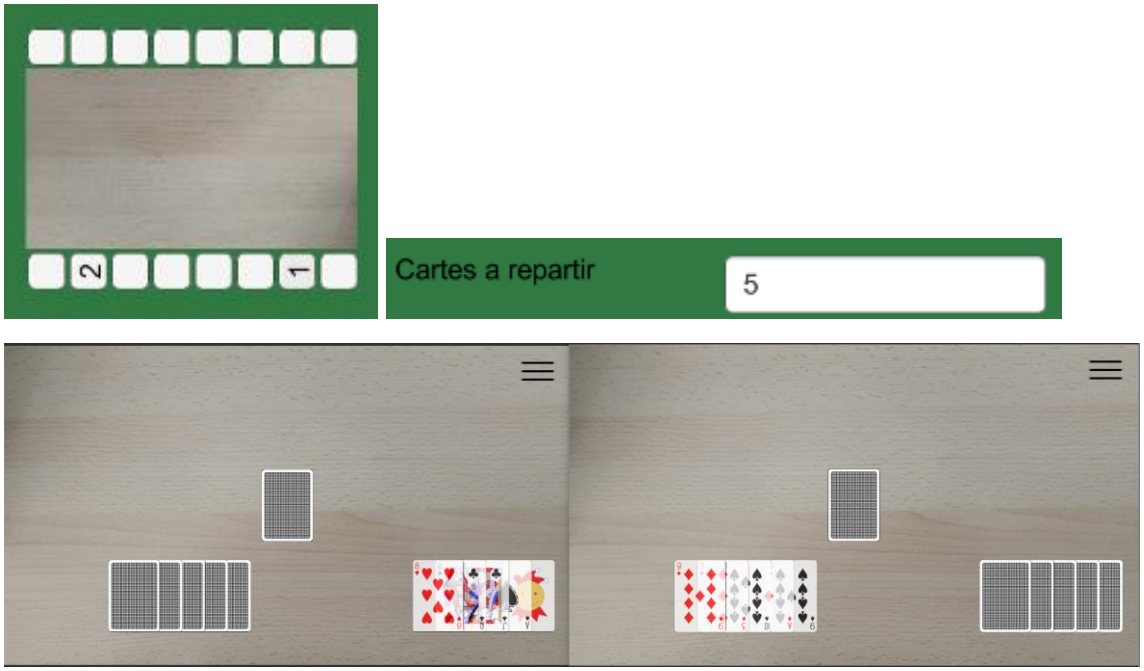

Figura 50. Configuració i prova de repartir cartes

Veiem que s'ha repartit la quantitat de les cartes que hem especificat en la creació de la sala i en la posició especificada, a la vegada també veiem que la funció de privadesa de les cartes també funciona correctament.

Requisit	Resultat	Valoració
L'usuari màster ha de poder repartir les cartes.	L'usuari màster pot repartir les cartes utilitzant el menú d'opcions implementat.	OK
Les cartes repartides han de passar a ser propietat del jugador al que se l'ha repartit.	Les cartes repartides passen a ser propietat del jugador repartit.	OK
Les cartes repartides s'han de sincronitzar amb tots els jugadors.	La posició de les cartes és sincronitzada amb tots els jugadors.	OK

Taula 9. Joc de proves repartiment de les cartes

8 Conclusions

S'ha tractat d'un projecte llarg i complex degut a les diferents dificultats plantejades, no tenia molta experiència amb el programari utilitzat, la única experiència que presentava era utilitzant l'entorn Unity degut a que el vam utilitzar en una assignatura.

Tampoc tenia molta experiència prèvia amb C#, però al ser aquest bastant paregut a alguns llenguatges de programació orientats a objectes i tenir el mateix funcionament va ser el que menys hem va costar d'adaptar-me.

Gràcies a les possibilitats que brinden les eines mencionades anteriorment ha sigut possible implementar la majoria de característiques que s'havien plantejat i inclús es poden seguir ampliant.

L'apartat que m'ha ocupat mes temps ha sigut la de depuració del codi, sempre hi ha alguna part del codi que s'ha implementat d'una forma i funciona correctament però al cap del temps acabes entenent millor el seu funcionament i s'implementa d'una forma més eficient.

Aquest projecte ha resultat ser una font d'aprenentatge de nous temaris com la sincronia d'objectes a través d'una xarxa, la abstracció de funcions bàsiques des de múltiples jocs de taula cap al món dels videojocs i la necessitat d'utilització de llibreries de certa complexitat, s'ha de dedicar temps a entendre el seu funcionament per a realitzar una millor implementació.

Un cop acabat, aquest projecte serveix de base per a la realització de qualsevol altre projecte similar en el futur.

9 Recursos utilitzats

[1] Unity. (2021). Unity.
https://unity.com/es

[2] PUN 2 - FREE | Network. (2020, 24 gener). Unity Asset Store.
https://assetstore.unity.com/packages/tools/network/pun-2-free-119922

[3] Photon Unity Networking 2: Class Index. (2021). Photon API
https://doc-api.photonengine.com/en/pun/v2/classes.html

[4] Introduction | Photon Engine. (2021). Photon Documentation.
https://doc.photonengine.com/zh-cn/pun/v2/demos-and-tutorials/pun-basics-tutorial/intro

[5] Microsoft. (2021, 9 novembre). Visual Studio: IDE i Editor de codi per a desenvolupadors de software y Teams. Visual Studio.
https://visualstudio.microsoft.com/es/

[6] Microsoft. (2021). Documents de C#. Microsoft Docs.
https://docs.microsoft.com/es-es/dotnet/csharp/

[7] Col·laboradors de Wikipedia. (2021, 27 agost). Algoritme de Fisher-Yates. Wikipedia, la enciclopèdia lliure.
https://es.wikipedia.org/wiki/Algoritmo_de_Fisher-Yates

[8] Implement a Drag and Drop Script with C#. (2021). Unity Forum.
https://forum.unity.com/threads/implement-a-drag-and-drop-script-with-c.130515/

10 Annex

10.1 Instal·lació

Gràcies a les possibilitats de compilació amb Unity per als diferents sistemes existents tenim la possibilitat d'incrustar el joc en una pàgina web.

En el cas de que la pàgina web que hem implementat estigui en línia només s'hauria d'accedir a aquesta per a poder jugar-hi.

També es pot utilitzar l'executable per a Windows en cas de que la pàgina web estigui fora de línia.

10.2 Requisits

Es requereix una connexió a internet.

10.3 Errors coneguts

- Es poden treure les cartes a fora del terreny de joc.
- Si s'intenta mesclar les cartes i repartir-les en un període de temps molt curt aquestes no es repartiran correctament.
- En el Lobby el text del títol podria solapar algun nom de jugador si la resolució és molt baixa.
- En el joc podria no haver suficient espai per jugar si la resolució és molt baixa.
- La mesura de les cartes és massa gran per a que hi puguin jugar 8 jugadors simultàniament, aquesta s'hauria d'adaptar a la quantitat de jugadors.