Universitat Autònoma de Barcelona

escola d'enginyeria

2221: VIRTUALITZACIÓ DE FIRES I CIUTATS

Memòria del Projecte Fi de Carrera
d'Enginyeria en Informàtica
realitzat per
Alejandro Bernabé Sánchez
i dirigit per
Enric Martí Gòdia
Bellaterra, 22 de juny de 2010

Universitat Autònoma de Barcelona

Escola Tècnica Superior d'Enginyeria

El sotasignat, Enric Martí Gòdia

Professor/a de l'Escola Tècnica Superior d'Enginyeria de la UAB,

CERTIFICA:

Que el treball a què correspon aquesta memòria ha estat realitzat sota la seva direcció per en

I per tal que consti firma la present.

Signat: .....

Bellaterra, 22 de juny de 2010

Taula de continguts

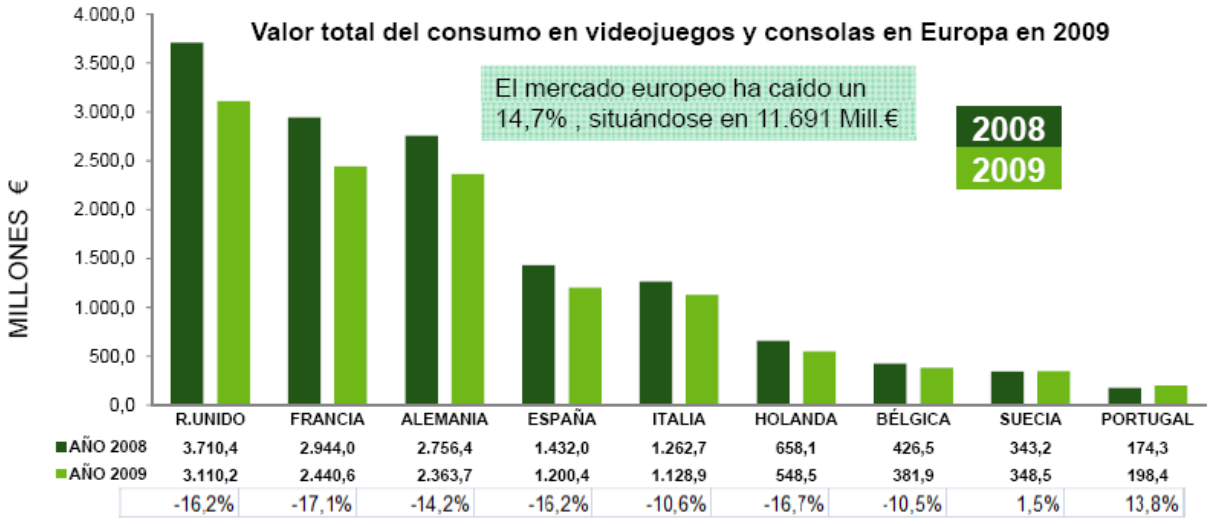
1. INTRODUCCIÓ

Situació de la indústria del videojoc a l'actualitat

Deixant de banda la crisis mundial que afecta a tota la població i a tots els sectors, la indústria del videojoc és una de les que ha obtingut un creixement més gran els últims anys.

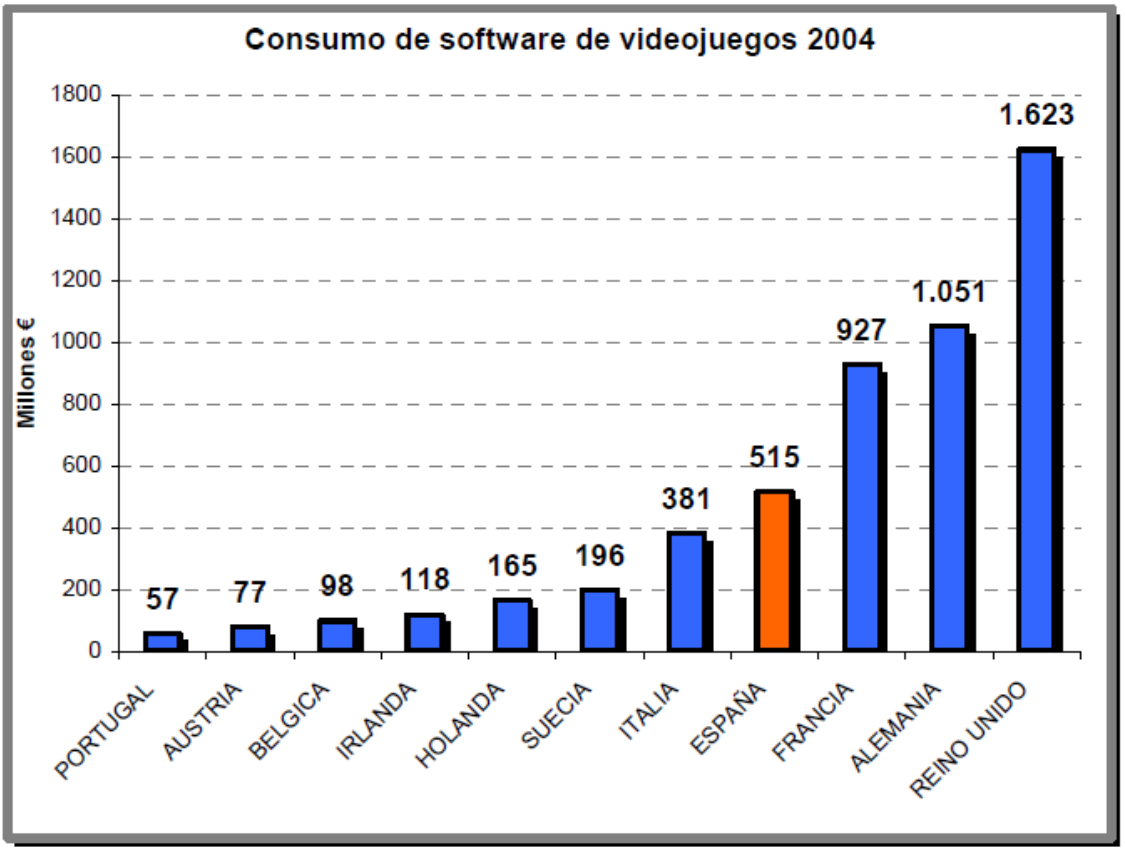
Segons ADESE (1) (Asociación Española de Distribuidores y Editores de Software de Entretenimiento) la industria del videojoc va aconseguir el 53% del mercat d'entreteniment audiovisual i interactiu, el qual està format per *cinema, DVD, música* i *videojocs*. De fet, les vendes aconseguides per la industria de software i hardware d'entreteniment supera la suma del que va registrar cinema, DVD i música junts.

Aquest creixement no és només perceptible a Espanya, sinó que es un efecte que s'està produint arreu del món. A tots els països de la UE (Unió Europea), EEUU (Estats Units) i Japó, principalment, són els països amb un nombre de consumidors més grans dins d'aquesta industria. Concretament, a Espanya, i dins el marc de la UE, és el 4t país amb un nombre més elevat de consum d'aquest sector.



Il·lustració 1: Valor total del consum de videojocs i consoles a la UE al 2009

Tal i com es pot observar a la *Il·lustració 1*, tot i la davallada que han sofert tots els països d'Europa (provocada per la crisi mundial econòmica), el mercat europeu mou a l'any més de 10.000 milions d'Euros, xifres astronòmiques i impensables fa pocs anys, on les xifres es situaven bora els 500 milions d'Euros anuals l'any 2004 a Espanya, segons la mateixa associació. Tal i com es pot observar al següent gràfic:



Il·lustració 2: gràfica del consum de software de videojocs l'any 2004, extret de (2)

Es per tot això que la industria del videojoc a esdevingut un món molt competitiu, essent només els productes amb més qualitat i amb una millor campanya de comunicació (publicitat).

L'efecte "Internet" sobre el món dels videojocs

El *boom* sofert per Internet és un fet i una realitat amb la que convivim dia a dia, i la industria del videojoc no podia quedar al marge. En una societat cada vegada més informatitzada i amb l'arribada de conceptes com "Web 2.0" i "Xarxes socials", les empreses i estudis de videojocs han hagut d'adaptar-se i redefinir el concepte de videojoc.

Cada cop més, sense deixar de banda el model clàssic de videojoc, i gràcies a les comunicacions que permet Internet, els videojoc tendeixen a suportar (o basar-se exclusivament) en la interacció multijugador.

Són molts ja els títols basats únicament en jugabilitat multijugador, com per exemple, un dels més coneguts arreu del món: *World of Warcarft*[1]. Segons els propis creadors del videojoc,

[1] http://www.worldofwarcraft.com

tenen un volum de gent registrada que supera els 11 milions de persones i, d'aquests, més d'un milió són jugadors actius actualment.

Un altre tipus de tendència que estan adquirint els videojocs, especialment els més orientats a ser multijugador, és la de "socialitzar" al màxim els videojocs. Amb tota mena d'estratègies (sistema d'amics, missatges privats, creació de perfils, etc), els videojocs intenten imitar les grans webs dedicades a aquests aspectes, com poden ser *facebook*[2] o *twitter*[3]. Aquestes pàgines mouen milions de persones diàriament i això implica un gran nombre de possibles jugadors, és per això que els videojocs intenten abordar o incloure aquestes temàtiques dins dels seus àmbits.

Il·lustració 3: Logotip del joc online World of Warcraft

Tal magnitud ha assolit la importància d'Internet a la indústria del videojoc que fins i tot les empreses més grans en desenvolupament de videoconsoles han creat xarxes per a que els jocs dins la seva plataforma puguin oferir serveis online i, des d'allà, també es puguin oferir serveis no relacionats directament amb els videojocs (com pot ser el lloguer de pel·lícules).

Objectiu del producte

L'objectiu del producte del qual forma part aquest projecte és el de crear una plataforma online en la qual convisquin temes com l'entreteniment, el món dels negocis i les xarxes socials dins un mateix món.

Així doncs, i de manera més concreta, el producte té els següents requeriments:

- Ser totalment jugable des del navegador, sense instal·lacions prèvies per evitar la limitació física a l'hora de jugar (no s'està obligat a jugar des de l'ordinador de casa, per exemple).
- Tenir un sistema de negocis on empreses puguin muntar el seu espai de negocis dins la plataforma i oferir serveis des de dins d'aquesta.

[2] http://www.facebook.com
[3] http://www.twitter.com

- Tenir un sistema de xarxa social per a que els usuaris puguin interactuar amb altres sense la necessitat de ser una relació estrictament empresària, sinó activitats socials.
- Tenir un sistema d'entreteniment, a través del qual els usuaris puguin gaudir i divertir-se dins la plataforma.

Motivacions

Com s'ha explicat anteriorment, el món del videojoc és un món en constant creixement, i que s'accentua amb l'entrada de les xarxes socials i Internet en aquest món. Així doncs, el fet de poder oferir aquests serveis i donada les possibilitats de difusió que existeixen actualment, han fet de motivació suficient per a la creació d'aquest projecte.

A més, la possibilitat d'unir 3 móns tan diferents com els esmentats anteriorment, preveuen un ventall d'usuaris molt més gran i voluminós.

És un sector molt nou i en el que gairebé no hi ha competència, així doncs ser un producte pioner dóna una motivació extra per al desenvolupament del producte.

Aproximacions

Existeixen molts productes de videojocs orientats a Internet, però no n'hi ha gaires que intentin oferir alguna cosa més que simple entreteniment online.

De fet, només hi ha un exemple clar i conegut de producte similar: *Second Life*[4].

Second Life és una plataforma 3D que ofereix un món "alternatiu" al real, on es poden fer activitats vàries imitant a les de la realitat. Així doncs, es pot tenir un *avatar*[5] amb el qual es pot tenir una feina, uns diners, unes possessions, una religió...

Aquest producte, però, té limitacions. Per exemple, fa ús d'un instal·lador, cosa que limita l'accés des de diferents terminals fora de casa. Un altre aspecte negatiu és la qualitat gràfica que ofereix, que no és del tot bona. També cal tenir present el fet que, en quant a entreteniment pròpiament dit (el que s'entén com a concepte de "videojoc") no en té gairebé, així que tanca la porta a un gran ventall de possibles usuaris.

D'aquesta manera, es poden comparar les característiques d'aquest producte amb les del nostre producte a la següent taula:

[4] http://secondlife.com/?v=1.1
[5] Representació de la teva persona dins un univers virtual

Taula 1: Comparació característiques dels productes

	SECOND LIFE	DOME[6]
Multijugador a través d'Internet	✓	✓
Executable des del navegador	×	✓
Sense instalacions	×	✓
Orientat a negocis	✓	✓
Orientat a xarxes socials	✓	✓
Orientat a entreteniment	×	✓
Qualitat gràfica	↓	↑

Objectiu del projecte

Lògicament, un videojoc, i més d'aquestes característiques, està desenvolupat per multitud de professionals, de diferents especialitats on cadascú afronta una part del producte.

Els objectius d'aquest projecte formaran, doncs, part del producte final, però sense contenir totes les funcionalitats descrites anteriorment.

El projecte afrontarà objectius de la part de programació que es requereix per a dur a terme el producte. Concretament, els objectius d'aquest projecte són:

- Creació d'un ***editor d'stands*** per a cobrir la necessitat de la part orientada a negocis a que els usuaris que així ho vulguin puguin crear el seu propi espai de negocis.
- Creació d'un ***editor d'avatars*** per a cobrir la necessitat de que la gent pugui personalitzar el seu avatar amb el qual viatjarà per la plataforma.
- Creació d'un sistema de ***frontend d'inventari*** per a que els usuaris puguin visualitzar i interactuar amb els objectes que comprin i tinguin emmagatzemats al seu inventari.
- Creació d'un conjunt de ***minijocs*** per a cobrir la part de la plataforma orientada a l'entreteniment.
- Creació d'un sistema de ***rànking*** per a donar suport als minijocs i oferir, en certa manera, una mica de *competitivitat* entre usuaris.
- Creació d'un sistema d' ***intercomunicació client-base de dades*** per a cobrir les necessitats que puguin sorgir durant el desenvolupament del joc de cridar a informació emmagatzemada a BBDD.
- Creació d'un sistema de ***control d'àudio*** per a poder configurar en qualsevol moment el volum de la música i dels efectes del joc.

De tots els objectius, només s'afronta en aquest projecte la part lògica: *gameplay*, interfície d'usuari, àudio, etc. Tot el que fa referència a modelatge 3D, creació de melodies, disseny gràfic de les interfícies...és obviat en aquest projecte, ja que aquest no és el propòsit d'aquest.

[6] Nom del producte del qual forma part aquest projecte

Resum per temes

Tot seguit es farà una breu explicació del contingut de cada un dels punts que vindran a continuació.

2. DESENVOLUPAMENT

Aquest capítol conté informació tècnica respecte al desenvolupament de cada un dels objectius esmentats del projecte. Descriurà el funcionament de cada un dels mòduls i la seva arquitectura. Com que el propòsit d'aquest punt no és el de explicar molt específicament com s'ha desenvolupat cada mòdul, s'evitarà, en la mesura del possible, la introducció de codi font al text. S'intentarà donar una idea general del funcionament d'aquests mòduls.

3. RESULTATS

Aquí s'oferiran mostres de quin ha sigut el resultat en cada un dels diferents mòduls sobre diverses situacions, per a demostrar el correcte funcionament dels mòduls. S'introduiran imatges i una petita descripció de què es pot veure i a que es degut aquell comportament.

4. CONCLUSIONS I MILLORES

En aquest capítol es farà una reflexió sobre el desenvolupament del projecte, així com un llistat de incidències i possibles millores de cara a una hipotètica continuació del projecte després de la seva entrega.

5. BIBLIOGRAFIA

Contindrà les referències bibliogràfiques de les quals s'hagi extret directa o indirectament informació per al desenvolupament del projecte.

2. DESENVOLUPAMENT

Eines utilitzades per al desenvolupament del projecte

A continuació es nombraran les eines software que han fet possible el desenvolupament del projecte.

Visual Studio 2008[7]

Il·lustració 4: Logo IDE Visual Studio 2008

IDE de desenvolupament de software creat per a escriure el codi font dels programes. En aquest cas, ha servit per a escriure els scripts del joc.

S'ha escollit aquest software perquè, gràcies a l'acord amb la UAB, els estudiants tenen llicència gratuïta i poden gaudir de totes les prestacions d'aquest producte.

Unity 3D v2.6[8]

Il·lustració 5: Logo del motor gràfic Unity 3D

Unity 3D és un motor gràfic comercial que permet el desenvolupament de videojocs sense preocupar-se de les capes més baixes. D'aquesta manera, els programadors només s'han de centrar en el joc en sí.

Unity 3D treballa tant amb *DirectX* com amb *OpenGL*, pel que es converteix en un motor gràfic multiplataforma, oferint suport tant per a *Windows* com per a MacOS.

[7] http://msdn.microsoft.com/es-es/vstudio/default.aspx
[8] http://unity3d.com

Una de les característiques més importants d'aquest producte (i motiu pel qual es va escollir) és la capacitat que té per a crear videojocs executats directament al navegador, sense necessitat d'instal·lacions prèvies ni necessitats d'un sistema operatiu específic.

Una altra característica important és els llenguatges per a *scripting* que suporta. Els *scripts* són petits codis interpretats que són passats al motor gràfic i, aquest, es capaç d'entendre-ho i traduir la informació per a transformar-ho a codi entès per a les capes més baixes del motor. Unity 3D suporta *C#* i *JavaScript*.

Per al desenvolupament del projecte s'ha escollit *C#* per la potència que ofereix aquest llenguatge respecte a *JavaScript*. És orientat a objectes i permet fer servir el framework propi de C#, cosa que ofereix una àmplia gama de possibilitats.

Mòduls realitzats

Com s'ha descrit al final del capítol 1, s'han realitzat una sèrie de mòduls que formen part d'un projecte que els engloba tots i del qual formen part. A continuació es descriuran de manera més detallada, però intentant deixar de banda al màxim possible la inserció de codi, per a evitar que el lector d'aquesta memòria es perdi entre les línies de codi.

Intercomunicació Client – BBDD

Objectiu del mòdul

Aquest mòdul té com a objectiu el de poder intercomunicar a l'usuari amb la base de dades. D'aquesta manera, l'usuari podrà accedir a informació "externa" al joc sense necessitat de deixar-lo. Amb aquest mòdul també es pot aconseguir guardar informació relacionada amb l'usuari, ja siguin accions que ha fet, objectes que té, diners que ha guanyat...i poder recuperar aquesta informació més endavant.

Per molts motius, aquesta metodologia és millor que, per exemple, guardar aquesta informació a disc. Primerament, al ser un videojoc executat des de navegador, fa que les restriccions de seguretat siguin molt altes i que, per exemple, sigui impossible guardar arxius a disc. A més a més, guardar aquests arxius a disc provoca una dependència de la màquina des d'on es juga, ja que aquesta informació només seria accessible des de la màquina des d'on s'hagi jugat, i aquest és un dels punts del projecte que es volia evitar. Per acabar, en el cas de guardar informació relacionada amb l'usuari a disc, podria provocar que terceres persones s'apoderessin d'aquesta informació i la fessin servir en benefici propi.

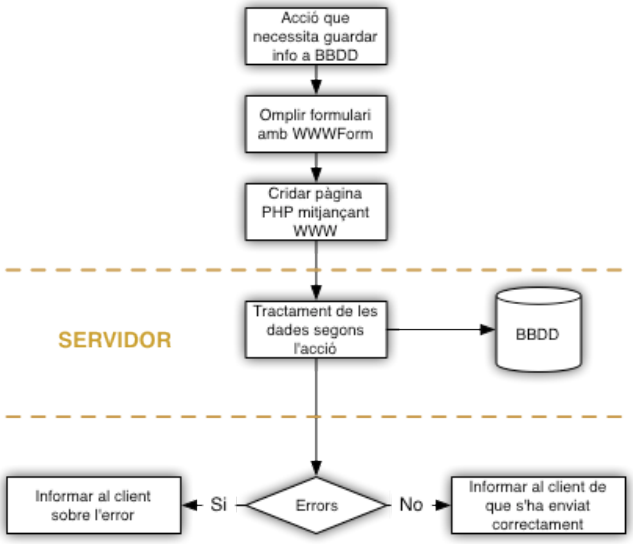
És per tot això que la intercomunicació entre la part client del joc i la BBDD es feia imprescindible.

Què ofereix Unity per a la realització del mòdul

En aquest cas, Unity ofereix un mecanisme molt bo per a realitzar la intercomunicació. Ofereix dos classes, accessibles des de C# i JavaScript, la `WWW` i la `WWWForm`.

Aquestes classes ofereixen intercomunicació amb pàgines externes al joc mitjançant formularis web, entre altres coses. D'aquesta manera, es recullen les dades des del client, s'omple un formulari web gràcies a la classe `WWWForm` i s'envia a una pàgina *php* mitjançant la classe `WWW` i, aquesta pàgina, s'encarrega de realitzar les *query* o les accions necessàries per a connectar amb la BBDD.

Diagrama de flux



Il·lustració 6: Diagrama de flux del mòdul d'interconnectivitat client - BBDD

Com es pot observar al diagrama, només la part del tractament de dades i la conseqüent interacció amb la BBDD és la que queda fora de la part del client.

Primerament es trobem amb alguna acció dins el joc que provoca o requereix informació de la BBDD. Aquesta acció desembocarà en l'ompliment d'un formulari web mitjançant la classe `WWWForm` que posteriorment s'enviarà a una pàgina web PHP mitjançant la classe `WWW` i a través de POST, així les dades no podrien ser visibles des d'un navegador.

Un cop a la pàgina PHP i gràcies a un dels camps del formulari on s'especifica l'acció a realitzar (camp obligatori i imprescindible en totes les peticions a BBDD per a saber què fer en cada cas) es realitzen les operacions necessàries i els tractaments necessaris amb les dades.

En cas de no haver-hi cap error, el PHP pot retornar (o no) algun tipus d'informació, en el cas de que fos necessari. En cas de que sí es produeixi un error a l'hora de fer la connexió amb la pàgina web o amb la interacció amb la BBDD, es retornarà al client una alerta de que hi ha hagut un error i especificant el motiu d'aquest error.

Segons una cosa o una altra, el client reaccionarà d'una manera o una altra per a que l'usuari sigui informat de què ha succeït.

Aquest procés requereix de temps, ja que s'han d'enviar a través d'Internet la petició a la pàgina web i fer el tractament amb les dades. Però això va en contrapartida a la definició de videojoc, ja que aquest ha de reaccionar en temps real i, per tant, el programa (en aquest cas el videojoc) no es pot quedar "bloquejat" fins a que sigui retornada la confirmació. És per això que tot aquest procés s'ha de produir de manera "paral·lela" mitjançant un *thread*. Per sort, Unity ofereix la possibilitat de crear diferents fils d'execució per a tractar aquest tipus de situacions. Gràcies a la crida a la funció `StartCoroutine(funció)` és capaç de llançar una funció en un thread i així evitar que el joc quedi a l'espera d'una resposta.

Informació addicional

Cal esmentar algun dels desavantatges d'aquesta metodologia i com s'ha solucionat. Si imaginem que una acció dins el joc requereix guardar informació a BBDD, però que aquesta informació està formada per 50 "cadenes de text", degut a que són molts els paràmetres a guardar. Això implicarà que s'han d'omplir 50 camps del formulari (més el camp d'acció), cosa que fa que la programació sigui poc eficient i tosca.

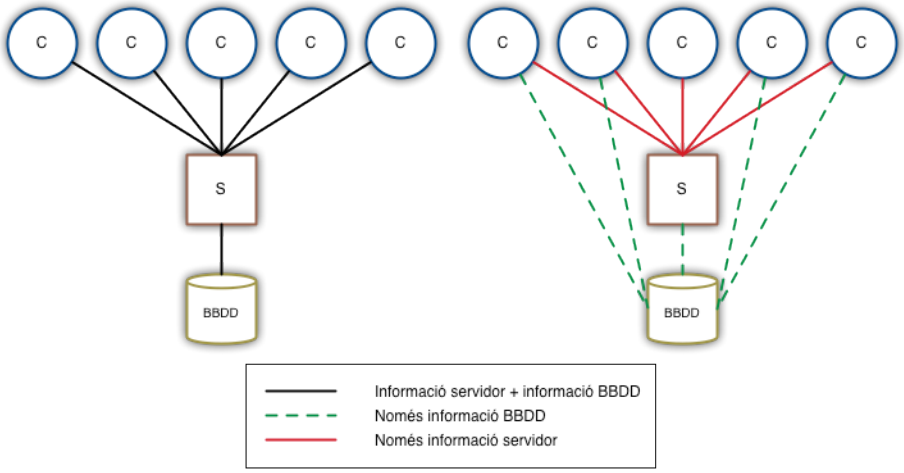
Per a evitar això, en els casos amb molta informació a guardar o rebre de BBDD, el que s'ha fet servir és estructures XML per a organitzar la informació millor. D'aquesta manera, pel formulari només serà una "cadena de text" molt llarga, igual que a l'hora d'insertar a la BBDD i pel programador, serà molt més eficient el tractament (construcció i parseig) de els dades.

Per què no a través de la part servidor?

S'ha "alliberat" d'aquesta funció al servidor per a evitar que una gran quantitat de peticions alhora provoquin que el servidor no pugui ser capaç de fer la seva funció principal: enviar la informació de posició entre usuaris.

D'aquesta manera el servidor només s'encarrega de la seva funció i evitem problemes de *lag*[9] durant el joc.

[9] Efecte que provoca que les accions que es succeeixen arribin tard a la resta d'usuaris que estan jugant.



Il·lustració 7: Esquerra: topologia on tota la informació passa a través del servidor. Dreta: topologia on el client accedeix directament a BBDD sense passar pel servidor

Control d'àudio

Objectiu del mòdul

L'objectiu d'aquest mòdul és el de tenir un control sobre tots els elements dins el joc que produeixen algun tipus de so (ja sigui melodia o efecte de so) per a poder pujar o baixar el volum d'aquests d'una manera eficient i sense que el joc vegi penalitzat el seu rendiment.

Què ofereix Unity per a la realització del mòdul

El motor en sí no ofereix concretament aquest tipus de control. Sí és cert que Unity ofereix uns components anomenats `AudioSource` que són els necessaris per a que els arxius d'àudio se sentin durant el joc i que, aquest component té un paràmetre configurable que permet configurar el volum d'aquest component.

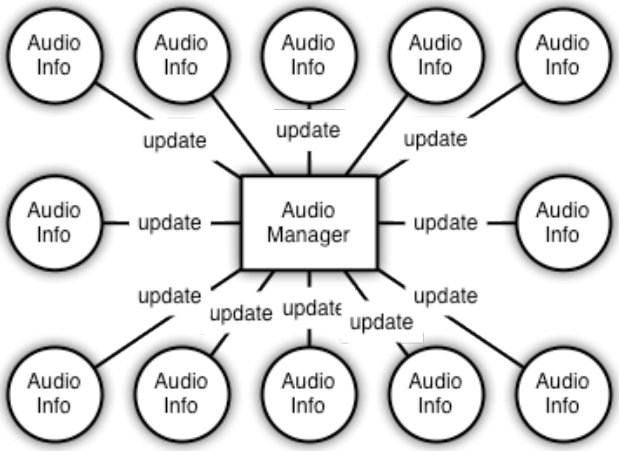
Però això no és exactament el que volem, o no directament. Imaginem la situació en la que tenim dos *GameObjects*[10] en la nostra escena i els dos han de reproduir un so, un una melodia de fons i l'altre un efecte de so que s'activa al rebre algun tipus d'estímul. Primerament, els dos objectes tindran volums diferents i per tant, farien dificultós poder emmudir-los i després recuperar el seu volum original. D'altra banda, pot donar-se el cas en el que voldríem emmudir o baixar el volum de la música però no dels efectes de so.

Per tot això, s'ha de dissenyar un sistema capaç de:

- *Diferenciar entre efecte de so i melodia*
- *No modifiqui el volum original de l'objecte*

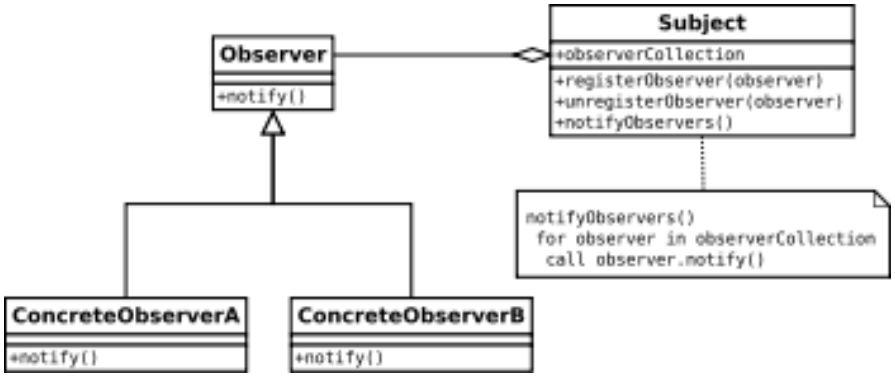
Diagrama

[10] Tots els objectes dins el Unity són tractats com a GameObjects. Cada GameObject pot contenir diferents components que són els que donaran el comportament d'aquest.



Il·lustració 8: Diagrama on es mostra la organització del mòdul de control de volum

Per a dur a terme aquest sistema per a que complís les necessitats que s'han especificat anteriorment, s'ha apostat per a adoptar el patró de disseny *observer*. Aquest consisteix en tenir un agent (en el nostre cas `AudioManager`) que conté o coneix un conjunt d'observadors (`AudioInfo`) els quals, quan l'agent canvia el seu estat, avisa a tots els observadors i els notifica aquest canvi.



Il·lustració 9: Exemple de patró de disseny observer

En el nostre cas, aquest `AudioManager` s'encarregarà de buscar a l'inici del joc tots els `AudioInfo` que es trobin a escena. Aquests últims es col·locaran a tots els objectes que requereixin reproduir algun tipus de so. Així doncs, només quan l'usuari decideixi canviar la configuració de so del joc, canviarà l'estat de l' `AudioManager` i aquest avisarà a tots els observadors per a que actualitzin el seu estat.

Aquest `AudioInfo` contenen la propietat de classificar els `AudioSource` depenent de si són melodia o efecte de so, per a poder controlar la situació abans esmentada.

Concluim doncs que, utilitzant aquest patró de disseny, aconseguim complir tots els requisits i evitar un cost computacional elevat, ja que evitem un possible *polling* per a cada `AudioInfo`.

FrontEnd inventari

Objectiu del mòdul

L'objectiu d'aquesta part és crear la part visual d'un sistema d'inventari. Aquesta part visual consisteix en:

- Visualització d'una graella amb caselles buides o plenes segons si està ocupada o no per algun tipus d'ítem.
- Carregar, des de BBDD, els ítems que té un usuari en concret.
- Visualitzar de cada item la seva quantitat.
- En cas de tenir un item, mostrar aquest a l'inventari amb una icona representativa.
- Mostrar la quantitat de diners que té l'usuari.

Queda, per tant, exclòs d'aquest mòdul tota la part de recol·lecció i interactivitat amb ítems.

Què ofereix Unity per a la realització del mòdul

Unity, com en el cas anterior no ofereix el mòdul en sí, però sí ofereix una sèrie d'eines per a la realització d'aquest.

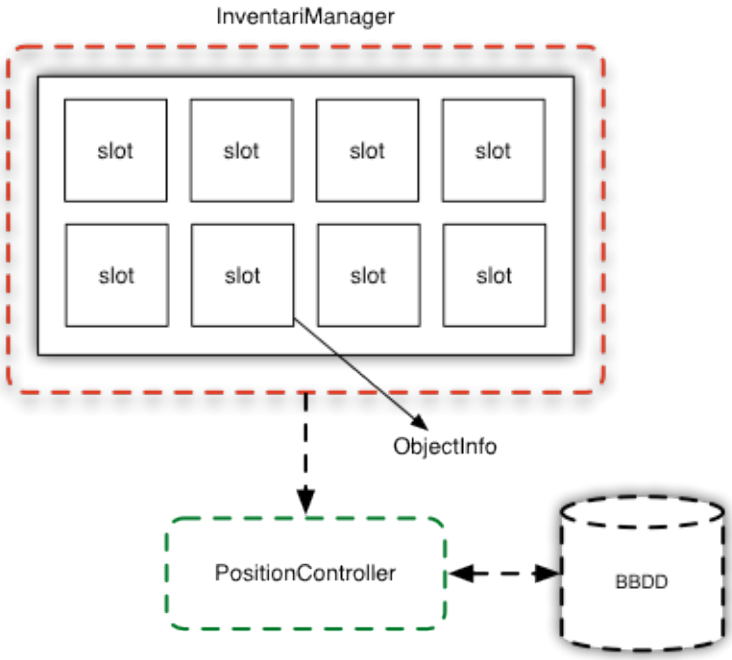
Al ser un mòdul on només té en compte la part visual, necessitarem dissenyar una interfície gràfica per oferir un resultat visual a l'usuari que hi jugui. Unity ens ofereix un sistema de GUI[11] per a aquest fi. Aquesta ofereix una sèrie d'objectes amb les seves característiques:

- *Label*: objecte que ofereix mostrar text per pantalla.
- *Button*: ofereix la possibilitat de mostrar un botó i provocar una acció apretant-lo.
- *Box*: objecte amb l'objectiu d'englobar una sèrie d'objectes per a marcar que és un conjunt.
- *Window*: finestra flotant que permet tenir a dins altres objectes GUI.
- *TextField*: camps on es pot escriure informació.
- *TextArea*: amb la mateixa funcionalitat que *TextField* però amb la diferència que aquest permet escriure en diferents línies.
- *Sliders*: permeten assignar algun valor a través d'una barra que té un valor mínim i un de màxim.
- *Scrollbars*: permeten el desplaçament per la GUI.

Així doncs, amb aquests elements i una part darrere de lògica, s'ha muntat aquest mòdul.

Diagrama

[11] *Graphic User Interface* (Interfície Gràfica d'Usuari)



Il·lustració 10: Diagrama amb els diferents components que formen l'inventari

Per a dur a terme aquest mòdul s'han realitzat 3 components, cada un amb una finalitat específica:

- `ObjectInfo`: és el component que han de tenir tots els objectes que puguin ser col·locats dins l'inventari. Són els encarregats de contenir la icona característica que apareixerà a l'inventari quan es tingui aquest objecte guardat en ell. Aquest script pot ser ampliat per a contenir més informació sobre l'objecte, com pot ser una descripció d'aquest.
- `PositionController`: és el component encarregat de guardar els objectes que hi ha a l'inventari. També és l'encarregat de comunicar-se amb la BBDD per a guardar i/o carregar els objectes de l'inventari.
- `InventariManager`: és el component "pare" de tot el sistema. És l'encarregat de dibuixar per pantalla l'inventari, fent servir la informació del `PositionController` per a saber quins espais estan ocupats i quins no i fent servir també el component `ObjectInfo` per a dibuixar la icona representativa. El component està dissenyat per a configurar el nombre de files i espais per files que es vulguin, per tant, el disseny i la distribució de l'inventari és totalment personalitzable.

Ranking

Objectiu del mòdul

L'objectiu d'aquest mòdul és el de crear un sistema capaç de recollir les dades emmagatzemades a BBDD relacionades amb les puntuacions dels jugadors als diferents

minijocs existents a la plataforma i mostrar-los per ordre en qualsevol moment que es requereixi.

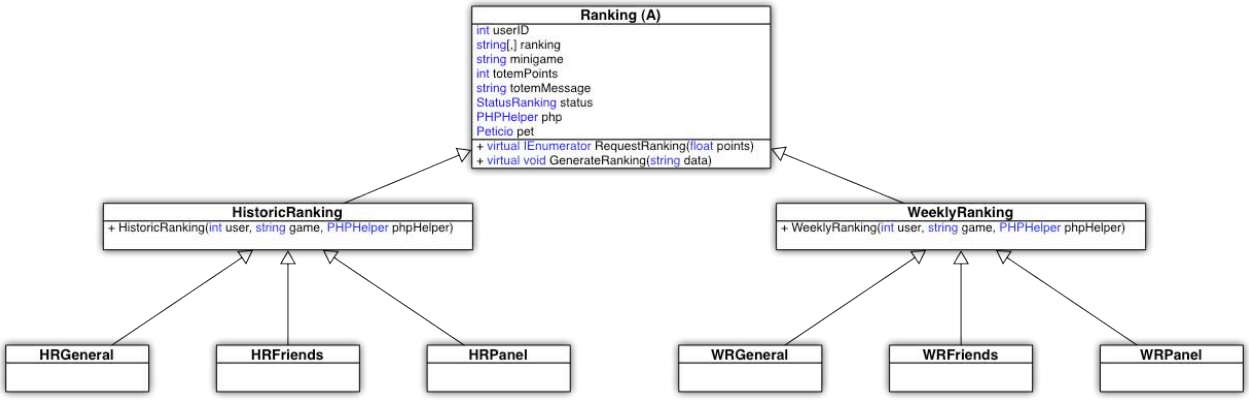
Aquest sistema ha de ser capaç de diferenciar dos tipus de rankings: històric i setmanal; i ha de ser capaç de mostrar diferents rankings segons si es vol la classificació general o es vol la classificació dels amics que tinguis dins la plataforma.

Què ofereix Unity per a la realització del mòdul

Per a aquest mòdul, Unity no ofereix res més que el sistema de GUI, del qual ja hem parlat al punt anterior, per a mostrar per pantalla la taula de resultats del ranking.

Així doncs, s'han hagut de crear unes classes que aconsegueixin complir amb els requeriments d'aquest mòdul.

Diagrama



Il·lustració 11: Diagrama de classes del mòdul del ranking

S'ha creat una estructura jerarquitzada per a cobrir totes les necessitats apuntades anteriorment però unint totes les característiques comunes dels rankings en un únic lloc. D'aquesta manera, s'han creat unes classes més abstractes o més generals, amb les característiques en comú de tots els rankings, i després unes altres classes més específiques, que hereten de les més generals i és en aquestes on es defineixen les diferencies que es necessiten per a cada tipus de ranking. Més concretament, s'han creat les següents classes:

- `Ranking`: classe abstracta. Classe més general de tota l'estructura. Conté les variables comunes per a tots els rankings i les funcions necessàries per a qualsevol ranking que es vulgui crear. Tot i això, les funcions s'han declarat `virtual` per a que puguin ser sobreescrites en classes derivades per si fos necessari un tractament especial dins d'aquestes funcions.
- `HistoricRanking` i `WeeklyRanking`: aquestes dues classes hereten directament de la classe base `Ranking`. Aquestes dues classes s'han creat per a poder tenir la diferència entre els rankings històrics atemporals i els rankings setmanals que cada setmana es resetegen per a donar pas a un nou període amb una nova classificació. La diferència bàsica entre ells serà aquest període de temps que marcarà l'inici o el fi d'un període.
- `HRGeneral`, `WRGeneral`, `HRFriends`, `WRFriends`, `HRPanel` i `WRPanel`: totes aquestes classes deriven de `HistoricRanking` i `WeeklyRanking`. Així doncs, cada un hereta el comportament d'un o altre. A partir d'aquí, cada un defineix si ha de ser un ranking general de tots els usuaris, o un ranking d'amics o un ranking per a ser mostrat en un panells públics per a tothom i que necessiten ser actualitzats cada període de temps.

Aquesta estructura es podria haver evitat i només declarar, com a molt, `Ranking`, `HistoricRanking` i `WeeklyRanking` i hagués funcionat de la mateixa manera. El problema és que amb un mètode com aquest és que no és gens escalable i, per tant, al mínim canvi en comportament o en estructura, implicarà un canvi força gros i requerirà canviar molt codi per a que totes les parts vegin actualitzats els seus comportaments. De la manera proposta en aquest projecte, s'han tingut en compte totes les possibilitats i, de cara al futur, escalar o modificar aquesta estructura implicarà canvis mínims.

Així doncs, treure una classe o afegir-ne una de nova no implicarà cap canvi a la resta de classes ja existents en aquesta estructura, per tant, es pot considerar que aquest sistema ofereix una modularitat força bona.

Editor d'stands

Objectiu del mòdul

L'objectiu d'aquest mòdul és crear un editor per a que els usuaris que així ho hagin contractat, puguin crear i personalitzar el seu espai de negocis, escollint l'estructura externa, els colors corporatius de l'empresa a la qual representa i el mobiliari interior.

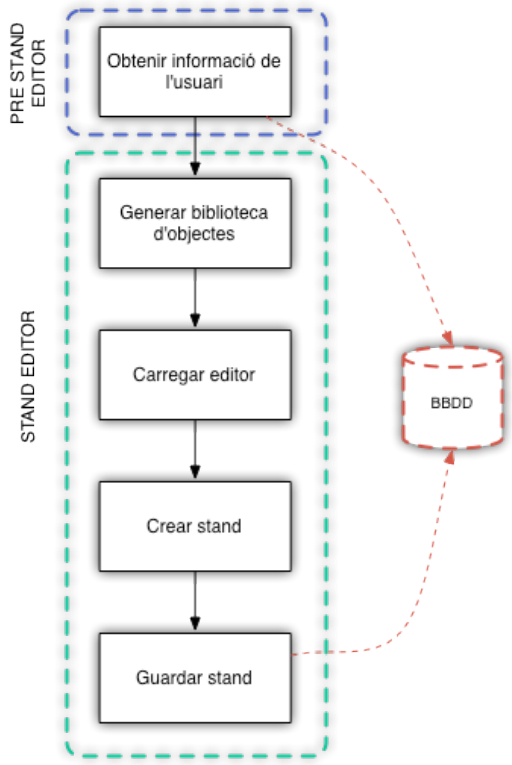
Què ofereix Unity per a la realització del mòdul

Aquest mòdul és un mòdul força gros, pel qual és lògic pensar que Unity no pot aportar una solució íntegra a aquest problema, però si pot aportar eines per a construir-lo. D'aquesta manera, Unity ofereix les següents eines per a la elaboració d'aquest mòdul:

- `Raycast`: mètode fet servir en gairebé qualsevol motor gràfic. Serveix per a llançar raigs des de un punt del món cap a una direcció fins a una certa distància. L'objectiu d'això és comprovar si aquest raig troba algun objecte al seu camí i poder actuar en conseqüència amb aquesta situació.
- `Input`: eina molt útil per a la detecció de interrupcions d'entrada procedents del ratolí o del teclat. És una eina que s'utilitza a gairebé tots els mòduls però que en aquest pren molta importància al tenir molts controls involucrats a l'editor. Segons si s'ha rebut l'event de que s'ha clicat una tecla o una altra, o si s'ha apretat un botó o un altre del ratolí s'haurà de generar una acció o una altra. Així doncs, aquesta eina esdevé imprescindible per a aquest mòdul.
- `Camera`: com en el cas anterior, no és un component únic en aquest mòdul, però que pren especial importància, degut a la necessitat, dins d'aquest mòdul, de poder tenir diferents punts de vista per a poder personalitzar molt millor el nostre stand. Aquest component permet crear càmeres des de les quals es pot visualitzar el que hi ha a escena. Hi ha multitud de paràmetres configurables, tals com *far plane, near plane*, angles d'obertura, escollir entre càmera ortogràfica o no, etc.

Tot i les eines que facilita el motor gràfic, en aquest mòdul ha sigut més important la "creativitat" del programador a l'hora de crear-lo, que no pas les facilitats obtingudes des del motor.

Diagrama



Il·lustració 12: Diagrama de flux de l'editor d'stands

Degut a que hi ha diferents tipus de stands i es poden contractar diferents tipus d'espais, és necessari conèixer d'avançat quin tipus d'usuari està intentant entrar a l'editor, per a oferir-li les possibilitats que ha contractat i no unes altres.

Així doncs, abans d'entrar a carregar l'editor, el primer que es fa es carregar una avantsala on es recopilarà la informació de l'usuari. Un cop el joc ha recopilat la informació procedent de la BBDD, ja pot passar a carregar l'escena de l'editor.

El primer que crea són les biblioteques de mobiliari i estructures que pot fer servir l'usuari. Un cop s'han generat les biblioteques, es carrega la resta de l'editor per a que estigui operatiu. Aquí és on entra la feina de l'usuari per a crear i personalitzar el seu espai de negocis.

Dintre d'aquesta part conviuen petits mòduls que, tots junts, permeten a l'usuari crear el seu stand. A continuació es nombren, a grans trets, aquests petits mòduls i les seves funcions:

- *GUI*: aquest mòdul és l'encarregat d'oferir de manera visual les diferents opcions que té l'usuari, capturar-les i transferir-les als mòduls encarregats de realitzar aquella acció.
- *DragObjects*: és el mòdul encarregat de permetre a l'usuari seleccionar objectes, moure'ls, rotar-los, eliminar-los, etc.
- *Cameras*: mòdul encarregat de permetre l'intercanvi entre una càmera i una altra i apropar-les o allunyar-les.
- *Options*: diferents opcions per a fer més amena i fàcil treballar amb l'editor.

Un cop l'stand ja està muntat i personalitzat al gust de l'usuari, es procedeix a guardar aquesta distribució a BBDD per a que, posteriorment, dins el joc es pugui plasmar l'stand tal i com el seu propietari l'ha deixat. Aquesta part es realitza mitjançant el mòdul d'interconnectivitat amb la BBDD i la distribució de l'stand es guarda mitjançant una estructura XML.

Editor d'avatars

Objectiu del mòdul

L'objectiu d'aquest mòdul és el de crear un editor per a que l'usuari es personalitzi el seu avatar, que és el que utilitzarà dins la plataforma. La personalització ha de consistir en escollir un nick i seleccionar una de les opcions que hi ha per a:

- Cabell
- Cara
- Samarreta
- Pantalons
- Sabates

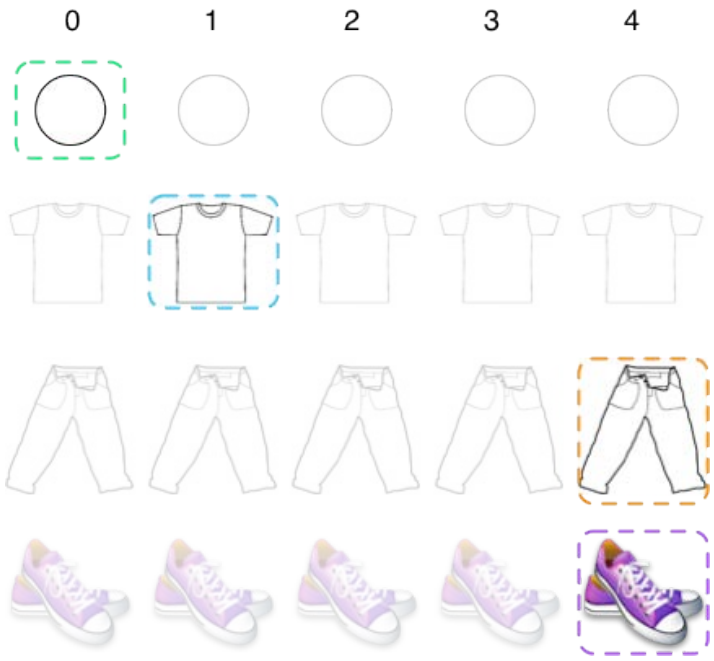
Un cop guardat l'avatar, l'usuari ha de ser capaç d'entrar al món i trobar-se al seu avatar amb la configuració triada a l'editor.

Què ofereix Unity per a la realització del mòdul

De la mateixa manera que a l'editor d'stands, el motor en sí no ofereix cap solució específica per al que es vol aconseguir, però si ofereix eines per a fer més fàcil el desenvolupament del mòdul.

Estructura de l'avatar

Abans d'explicar el funcionament de l'editor, cal fer menció a l'estructura del personatge per a entendre com es varia cada part personalitzable.

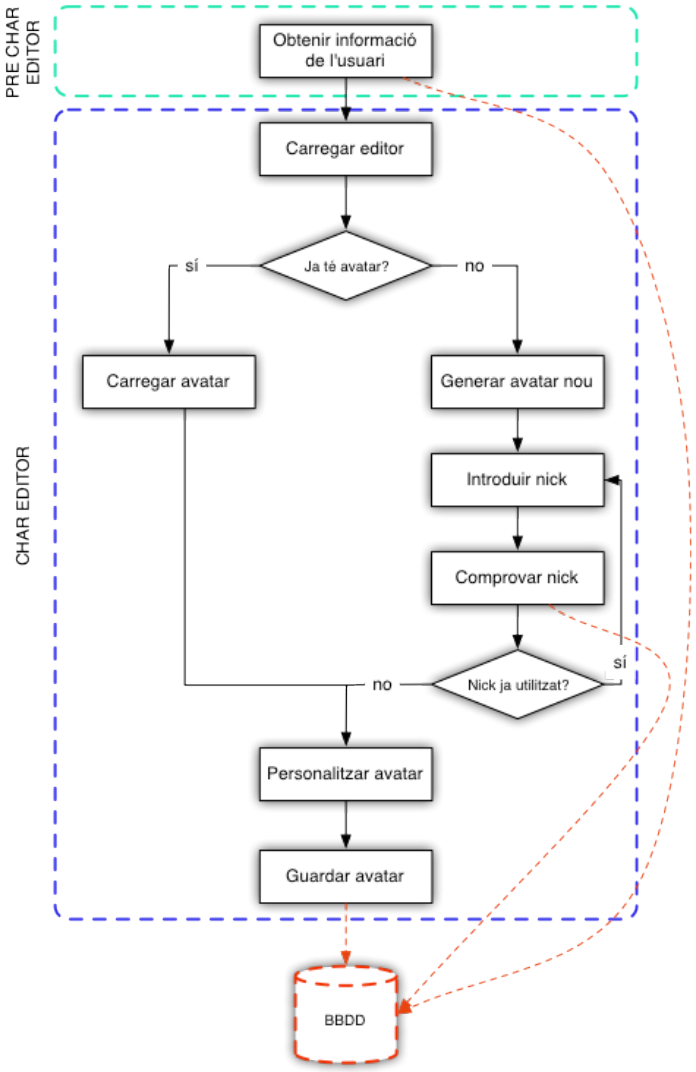


Il·lustració 13: cada part del personatge es configura mitjançant identificadors

Els personatges estan dividits, bàsicament, en 2 models: home i dona. Cada un d'aquests conté a sobre totes les combinacions possibles de roba. Per tant, l'objectiu és insertar cada peça de roba de cada tipus en un vector per a separar per parts. Un cop fet això, és fàcil tenir identificada la peça actual o peça de roba seleccionada, només necessitem un simple identificador que serveixi per a indicar dins el vector quina és la peça que correspon a la seleccionada.

D'aquesta manera, només mostrem la peça a la qual apunta l'identificador i les altres són desactivades pera que no es renderitzin. Per a passar entre una peça i una altra també és molt senzill, ja que consisteix simplement en incrementar o decrementar aquest identificador. Val a dir que hi haurà un identificador per a cada part del personatge que sigui personalitzable. Així doncs, i d'acord amb el que hem indicat anteriorment hi haurà 5 identificadors diferents.

Diagrama



Il·lustració 14: Diagrama de flux de l'editor d'avatars

Com es pot observar al diagrama, s'ha seguit una estructura semblant a la de l'editor d'stands. I és que, com passava en aquell mòdul, necessitem la informació de l'usuari abans de carregar l'editor, ja que pot ser que aquest ja s'hagi creat anteriorment un avatar i simplement vulgui editar-lo.

Així doncs, tenim una escena de avantsala que ens serveix per a recopilar la informació de l'usuari mitjançant l'accés a BBDD. Un cop tenim aquesta informació, segons si ja té avatar o no, carregarem el seu avatar amb l'actual configuració o en generarem un per defecte per a que comenci a editar-se'l.

En cas de no tenir encara avatar, s'habilitarà la possibilitat d'introduir un nick que serà el que el representarà dins el joc. Aquest nick s'haurà de comprovar per a que dos usuaris no tinguin el mateix nick. Un cop ja tenim el nick, ja podem personalitzar el nostre avatar.

Com a qualsevol editor de personatges, tindrem una previsualització de l'avatar per a veure com quedaria l'actual configuració. D'aquesta manera és molt més agradable editar l'avatar.

Tot i que al diagrama apareix la introducció del nick abans que de la personalització de l'avatar, realment es pot realitzar en qualsevol moment. Però fins que no s'hagi comprovat el nick i hagi sigut donat per bo, l'usuari no podrà guardar la configuració. En cas de no ser el primer cop que es configura l'avatar, es pot guardar en qualsevol moment de l'edició.

Un cop ja s'ha acabat d'editar i personalitzar l'avatar, ja es pot guardar la configuració d'aquest a BBDD. Com que les parts personalitzables són sempre les mateixes i no pot créixer, no es necessari generar un XML per a guardar-ho a BBDD, sinó que només s'omplen tants camps com identificadors tenim i guardem cadascun d'aquests a BBDD.

Minijocs

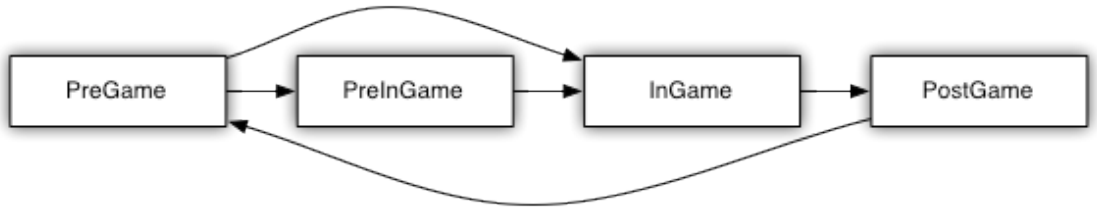
Objectiu del mòdul

L'objectiu d'aquest mòdul és la de crear minijocs per a cobrir la part dedicada a l'oci de la plataforma de la qual forma part aquest projecte. Els usuaris han de poder jugar amb el seu personatge i poder registrar els resultats obtinguts a BBDD per a que, posteriorment, es reflexin al ranking.

Característiques comunes

Tots els minijocs tenen els seus propis objectius i requeriments, però entre ells tenen aspectes semblants o comuns.

- `GameStatus`: enum que marca els estats en què es pot trobar qualsevol dels minijocs. Hi ha definits 4 estats:



Il·lustració 15: possibles estats dins un minijoc

  - `PreGame`: estat inicial del minijoc. En aquest estat el joc encara no està iniciat, per tant, el jugador encara no pot controlar les accions de l'avatar. Serveix normalment per a iniciar variables i estructures necessàries per al joc.
  - `PreInGame`: estat immediatament davant del cos principal del joc. No sempre és necessari passar per aquest estat per a iniciar el minijoc. Sol servir per a configurar aspectes que no podien ser considerats a l'estat anterior ni tampoc al següent. Estat obligatori si el minijoc té compte enrere.

- `InGame`: estat en el que l'usuari ja es troba plenament jugant. És el cos principal del minijoc i és on es dona el control de l'avatar a l'usuari i es comença a controlar les possibles accions d'aquest, com per exemple, contar el nombre de punts que porta, controlar el temps que resta, etc.
- `PostGame`: últim estat del minijoc. S'hi arriba després d'esgotar el temps o arribar a algun objectiu. Serveix per a regsitrar la puntuació a BBDD i recollir la informació del ranking per a mostra-ho per pantalla. També serveix per a donar els premis s'hi s'ha arribat a algun objectiu que doni premis.

- *Ranking*: tots els minijocs han de tenir algun tipus de puntuació que pugui quedar registrada a la BBDD i pugui crear un ranking amb el qual creï competitivitat entre els usuaris de la plataforma. També serveix per a oferir premis en el cas d'assolir o superar uns objectius dins del joc. El ranking sempre té la mateixa estructura: posició, nick i puntuació. Al ranking es mostraran els 10 primers classificats més la marca del propi jugador en el cas de no està inclosa entre els 10 primers i marcada amb diferent color.

POSICIÓ	JUGADOR	PUNTUACIÓ
1	jugador 1	210
2	jugador 3	208
3	jugador 21	200
4	jugador 90	199
5	jugador 21	196
6	jugador 11	189
7	jugador 3	189
8	jugador 7	170
9	jugador 111	155
10	jugador 12	154
51	jugador 10	83

Il·lustració 16: organització del ranking en pantalla

El ranking sempre apareixerà al final del joc (`PostGame`) amb la classificació actualitzada en el cas de que s'hagi superat algun record.

- *Avatar*: com ja s'ha indicat anteriorment, el jugador jugarà amb el seu avatar, per tant, el minijoc ha de fer un procés previ que consisteix a la recopilació de la informació de l'usuari per a "muntar" l'avatar tal i com l'usuari el veu fora dels minijocs. El procediment és força semblant al que es faria a l'editor d'avatars. Dintre del joc hi ha un objecte que guarda permanentment la informació del jugador, així que l'únic que ha de fer el minijoc és accedir a aquest objecte i consultar aquesta informació. Un com ja té la informació només ha de muntar l'avatar.

Què ofereix Unity per a la realització del mòdul

De la mateixa manera que als editors, aquest és un mòdul molt gros com per a que Unity ofereix algun mecanisme específic per a la seva resolució. Per tant, aquí es necessita d'un disseny previ amb uns requisits ben definits i un *gameplay*[12] ben estructurat.

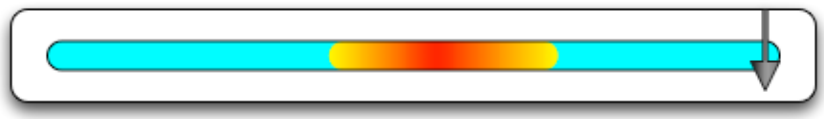
Tot i això, s'han fet servir algunes eines prou significatives del motor que han ajudat a la creació i desenvolupament dels minijocs:

- `Physics`: el motor ofereix un sistema de físiques que permet als objectes dins les escenes actuar afectats per unes físiques que simulen les del món real. D'aquesta manera, es pot llençar una pilota amb una certa força que aquesta es veurà afectada per la gravetat i caurà.
- `Physic Material`: lligat a l'anterior punt, aquests materials serveixen per a donar propietats als objectes i actuar d'una manera o d'una altra amb el contacte amb altres objectes. Així doncs, podem tenir una superfície amb poca fricció, simulant el gel, on l'avatar llisqui sobre ella sense poder frenar ràpidament.
- `Trigger`: aquests components són molt comuns als videojocs. Consisteixen en una àrea en la qual, si un objecte entra o surt, aquest llença un event i es crida una funció. Així doncs, es poden considerar com a "reactors" sobre l'entorn. Per exemple, a l'entrar a un trigger es podria llençar la funció de canvi de nivell perquè s'ha arribat al final de l'actual.
- `Collider`: també relacionat amb les físiques, és un component que permet que els objectes no puguin ser travessats per altres objectes.

Basket

Aquest minijoc consisteix en fer el nombre màxim de punts en 2 minuts encistellant la pilota des de diferents posicions del camp.

Per encistellar ens haurem de guiar per una barrar de precisió que ens marcarà la posició on la probabilitat de fer diana és pràcticament 100%. La barra està divida per colors de la següent manera:



Il·lustració 17: barra de precisió del minijoc de bàsquet

- *Blau*: molt lluny de la zona bona, per tant, pràcticament impossible encistellar.

[12] Jugabilitat d'un joc

- *Taronja*: zona propera a la vermella. Més o menys es té un 50% de probabilitat d'encert.
- *Vermell*: en aquesta zona la probabilitat d'encert és pràcticament del 100%.

Els punts que s'aconsegueixen a l'aconseguir cistella van en funció del nivell en que es trobi en aquell moment l'usuari. S'inicia el joc al nivell 1 i es puja un nivell cada cop que s'encistella i se'n baixa un cada cop que es falla. Per tant, cada cistella valdrà tants punts com el nivell en el que es trobi l'usuari al realitzar la cistella.

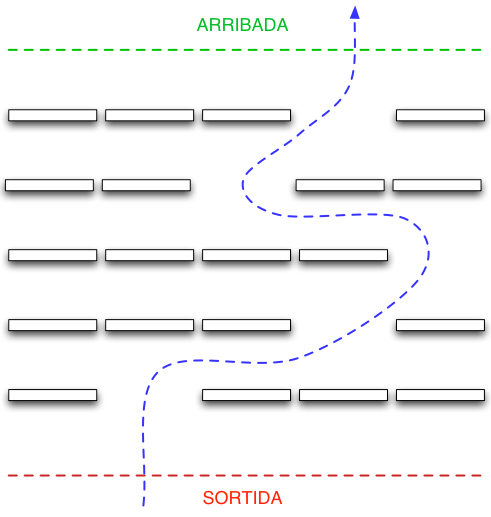
La duració de la partida és de 2 minuts i només durant aquest temps es podrà encistellar. Cada cop que es llança una pilota, hi ha un temps de 5 segons per a donar temps a que la pilota entri. En cas contrari, es calcularà una nova posició i es tornarà a tirar.

Un cop finalitzen els 2 minuts, s'enviarà el resultat a BBDD i es mostrarà el ranking amb els 10 primers classificats més la classificació de l'usuari, resaltada d'un altre color, com s'ha explicat anteriorment.

Cursa

Aquest joc consisteix en una carrera d'obstacles, on l'usuari haurà de passar pels forats lliures i així fins a arribar a la meta. L'usuari haurà d'anar prement la tecla espai per aconseguir velocitat. Contra més ràpid es premi, més velocitat s'assoleix.

El recorregut està format per valles obstaculadores que no permeten el pas. A l'inici de la partida, el joc calcula un camí aleatori, traient una valla per fila. Així doncs, l'usuari haurà de passar per aquests espais per arribar a la meta en el menor temps possible.

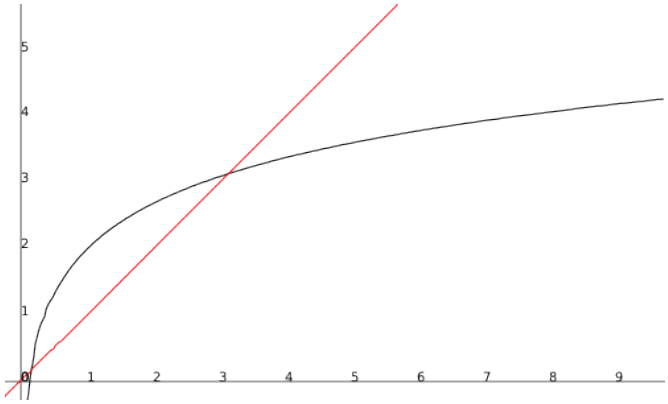


Il·lustració 18: generació de camins dins el minijoc de la cursa

La durada de la partida depèn del que trigui l'usuari en finalitzar el recorregut. L'objectiu és arribar en el menor temps possible.

Per a la velocitat, s'ha realitzat una combinació entre una funció logarítmica per l'augment de la velocitat i una funció lineal per a disminuir la velocitat. D'aquesta manera, s'aconsegueix

que, a l'inici, a velocitats baixes sigui fàcil augmentar la velocitat amb una freqüència de pressions baixa, però que, a l'anar assolint velocitat, costi cada cop més augmentar-la.



Il·lustració 19: la gràfica negra indica com augmenta la velocitat. La gràfica vermella com disminueix

Així, l'usuari, mentre manté el botó dret polsat per a orientar a l'avatar, ha d'anar prement el més ràpidament possible l'espai per a assolir velocitat. La coordinació d'ambdues coses és el que permet aconseguir un bon temps.

Igual que amb els altres jocs, al final s'envia el resultat a BBDD i es mostra el ranking.

Collect

Aquest joc té dos variants que es basen en la mateixa premissa: agafar boles. La primera variant consisteix en agafar el major número de boles, repartides per una escena, en un temps determinat (1 minut). El joc finalitza quan s'ha esgotat el temps o quan s'han agafat totes les boles del mapa. La segona variant consisteix en agafar totes les boles que es troben amagades al mapa en el menor temps possible. El joc finalitza quan s'han agafat totes les boles del mapa.

En aquest joc també es troba present la mecànica de la cursa pel que fa a velocitat de l'avatar. Així doncs, es torna a repetir la coordinació entre mà esquerra i dreta per a marcar un bon temps.

Laberint

Aquest minijoc és l'únic que és de caire online. En aquest joc pots veure a altres persones jugant al mateix temps que tu, pel que apareix un aspecte de col·laboració entre usuaris.

L'objectiu d'aquest joc és completar el laberint en el menor temps possible. Quan s'arriba al final apareix, igual que als altres jocs, el ranking amb els millors temps obtinguts.

Tòtems

A cada minijoc es pot aconseguir uns premis anomenats *tòtems*. Aquests tòtems són un sistema de punts que serveix per a incentivar als usuaris i premiar-los d'alguna manera per jugar. Aquests s'aconsegueixen només a l'assolir una mínima puntuació. Segons quina puntuació és poden donar 0, 1, 2, 3 o 5 tòtems per partida.

Aquests tòtems seran intercanviables per ítems en la futura tenda que s'implementi a la plataforma. Es pot consultar en qualsevol moment la quantitat de tòtems guanyats des de l'inventari.

3. RESULTATS

A continuació es mostraran imatges identificatives de cada un dels mòduls realitzats, per a comprovar com ha quedat estèticament cada un d'ells.

Editor d'avatars

Il·lustració 20: editor d'avatars on l'usuari esta editant el seu personatge

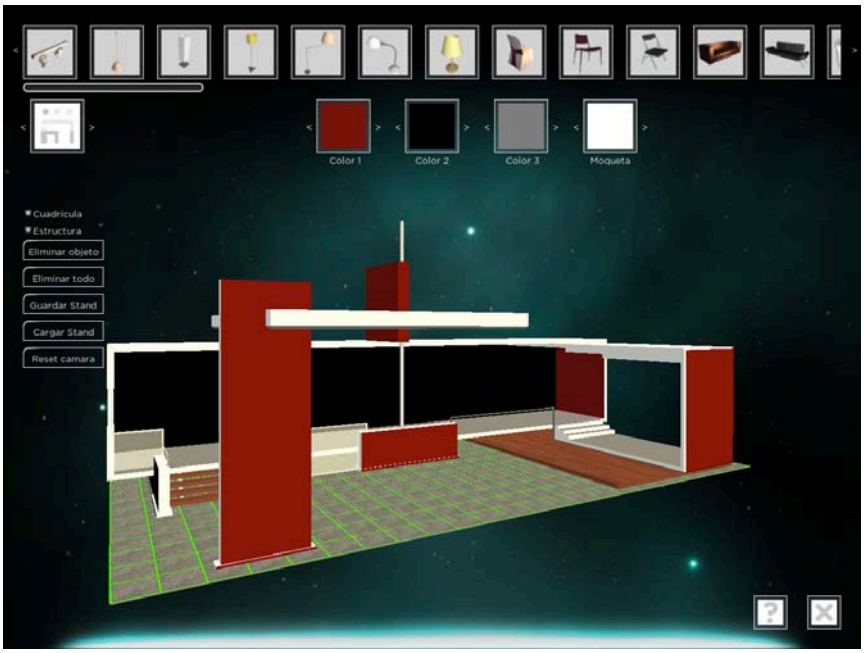
Aquí es pot observar l'editor d'avatars, amb la previsualització de com quedaria l'avatar amb la configuració actual i a la dreta la interfície que permet canviar la configuració.

També es pot observar el nick a sobre del personatge. Això indica que l'usuari ja tenia avatar configurat i que l'està editant. Només apareixerà el nick a sobre en aquesta situació, mai quan s'està configurant el personatge per primer cop.

Il·lustració 21: editor d'avatars. L'usuari està creant l'avatar per primer cop

En aquest cas, ens trobem en el mateix editor d'avatars, però ara en la situació de la configuració de l'avatar per primer cop. Podem observar com ha aparegut un camp per introduir un nick (àlies) i comprovar la seva disponibilitat. També es pot observar que no hi ha botó "Guardar". Això és degut a, com s'ha esmentat al diagrama corresponent, no es pot guardar fins que no es té un nick vàlid.

Editor d’stands



Il·lustració 22: editor d'stands. Aspecte inicial.

Aquí tenim la pantalla inicial de l’editor d’stands, un cop ja s’han recopilat les dades relatives a l’usuari. Com es pot observar a la imatge, trobem a la part de dalt els objectes de la biblioteca que l’usuari pot fer servir, així com també els colors que es poden escollir per a l’estructura.

A la part esquerra trobem unes quantes opcions útils per a la personalització de l’stand.

Il·lustració 23: editor d'stands. Aspecte final.

Aquí podem observar l'aspecte final de l'stand després de configurar-lo. Ara ja només quedaria guardar l'stand mitjançant la opció disponible a la part esquerra i ja estaria llest.

Un exemple de com es guardaria la informació a base de dades el trobem a continuació:

```
<?xml version="1.0" encoding="UTF-8"?>

<usuari id="158">
   <stand tipus="mixto" id="4">
      <estructura tipus="StandMixt1">
         <color id="26" R="0,4509804" G="0,07058824" B="0,07058824" A="1" />
         <color id="0" R="0" G="0" B="0" A="1" />
         <color id="10" R="0,5058824" G="0,5019608" B="0,5058824" A="1" />
         <color id="28" R="1" G="1" B="1" A="1" />
         <objectes>
            <object nom="gLightTable001" x="2,25" y="0,3006491" z="2,75" rx="0" ry="0" rz="0" />
            <object nom="gCouch001" x="6,5" y="0,4600797" z="4,25" rx="0" ry="135" rz="0" />
         </objectes>
      </estructura>
   </stand>
</usuari>
```

Així doncs, quan es necessites recuperar la configuració de l'stand, s'hauria d'analitzar aquesta estructura XML i interpretar-la.

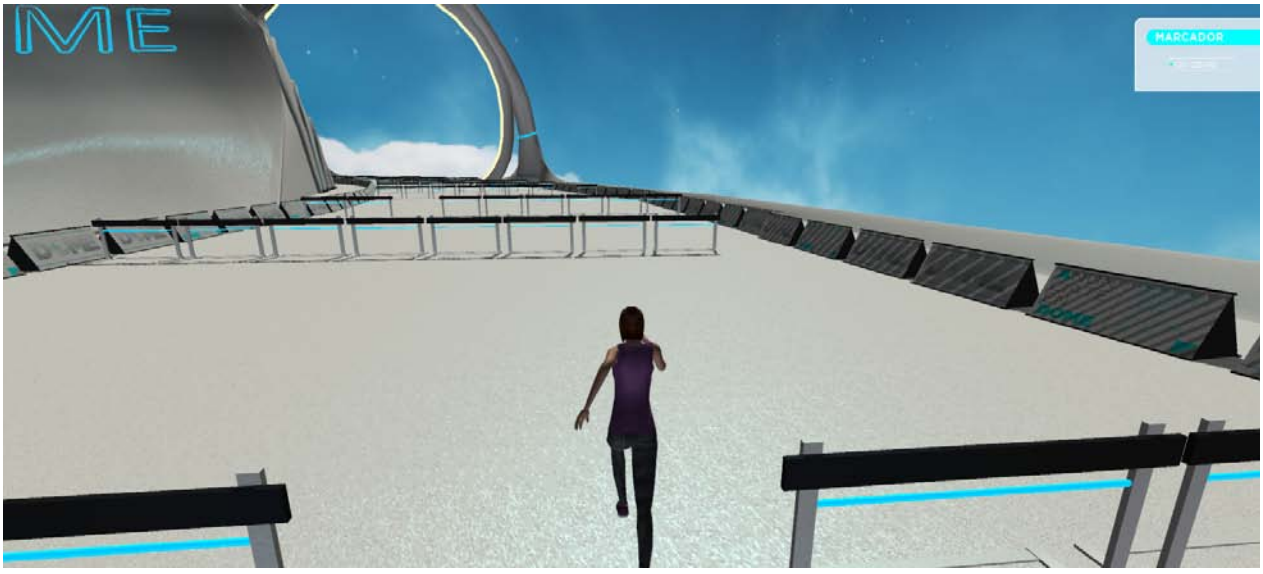
Minijocs

Il·lustració 24: bàsquet. Moment en el que el personatge està llançant a cistella

Aquí ens trobem jugant al minijoc de bàsquet. Podem observar a la part de dalt la barra de precisió que indica la probabilitat d'encert. A la part esquerra, també podem trobar, mitjançant interfície gràfica, el temps restant, el punts aconseguits fins al moment i el nivell actual de joc.

Ilustración 25: bàsquet. Final de la partida

Aquí també ens trobem al minijoc de bàsquet, concretament al final del temps de joc. Un cop s'acaba el temps se'ns mostra el ranking amb el nostre rècord ressaltat. També podem observar com, donada la puntuació obtinguda (recordem, la puntuació es troba a la part dreta de la pantalla) hem rebut com a premi una quantitat de tòtems.



Il·lustració 26: Cursa. L'avatar es troba corrent entre les valles

Ara ens trobem al minijoc de la cursa. Es pot observar com segueix el mateix disseny d'interfície i podem trobar el marcador (en aquest cas mostrant el temps que ha passat).

Il·lustració 27: Laberint. Realitzant el recorregut

Ara estem situats al minijoc de laberint. Com que aquest és un joc online, tenim totes les possibilitats disponibles, com pot ser el xat, les opcions, la llista d'usuaris, etc.

Així doncs, mentre es juga, es pot parlar amb altres persones que també estiguin en aquell moment jugant (on entra en escena el concepte de joc col·laboratiu).

Ilustración 28: Collect. L'usuari està recol·lectant les boles.

Per últim, tenim el joc de recol·lectar boles. Al xocar amb una bola, aquesta desapareix, emetent un efecte de so i deixant anar unes partícules per a poder identificar que s'ha agafat una bola.

Segueix la mateixa estructura que tots els altres minijocs, així que podem trobar la informació del joc a la part de dalt a l'esquerra.

Il·lustració 29: vista general de la plataforma

En aquesta il·lustració podem trobar una vista general de la plataforma. Posem observar la interfície general del joc. Aquesta consisteix en:

- Nick de l'usuari
- Chat
- Panell de control: conté diferents tipus d'opcions genèriques.

Il·lustració 30: vista de l'inventari

Aquí podem observar el disseny del *frontend* de l'inventari desenvolupat en aquest projecte. Podem veure com consisteix en una graella d'espais, els quals els que estan ocupats, tenen una icona representativa i indica la quantitat d'aquell objecte.

Sota la graella hi ha espai per a indicar la quantitat de monedes que té l'usuari. L'inventari es mostra o s'amaga des del panell de control.

Ilustración 31: vista de la configuració d'àudio

Per últim, podem veure la finestra d'opcions del joc, on es troben les opcions d'àudio. Des d'aquí es poden configurar els nivells de melodia i efectes de so. Quan aquests valors són modificats, canvia l'estat del manager i avisa a tots els seus observadors.

Les opcions del joc també són accessibles des del panell de control general del joc.

4. CONCLUSIONS I MILLORES

Un cop realitzat aquest projecte i havent realitzat tots els mòduls requerits, podem extreure unes quantes conclusions i poder fer una mica d'autocrítica respecte als resultats obtinguts. Així doncs, podem concloure després de realitzar aquest projecte que:

- El motor gràfic Unity3D ofereix una API extensa i diversa que permet realitzar el *gameplay* de videojocs sense preocupar-se de les capes baixes d'aquest.
- L'ús del llenguatge C# com a llenguatge scripting ha sigut molt satisfactori, al beneficiar-se de ser un llenguatge OO[13] i al poder ampliar les possibilitats del joc gràcies a l'ús de les pròpies classes de C#.
- La programació dins el món dels videojocs requereix l'ús de mètodes i dissenys pensats per al rendiment, ja que el temps de resposta d'aquestes aplicacions és un tema crític.
- El món dels jocs online dóna un ventall molt ampli amb el qual jugar. Permet orientar els jocs en qualsevol àmbit, ja sigui més en l'àmbit professional o laboral, o l'àmbit lúdic o social.
- Realitzar un joc online sense instal·lacions requereix la minimització de les càrregues per a evitar avorriment o frustració per part de l'usuari que intenta jugar.
- La realització d'aquest projecte ens ha ofert una bona quantitat de coneixements sobre el desenvolupament de videojocs, ja no tan sols en l'àmbit de programació, sinó en temes de modelat, so o disseny d'interfícies.
- Un bon disseny d'un joc, per molt petit i senzill que sigui, garanteix una millor acceptació per part dels usuaris que hi juguen. Per tant, el disseny ha de ser una etapa molt important a l'hora de realitzar un videojoc.
- Realitzar una acurada interfície d'usuari garanteix que l'usuari es senti més integrat amb el joc i no es "perdi".
- Treballar al món dels videojocs requereix de molt personal, molt de temps i molt d'esforç. Tot l'equip ha d'estar ben complementat per assegurar un bon producte.

Al ser un videojoc online sense instal·lacions, és més fàcil crear actualitzacions i millorar la plataforma amb més contingut o millorar el ja existent. Per tant, després de realitzar el projecte, hi ha una sèrie de millores que es podrien aplicar a joc:

- Millorar el sistema de personalització dels avatars. L'actual sistema és senzill i ràpid d'utilitzar, però oferirà problemes de cara a una possible ampliació. A més, el fet de portar totes les combinacions a sobre, fa que requereixi més memòria. L'objectiu seria

[13] Orientat a Objectes

crear uns personatges capaços de carregar *on demand*[14] les peces que necessiti i que només en portés les actuals.

- Millorar l'editor d'stands per a ser més amigable amb l'usuari, millorant la interacció entre els dos. També s'hauria de millorar la càrrega d'aquest editor, ja que ara mateix necessita carregar tots els objectes de la biblioteca. La millora passaria per a realitzar, com es proposa també a l'editor d'avatars, una petició *on demand* del mobiliari. Així la càrrega inicial de l'editor seria mínima.
- Realització del *backend* de l'inventari per a que sigui possible agafar objectes, deixar-los anar o fer accions amb els objectes.
- Millora en el sistema de creació de camins del minijoc de la cursa. Evitar que hi hagin camins molt rectes i camins molt revirats. Una possible solució seria marcar un interval de metres a recórrer i que el camí s'hagi d'ajustar a aquest varem.
- Creació de variants en el minijoc del laberint. Oferir variants com: buscar la clau per poder sortir o buscar el robot i atrapar-lo. D'aquesta manera es tornaria un joc més atractiu per a l'usuari.
- Realització d'algun minijoc més a per donar més varietat a la part lúdica de la plataforma. Un possible minijoc seria un minigolf.

Hi pot haver més millores, ja que tal i com està definit la plataforma, no necessàriament ha de tenir un "final", per la qual cosa es pot anar ampliant i millorant constantment.

De manera autocrítica, creiem que aquest projecte ha requerit un esforç molt gran, però que els resultats han sigut molt positius, tan de cara a l'usuari com de cara a l'enriquiment cultural que hem obtingut desenvolupant-ho. Els resultats creiem que són bons i que, amb temps i més gent involucrada, es pot aconseguir un videojoc de gran mida i qualitat.

[14] Sota demanda

5. BIBLIOGRAFIA

1. **ADESE.** Asociación Española de Distribuidores y Editores de Software de Entretenimiento. [En línia] [Data: 08 / 06 / 2010.] http://www.adese.es/pdf/dossier_prensa%20_balance_economico_2009.pdf.

2. **Martín, Ignacio G.** N-Economia. [En línia] [Data: 8 / Junio / 2010.] http://www.n-economia.com/notas_alerta/pdf/ALERTA_NE_17-2005.PDF.

3. **Unity3D.** Unity3D - Scripting Reference. [Online] [Cited: Junio 10, 2010.] http://unity3d.com/support/documentation/ScriptReference/index.html.

ANNEXOS

Annex 1: Manual d'usuari

A continuació es donaran les instruccions bàsiques per a entrar a jugar a la plataforma:

1. Entrar a la pàgina web i anar al registre de nou usuari.
2. S'obrirà l'editor d'avatars. Personalitzar l'avatar i buscar nick correcte. Un cop fet això, prémer *Siguiente*.
3. Acabar d'omplir les dades de registre.
4. Confirmar el mail enviat al finalitzar el registre.
5. Tornar a la pàgina i prémer el botó *Entrar en DOME*.
6. Esperar a que es carregui el joc.
7. Un cop carregat, ja estareu dins la plataforma.

Els controls del personatge són els següents:

- *Tecles de cursor i/o W A S D*: permeten moure l'avatar en totes les direccions.
- *Botó esquerra*: permet moure la càmera sense moure la orientació del personatge. També serveix per a seleccionar altres usuaris de la plataforma.
- *Botó dret*: permet moure la càmera, fent que l'avatar miri cap allà on s'apunta.
- *F*: intercanvi entre caminar i córrer.
- *R*: activa i/o desactiva el córrer automàtic.
- *Control*: permet la preselecció d'usuaris.
- *Botó dret i esquerra alhora*: permet córrer a l'avatar.

Altres controls existents només en alguna part en concret del joc:

Bàsquet

- *Espai*: llançament de la pilota.

Cursa i collect

- *Espai*: permet augmentar la velocitat de moviment. S'ha de prémer contínuament per a assolir una velocitat alta.

Per a jugar als minijocs s'ha d'entrar a la plataforma i apropar-se a una de les zones on hi ha minijocs. Cadascú tindrà un espai representatiu. L'usuari ha d'apropar-se i prémer al botó *Entrar* que apareixerà per pantalla. En qualsevol moment del minijoc es pot tornar a l'escena principal i abandonar el joc actual.

Annex 2: codi font

A mode de demostració, s'adjunta el codi font de dos dels mòduls realitzats per a veure com han sigut desenvolupats.

Control d'àudio

AudioInfo.cs

```
using UnityEngine;
using System.Collections;

/// <summary>
/// Necessitem un AudioSource al GameObject per a que funcioni correctament
/// </summary>
[RequireComponent(typeof(AudioSource))]

public class AudioInfo : MonoBehaviour
{
      /// <summary>
      /// Variable booleana que indica si el clip d'àudio és efecte o melodia
      /// </summary>
    public bool isFX;

    /// <summary>
    /// Volum inicial de l'AudioSource
    /// </summary>
      float lastVolume;

    /// <summary>
    /// Funció que es crida quan es crea el GameObject.
    /// Guardem el volum per a fer-lo servir per multiplicar
    /// pel factor posteriorment
    /// </summary>
      void Start()
      {
            lastVolume = transform.audio.volume;
      }

    /// <summary>
    /// Aquesta funció és cridada quan es produeix un canvi
    /// al manager.
    /// Modifica el volum d'aquest AudioSource
    /// </summary>
    /// <param name="volumes">
    /// Vector amb els factors de multiplicació.
    /// -volumes.x fa referència al factor dels efectes de so.
    /// -volumes.y fa referència al factor de les melodies.
    /// </param>
      void OnChangeVolume(Vector2 volumes)
      {
            transform.audio.volume = isFX ? lastVolume * volumes.y : lastVolume *
volumes.x ;
      }
}
```

AudioManager.cs

```
using UnityEngine;
using System.Collections;
using System.Collections.Generic;

public class AudioManager : MonoBehaviour
{
    /// <summary>
    /// Array d'observadors. En aquest cas, AudioSource's.
    /// </summary>
      AudioSource[] audios;

    /// <summary>
    /// Variables necessària per a la part de servidor.
    /// Guarda informació relacionada amb l'usuari i la
    /// connexió.
    /// </summary>
      DataHolder holder = null;

    /// <summary>
    /// Flag de control que serveix per a executar una
    /// funció un únic cop.
    /// </summary>
      bool m_bOnce = true;


    /// <summary>
    /// Funció que es crida quan es crea el GameObject.
    /// Recopila tots els AudioSource de l'escena per a
    /// tenir-los com a observadors d'ell mateix.
    /// També busca el GameObject que conté la informació
    /// del servidor.
    /// </summary>
      void Start()
    {
            audios = Tools.FindObjectsOfType<AudioSource>();

            GameObject Dh = GameObject.FindGameObjectWithTag("dataholder");
            if (Dh != null)
            {
                  holder = Dh.GetComponent<DataHolder>();
                  if (holder == null)
                  {
                        Debug.LogError("No se ha encontrado el componente
'DataHolder' en AudioManager");
                  }
                  else
                  {

                        Debug.Log("Volume Music: " + holder.VolumeMusic + " Volume
FX: " + holder.VolumeFx);
                  }
            }
            else
            {
                  Debug.LogError("No se ha encontrado el objeto 'DataHolder' en
AudioManager");
            }
```

```
    }
    /// <summary>
    /// Aquesta funció es crida un cop per a cada
    /// frame. Només servirà per, al primer cop,
    /// avisar del canvi d'estat del manager.
    /// </summary>
    void Update()
    {
        if (m_bOnce)
        {
            ChangeVolume(holder.VolumeMusic, holder.VolumeFx);
            m_bOnce = false;
        }
    }

    /// <summary>
    /// Funció encarregada de recórrer tots els
    /// observadors i avisar-los del canvi d'estat.
    /// </summary>
    /// <param name="inMusic">
    /// Factor de multiplicació dels clips de melodia.
    /// </param>
    /// <param name="inFX">
    /// Factor de multiplicació dels efectes de so.
    /// </param>
    public void ChangeVolume(float inMusic, float inFX)
    {
        Vector2 v = new Vector2(inMusic, inFX);
        foreach (AudioSource audio in audios)
        {
            audio.gameObject.SendMessage("OnChangeVolume", v,
SendMessageOptions.DontRequireReceiver);
        }
    }


    /// <summary>
    /// Funció útil mentre s'està a l'editor.
    /// Mostra una icona representativa a l'escena
    /// per a poder identificar ràpidament el GameObject.
    /// </summary>
    void OnDrawGizmos()
    {
        Gizmos.DrawIcon(this.transform.position, "audiomang.png");
    }

}
```

FrontEnd inventari

ObjectInfo.cs

```
using UnityEngine;
using System.Collections;

public class ObjectInfo : MonoBehaviour
{
    /// <summary>
    /// Icona representativa de l'objecte de l'inventari.
    /// </summary>
    public Texture iconTexture;

    /// <summary>
    /// Nom de l'objecte.
    /// </summary>
    public string nameOfObj = "Put your Object name";


    /// <summary>
    /// Funció que s'executa quan es crea el GameObject.
    /// Per al frontend no es necessari realitzar res en
    /// aquesta funció.
    /// </summary>
      void Start ()
      {

      }
}
```

PositionController.cs

```
using UnityEngine;
using System.Collections;
using System.Xml;
using System.Xml.Serialization;

public class PositionController : MonoBehaviour
{
    /// <summary>
    /// Array amb els objectes que es troben a
    /// l'inventari.
    /// </summary>
    public GameObject[] objects;

    /// <summary>
    /// Array amb el nombre d'elements de cada
    /// objecte de l'inventari.
    /// </summary>
    public int[] numObjects;

    /// <summary>
    /// Nombre de monedes del tipus 1.
    /// </summary>
    int monedas1 = 0;
```

```
    /// <summary>
    /// Nombre de monedes del tipus 2.
    /// </summary>
    int monedas2 = 20;

    /// <summary>
    /// Component encarregat de la comunicació
    /// amb BBDD.
    /// </summary>
    public PHPHelper php;

    /// <summary>
    /// Component relacionat amb les connexions
    /// al servidor.
    /// </summary>
      DataHolder m_DHolder = null;


    /// <summary>
    /// Funció que s'executa només quan es crea
    /// el GameObject que conté aquest script.
    /// Busca l'objecte amb la informació del servidor
    /// per a guardar-se una referència a aquest.
    /// </summary>
    void Start()
    {
            GameObject Dh = GameObject.FindGameObjectWithTag("dataholder");
            if (Dh != null)
            {
                  m_DHolder = Dh.GetComponent<DataHolder>();
                  if (m_DHolder == null)
                  {
                        Debug.LogError("No se ha encontrado el componente
'DataHolder' necesario para Inventory");
                  }
            }
            else
            {
                  Debug.LogError("No se ha encontrado el objeto 'DataHolder'
necesario para Inventory");
            }
    }

    /// <summary>
    /// Propietat per a la variable monedas1.
    /// </summary>
    public int Monedas1
    {
        get { return monedas1; }
        set { monedas1 = value; }
    }

    /// <summary>
    /// Propietat per a la variable monedas2.
    /// </summary>
    public int Monedas2
    {
        get { return monedas2; }
        set { monedas1 = value; }
```

```
}

/// <summary>
/// Funció que retorna l'objecte situat a la
/// posició i de l'inventari.
/// </summary>
/// <param name="i">
/// Posició dins l'inventari.
/// </param>
/// <returns>
/// L'objecte dins la posició indicada.
/// </returns>
public GameObject GetObject(int i)
{
    return objects[i];
}

/// <summary>
/// Funció que retorna la quantitat
/// que hi ha d'un element a l'inventari
/// </summary>
/// <param name="i">
/// Posició dins l'inventari.
/// </param>
/// <returns>
/// La quantitat de l'objecte indicat.
/// </returns>
public int GetNumObjects(int i)
{
    return numObjects[i];
}

/// <summary>
/// Funció que afegeix un objecte nou
/// a la primera posició lliure de l'inventari.
/// </summary>
/// <param name="ob">
/// Objecte que es vol guardar.
/// </param>
public void AddObject(GameObject ob)
{
    for (int i = 0; i < objects.Length; i++)
    {
        if (objects[i] == null)
        {
            objects[i] = ob;
            break;
        }
    }
}

/// <summary>
/// Funció que genera l'estructura XML
/// de l'inventari per a guardar-lo a BBDD.
/// </summary>
/// <param name="idUser">
/// Idetificador d'usuari.
/// </param>
public void GenerateXmlOfInventory(int idUser)
{
    XmlDocument xmlDoc = new XmlDocument();
```

```
XmlNode xmlNode = xmlDoc.CreateXmlDeclaration("1.0", "UTF-8", null);
xmlDoc.AppendChild(xmlNode);

//<inventario>
XmlNode inventariNode = xmlDoc.CreateElement("inventario");

//<item>
for (int i = 0; i < objects.Length; i++)
    {
    if (objects[i] != null)
    {
        XmlNode itemNode = xmlDoc.CreateElement("item");

        XmlAttribute itemPos = xmlDoc.CreateAttribute("pos");
        XmlAttribute itemName = xmlDoc.CreateAttribute("nombre");
        XmlAttribute itemNum = xmlDoc.CreateAttribute("cantidad");

        itemPos.Value = i.ToString();
        itemName.Value = objects[i].name;
        itemNum.Value = numObjects[i].ToString();

        itemNode.Attributes.Append(itemPos);
        itemNode.Attributes.Append(itemName);
        itemNode.Attributes.Append(itemNum);

        inventariNode.AppendChild(itemNode);
    }


    }

//<moneda>
XmlNode coinNode1 = xmlDoc.CreateElement("moneda");
XmlAttribute coinType = xmlDoc.CreateAttribute("tipo");
XmlAttribute coinNum = xmlDoc.CreateAttribute("cantidad");

coinType.Value = "1";
coinNum.Value = monedas1.ToString();

coinNode1.Attributes.Append(coinType);
coinNode1.Attributes.Append(coinNum);

XmlNode coinNode2 = xmlDoc.CreateElement("moneda");
XmlAttribute coinType2 = xmlDoc.CreateAttribute("tipo");
XmlAttribute coinNum2 = xmlDoc.CreateAttribute("cantidad");

coinType2.Value = "1";
coinNum2.Value = monedas2.ToString();

coinNode2.Attributes.Append(coinType2);
coinNode2.Attributes.Append(coinNum2);

inventariNode.AppendChild(coinNode1);
inventariNode.AppendChild(coinNode2);

xmlDoc.AppendChild(inventariNode);


//formulari
WWWForm form = new WWWForm();
form.AddField("id_usuario", m_DHolder.PlayerID.ToString());
```

```
        form.AddField("xml", xmlDoc.OuterXml);
        Debug.Log(xmlDoc.OuterXml);
        php.Peticio(Peticio.Guardar_inventario, form);
    }

    /// <summary>
    /// Funció que crida a BBDD per a que retorni
    /// l'inventari de l'usuari.
    /// </summary>
    /// <param name="idUser">
    /// Identificador de l'usuari.
    /// </param>
    public void LoadObjectsFromBBDD(int idUser)
    {
        WWWForm form = new WWWForm();
            form.AddField("id_usuario", m_DHolder.PlayerID.ToString());
        php.Peticio(Peticio.Abrir_inventario, form, OrganizeInventory);
    }

    /// <summary>
    /// Interpreta la informació obtinguda de BBDD
    /// i genera l'inventari.
    /// </summary>
    /// <param name="data">
    /// Estructura XML amb tota la informació de l'inventari.
    /// </param>
    void OrganizeInventory(string data)
    {
        XmlDocument xmlDoc = new XmlDocument();
            Debug.Log("Inventary XML: " + data);
        xmlDoc.LoadXml(data);
        XmlNodeList itemsList = xmlDoc.GetElementsByTagName("item");
        foreach (XmlNode item in itemsList)
        {
            int i = int.Parse(item.Attributes["pos"].Value);
            int n = int.Parse(item.Attributes["cantidad"].Value);
            GameObject g =
(GameObject)Resources.Load(item.Attributes["nombre"].Value);
            objects[i] = g;
            numObjects[i] = n;
        }
    }
}
```

InventariManager.cs

```
using UnityEngine;
using System.Collections;

public class InventariManager : MonoBehaviour
{
    /// <summary>
    /// Referència al component PositionController
    /// </summary>
    public PositionController pController;

    /// <summary>
    /// Skin de la interfície
    /// </summary>
    public GUISkin skin;
```

```
/// <summary>
/// Style utilitzat a la interfície
/// </summary>
public GUIStyle style;

/// <summary>
/// Style utilitzat a la interfície
/// </summary>
public GUIStyle styleLabel;

/// <summary>
/// Flag per a mostrar o ocultar l'inventari
/// </summary>
bool showInventory;

/// <summary>
/// Mides de l'inventari.
/// </summary>
Vector2 sizeInventory = new Vector2(250, 200);

/// <summary>
/// Posició de l'inventari
/// </summary>
Vector2 posInventory = new Vector2(260, 210);

/// <summary>
/// Espai de marge entre els espais de
/// l'inventari.
/// </summary>
Vector2 spaceBetweenButtons = new Vector2(5, 5);

/// <summary>
/// Mida dels espais de l'inventari.
/// </summary>
Vector2 sizeButton = new Vector2(40,40);

/// <summary>
/// Número de files que té l'inventari.
/// </summary>
int numRows = 3;

/// <summary>
/// Número d'espais per fila.
/// </summary>
public int numButtonsPerRow;

/// <summary>
/// Textura utilitzada a l'inventari
/// </summary>
Texture tmpTexture;

/// <summary>
/// Component relacionat amb les connexions
/// al servidor.
/// </summary>
  DataHolder m_DHolder = null;

/// <summary>
/// Retorna el rectangle on es trobarà l'inventari
/// i les mides que farà.
```

```
    /// </summary>
    /// <returns>
    /// Rectangle amb les mides.
    /// </returns>
    public Rect GetRectInventory()
    {
        return new Rect(Screen.width - posInventory.x, Screen.height -
posInventory.y, sizeInventory.x, sizeInventory.y);
    }

    /// <summary>
    /// Propietat per a la variable
    /// showInvetory.
    /// </summary>
    public bool ShowInventory
    {
        get { return showInventory;}
        set {showInventory = value;}
    }

    /// <summary>
    /// Funció que es crida quan es crea el
    /// GameObject.
    /// Inicialitza el flag de mostrar l'inventaria
    /// false i calcula la mida dels botons.
    /// Després carrega de BBDD l'inventari de
    /// l'usuari.
    /// </summary>
      void Start ()
    {
            GameObject Dh = GameObject.FindGameObjectWithTag("dataholder");
            if (Dh != null)
            {
                  m_DHolder = Dh.GetComponent<DataHolder>();
                  if (m_DHolder == null)
                  {
                        Debug.LogError("No se ha encontrado el componente
'DataHolder' necesario para Inventory");
                  }
            }
            else
            {
                  Debug.LogError("No se ha encontrado el objeto 'DataHolder'
necesario para Inventory");
            }

        showInventory = false;
        CalculateButtonSize();
        LoadInventory(m_DHolder.PlayerID);

      }


    /// <summary>
    /// Funció dedicada exclusivament a la
    /// interfície.
    /// És on es dibuixa l'inventari.
    /// </summary>
    void OnGUI()
    {
        GUI.skin = skin;
```

```
        if (showInventory)
        {
            GUI.BeginGroup(new Rect(Screen.width - posInventory.x, Screen.height -
posInventory.y, sizeInventory.x, sizeInventory.y));
                GUI.Box(new Rect(0, 0, sizeInventory.x,
sizeInventory.y),"Inventario");
                for (int i = 0; i < numRows; i++)
                {
                    for (int j = 0; j < numButtonsPerRow; j++)
                    {
                        if (pController.GetObject(i * numButtonsPerRow + j) ==
null)
                        {
                            GUI.Button(new Rect(spaceBetweenButtons.x * (j + 1) +
j * sizeButton.x, 20 + spaceBetweenButtons.y * (i + 1) + i * sizeButton.y,
sizeButton.x, sizeButton.y), "");
                        }
                        else
                        {
                            tmpTexture = pController.GetObject(i *
numButtonsPerRow + j).GetComponent<ObjectInfo>().iconTexture;
                            GUI.Button(new Rect(spaceBetweenButtons.x * (j + 1) +
j * sizeButton.x, 20 + spaceBetweenButtons.y * (i + 1) + i * sizeButton.y,
sizeButton.x, sizeButton.y), tmpTexture, style );
                            int n = pController.GetNumObjects(i * numButtonsPerRow
+ j);
                            if (n != 1)
                            {
                                float offset = 0.7f;
                                if (n > 9)
                                {
                                    offset = 0.5f;
                                    if (n > 99)
                                    {
                                        offset = 0.3f;
                                    }
                                }
                                GUI.Label(new Rect((spaceBetweenButtons.x * (j +
1) + j * sizeButton.x) + offset * sizeButton.x, (20 + spaceBetweenButtons.y * (i +
1) + i * sizeButton.y) + 0.6f * sizeButton.y, 30, 20), n.ToString(), styleLabel);
                            }

                        }

                    }
                }
                GUI.Label(new Rect(10,sizeInventory.y-25,100,20),"wolvis 1:
"+pController.Monedas1);
                GUI.Label(new Rect(sizeInventory.x * 0.5f, sizeInventory.y - 25,
100, 20), "wolvis 2: " + pController.Monedas2);
            GUI.EndGroup();
        }
    }

    /// <summary>
    /// Funció que calcula la mida dels espais
    /// de l'inventari.
    /// </summary>
    void CalculateButtonSize()
    {
        float totalSpaces = (numButtonsPerRow + 1);
```

```
            float pixelsForSpaces = totalSpaces * spaceBetweenButtons.x;
            sizeButton.x = (sizeInventory.x - pixelsForSpaces) / numButtonsPerRow;
            sizeButton.y = sizeButton.x;
        }

        /// <summary>
        /// Retorna el nombre d'espais que té
        /// l'inventari.
        /// </summary>
        /// <returns>
        /// Quantitat d'espais.
        /// </returns>
        public int GetTotalSlots()
        {
            return numButtonsPerRow * numRows;
        }

        /// <summary>
        /// Funció que carrega la informació de
        /// l'inventari de BBDD.
        /// </summary>
        /// <param name="idUser">
        /// identificació d'usuari.
        /// </param>
        void LoadInventory(int idUser)
        {
            pController.LoadObjectsFromBBDD(idUser);
        }
    }
```

RESUM

El món del videojoc està en constant evolució. Les últimes tendències marquen que els jocs online són els més acceptats per gran part dels usuaris. Aquest projecte forma part d'un videojoc orientat tant a la part més lúdica dels videojocs (visió clàssica), com a les xarxes socials i els negocis, combinat tot en una mateixa plataforma. Tot això integrat en un videojoc executable des del navegador i sense necessitat de cap tipus d'instal·lació. S'han desenvolupat una sèrie de mòduls per a satisfer les necessitats i els requeriments d'aquesta plataforma.

RESUMEN

El mundo de los videojuegos está en constante evolución. Las últimas tendencias marcan que los juegos online son los más aceptados por gran parte de los usuarios. Este proyecto forma parte de un videojuego orientado tanto a la parte más lúdica de los videojuegos (visión clásica), como a las redes sociales y a los negocios, combinado todo en una misma plataforma. Todo esto integrado en un videojuego ejecutable de navegador i sin necesidad de ningún tipo de instalación. Se ha desarrollado una serie de módulos para satisfacer las necesidades y los requisitos de esta plataforma.

ABSTRACT

The gaming world is in constant evolution. The last tendencies explain that online videogames are more accepted by users' community. This project is part of a videogame oriented both the lighter side of gamming (classic vision), as social networks and businesses, combined everything in the same platform. All of this embedded in a browser videogame with no installations. It has been developed some parts to satisfy the needs of the platform.