Institut Puig Castellar
Santa Coloma de Gramenet

Desenvolupament d'un videojoc multiplataforma en xarxa.

CFGS Desenvolupament d'Aplicacions Multiplataforma

Adrián García Belmonte
Pratik Kumar Patel
2nDAM
2019-2020

Resum del projecte:

El projecte tracta sobre el desenvolupament d'un videojoc per posar a prova tot el que s'ha après al llarg del curs, aquest és del gènere ***Battle Royale*** on es combinen elements de supervivència amb la jugabilitat d'un últim jugador en peu.

El joc depèn d'altres jugadors, per això ha de ser implementat un sistema multijugador i la compatibilitat amb el major nombre de plataformes possibles.

En els següents capítols veurem les primeres passes per a aquest desenvolupament, interacció entre els jugadors, preparació de l'escenari i la connectivitat en xarxa.

Finalment veurem les conclusions finals, fins on s'ha arribat i els possibles objectius / millores restants.

Paraules clau:

Videojoc, battle royale, mòbils, xarxa, multiplataforma.

Abstract:

The project is about developing a video game to test everything learned throughout the course, this is the Battle Royale genre where elements of survival are combined with the gameplay of a last standing player.

The game depends on other players so it must be to implement a multiplayer system and compatibility with as many platforms as possible.

In the following chapters we will see the first steps for this development, player interaction, stage preparation and network connectivity.

Finally we will see the final conclusions, how far we have come and the possible remaining objectives / improvements.

Keywords:

Video game, battle royale, scrum, mobile's, network.

Index

1. Introducció

1.1 Context i justificació de el projecte

L'elecció que es va proposat és un joc multiplataforma i multijugador per a dispositius mòbils i ordinadors, que es desenvoluparà amb un entorn gràfic (3D i 2D).

La nostra motivació principal era la de realitzar un joc en 3D ja que ja havíem fet alguns projectes en 2D i ens semblava interessant realitzar-ho en 3D. A més, l'opció de poder aprendre un altre llenguatge com C#. El motor **Unity** ens sembla una bona iniciativa per poder aplicar tots els coneixements d'aquests dos anys en alguna cosa diferent i fora de l'àrea de confort.

Desenvolupar un joc de la que tens una idea clara juntament amb l'actitud positiva, provocarà un desenvolupament satisfactori per a tothom. Al completar aquest treball el que volem és que sigui una aplicació entretinguda i confortable per a qualsevol tipus d'usuari.

El Joc està catalogat de tipus ***Battle Royale***, on consisteix en acabar amb els altres jugadors que també hi son a la partida, l'últim supervivent es el que guanya la partida.

1.2 Objectius del Treball

A l'hora de complir amb èxit el treball volem aconseguir els següents apartats:

1. Realitzar un joc en 3D.
2. Aprendre un altre llenguatge de programació (C#).
3. Aprendre a utilitzar el motor gràfic **Unity**.
4. Aconseguir que el joc funcioni de forma fluïda.
5. Posar música de fons mentre jugues la partida.
6. Descobrir el món del desenvolupament de videojoc.

1.3 Enfocament i mètode seguit

Atès que es tracta d'un projecte nou, hem arribat a la conclusió que, un cop enfocada la idea del projecte, l'estratègia principal a seguir és repartir el treball en dues tasques principals, per resoldre amb facilitat la gestió degut al escàs temps.

Un de nosaltres s'encarrega de la part del desenvolupament del joc. L'altre treballa la part de disseny, proposant noves idees de funcionalitats i millores per implementar en el projecte i les funcionalitats, com ara el disseny o millora d'un personatge.

1.4 Planificació de el projecte

El projecte ho fem en un grup de dues persones, la gestió serà primordial per complir els objectius i data del lliurament, així que efectuarà les tasques amb un diagrama planificat com a referència principal per aconseguir la feina per fer.

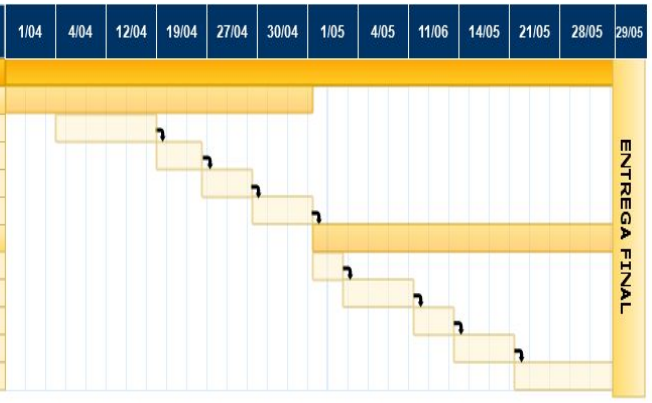
Per realitzar aquest projecte ens hem dotat de varies eines fonamentals la més rellevant consistiria a **Unity**, per desenvolupar el joc en general.

Un cop descarregada la versió estable de l'**Unity**, s'ha realitzat una sèrie de diagrames per tractar d'organitzar, orientar i preparar l'organització de les tasques.

No obstant això, cal destacar que la complexitat en què ha derivat el projecte i el funcionament específic d'**Unity** han fet que tots aquests diagrames, que en un principi vam dissenyar amb una certa viabilitat, han quedat totalment eclipsats i obsolets perquè puguin representar fidelment cada un dels components que componen el nostre joc.

A continuació adjuntem el nostre diagrama de Gantt sobre la planificació del projecte. A l'annex 1 es pot consultar aquesta informació ampliada.

N	Tasques	Duració	Inici	Final
1	**Desenvolupament de un videojoc multiplataforma en xarxa.**	165H	01.04.20	30.05.20
2	**Tasques del mes d'Abril**	78H	04.04.20	30.04.20
3	Disseny de esquemes amb UML	24H	04.04.20	12.04.20
4	Estudi sobre les eines del motor grafic	18H	13.04.20	19.04.20
5	Creació del Jugador (colisions, moviment, armes)	18H	20.04.20	26.04.20
6	Creacio del Mapa (Edificis, decoracio, etc)	18H	27.04.20	30.04.20
7	**Tasques del mes de Maig**	84H	1.05.20	29.05.20
8	Programa la conectivitat en Xarxa	18H	01.05.20	03.05.20
9	Programa el programa servidor	18H	04.05.20	10.05.18
10	Exportacio a diferents plataformes	19H	11.05.18	14.05.20
11	Publicacio a la Google Play	10H	15.05.20	21.05.20
15	Estudi de dades i conclusions	18H	22.05.20	28.05.20



1.5 Estudi econòmic i pressupostari

1.5.1 Pla de control del pressupost

Per a la realització del pressupost, s'ha tingut en compte els costos materials, és a dir equip informàtic, programari utilitzat, cost dels recursos humans i altres elements:

Per dur a terme el projecte s'han considerat els següents elements:

- Recurs Humans: Salaris de cadascun dels rols participants al projecte:
  - Adrián García: Programador (1.100€/165H)
  - Pratik Kumar: Programador (1.100€/165H)

- Material: Hardware i Software
  - 2 Equips sobretaula (3500€)
  - Llicencia de Software (Gratuït)
  - Models i dissenys del joc (20€)
  - Renta dels servidors (30€/mes)

Pressupost inicial total: 5.750€

2. Descripció del projecte

2.1 Anàlisi de requisits

2.1.1 Requisits funcionals

El joc requerirá d'un menú principal on els jugadors tenen les opcions d'unirse a la partida, apartats de configuració del so i els gràfics, i si volen sortir.

Una vegada el jugador s'uneix a la partida, aquest requereix de poder fer el control del personatge per això s'ha tingut en compte els diversos dispositius on es pot ejecutar el joc i se ha organitzat els controls de la següent manera:

- Control del personatge

  Pel control en **ordinadors**, s'utilitzen les tecles següents:

  - W o ↑: Permet el moviment cap cap amunt
  - A o ←: Permet el moviment cap a la dreta
  - S o ↓: Permet el moviment cap abaix
  - D o →: Permet el moviment cap l'esquerra
  - Space: Permet fer un tret cap a la direcció apuntat

  En cas de la rotació, s'utilitza la posició del ratolí per determina la seva rotació, és a dir, el personatge sempre apunta al ratolí.

Rotació del personatge en funció de la posició del ratolí

En cas de altres dispositius com **mòbils o tauletes** s'utilitzen un joystick, on la seva direcció determina cap on es mou el personatge i apuntat amb un el butto de fer el tret.

Controls en dispositius mòbils

2.1.2 Requisits no funcionals

Per fer ús del joc, es requerirà una connexió a Internet prou estable per veure els altres jugadors de manera fluida, a més d'un equip el suficientment potent per carregar els gràfics.

Especificacions mínimes:

- Per ordinadors: Gràfica integrada amb un mínim de 1GB de memòria.
- Android: Versió 5.0 Lollipop o superiors (API 21)
- iOS: Versió 10.0 o superiors.

Per millor experiencia es recomana jugar mitjançant una connexió a través de WiFi o en cas de un ordinador utilitzant un cable directament al router.

2.2 Tecnologies

A l'inici del projecte ens vam anar fixant en **Godot**, aquest és un motor gràfic multiplataforma per realitzar videojocs, ho vam veure amb bona opció però a al final ens vam decantar per un altre a causa de el temps extra que ens portaria aprendre un altre llenguatge.

Unity és el que es coneix com un motor de desenvolupament o motor de jocs. El terme motor de videojoc, game engine, fa referència a un programari el qual té una sèrie de rutines de programació que permeten el disseny, la creació i el funcionament d'un entorn interactiu; és a dir, d'un videojoc.

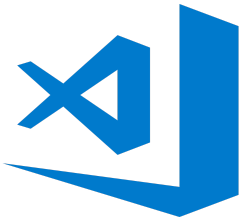
Com s'ha dit, aquests scripts podem escriure'ls en algun dels llenguatges que **Unity** soporta de forma nativa: C #, UnityScript i Boo. Pots llegir una mica més avall una descripció d'aquests llenguatges.

Nosaltres emprarem C # en tots els nostres exemples i projectes per diverses raons:

És el llenguatge que prima el propi motor de jocs **Unity**, ja que li dóna un suport especial i bassa tota la seva documentació en ell.

És que és el més utilitzat, amb molta diferència (veure gràfic baix), per la comunitat d'usuaris d'**Unity**, el que facilita després l'aprofitament de scripts creats per tercers en els nostres propis videojocs.

La seva similitud amb Java, ho fan molt fàcil d'utilitzar per aquells que treballen en l'entorn Android.



El Visual Studio Code pot ser un gran acompanyant per a **Unity** per editar i depurar fitxers C #. Totes les funcions de C # són compatibles i molt més. A la pantalla següent, podeu veure el color del codi, la concordança entre claudàtors, IntelliSense, CodeLens i això és només l'inici

---

Mixamo és una empresa de tecnologia de gràfics per ordinador en 3D. La companyia desenvolupa i ven serveis basats en la web per a l'animació de personatges en 3D. Les tecnologies de Mixamo utilitzen mètodes d'aprenentatge automàtic per automatitzar els passos de l'procés d'animació de personatges, inclòs el modelatge 3D en aparells i l'animació 3D.

Draw.io és l'eina google que et permet dibuixar tot tipus de diagrames, ja siguin diagrames UML o un dibuix qualsevol. Dins del projecte aquesta eina la utilitzarem per a fer el mock -ups de l'aplicació.

Modelio és una eina utilitzada per al desenvolupament de diagrames UML. És un programari de codi obert. Principalment s'utilitza en les primeres fases de desenvolupament per a establir l'arquitectura que tindrà el teu programa. Dins d'aquest projecte s'utilitza per a fer diagrames de classes i estat.

2.3 Estructura del projecte

Com es pot apreciar en les següents imatges, es defineixen dues escenes dins el projecte, els GameObjects que apareixen en gris, pertanyen a les diferents pantalles que s'activaran al desencadenar algun dels següents esdeveniments:

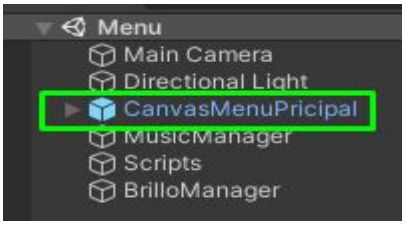
Escena del Menú Principal

1. Pantalla de jugar a el donar button de jugar.
2. Botó d'èxit per sortir de el joc.
3. Botó d'opcions per configurar el gràfic de el joc i so.

Escena del joc

1. Mini mapa	
2. Barra de vida	
3. Joysticks	
4. Menú de pausa	

2.3.1 GameObjects i Prefabs



Els GameObjects són els components bàsics en la qual es fonamenta **Unity**. Aquests objectes representen tot el contingut d'una escena, des d'objectes tangibles del món com a personatges, fons i ítems varis, fins a objectes intangibles com la pròpia cambra, o els botons de la interfície de joc. Això vol dir que qualsevol objecte que estigui en escena tant actiu com a inactiu, serà un **GameObject** necessàriament. Com els objectes que són, cada **GameObject** pot tenir un únic objecte pare, funcionalitat de gran utilitat que pot servir per a agrupar diversos objectes en una única referència, posició, etc.

Els GameObjects són la unitat bàsica en la qual es fonamenta **Unity**. Aquests objectes representen tot el contingut d'una escena, des d'objectes tangibles del món com a personatges, fons i ítems varis, fins a objectes intangibles com la pròpia càmera, o els botons de la interfície de joc.

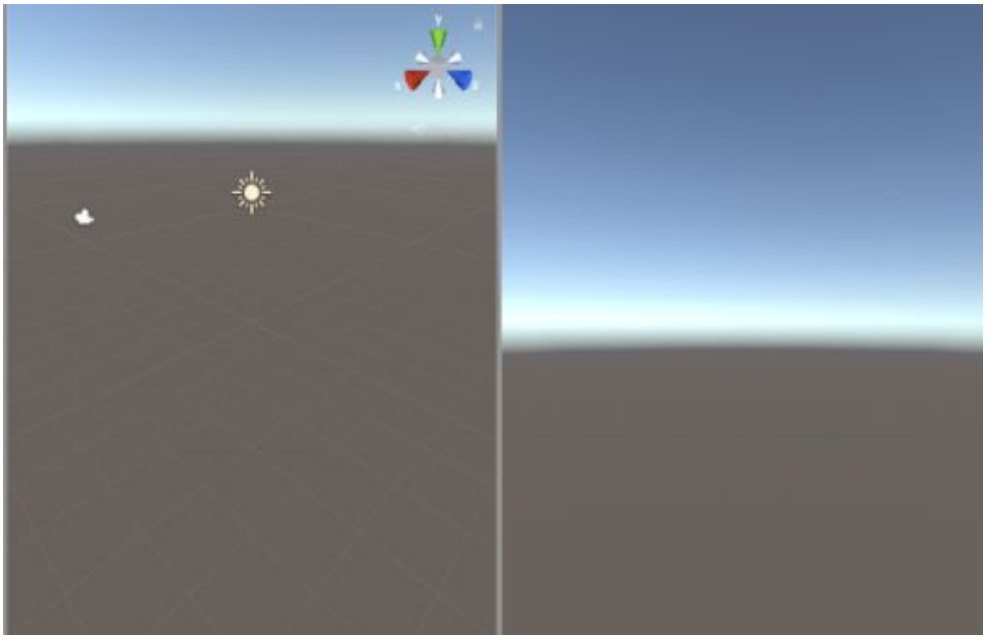
Això vol dir que qualsevol objecte que estigui en escena tant actiu com a inactiu, serà un **GameObject** necessàriament. Com els objectes que són, cada GameObject pot tenir un únic objecte pare, funcionalitat de gran utilitat que pot servir per a agrupar diversos objectes en una única referència, posició, etc.

2.3.2 Scene

Al llarg de l'explicació dels GameObjects s'han anat fent referències a les escenes del joc. Les escenes, o scenes en el motor, són un entorn compost per GameObjects que representa un espai del joc. És a dir, un joc en **Unity** es compon d'una successió de scenes cadascuna amb els seus GameObjects.

Els objectes que contingui una escena seran carregats en el moment de carregar l'escena, i destruïts en carregar una altra. No obstant això, es pot indicar en el script de qualsevol objecte que no es destrueix en carregar una altra escena.

Encara que això suposa haver de portar un seguiment d'aquells objectes que romandran sempre carregats per a evitar tornar a carregar-los i duplicar-los, també comporta menys accesos a memoria, la qual cosa es tradueix en major rapidesa de càrrega. Carregar una escena suposa una despesa computacional important, ja que cal carregar tots els objectes i components que la constitueixen. Per això és important estudiar bé com dividir el joc en escenes, evitant estar constantment carregant l'una o l'altra.



Vista del escenari en Unity

3. Desenvolupament del joc

3.1 Desenvolupament inicial

Per començar a utilitzar **Unity** ens vam anar a la oficial per veure el funcionament bàsic del motor i començar a col·locar els personatges i programar documentació el seu comportament.

- **Càmera:** situada des del punt de vista de el personatge principal, que proporciona una major visualització de l'entorn que envolta el jugador.

- **Personatge:** És l'actor principal de l'escena el qual controla el jugador.

- **Lògica de sons** associats a:
    1. Tret de l'arma de el personatge manejat per l'usuari.
    2. Col·lisió de l'enemic amb el jugador.
    3. Col·lisió entre el jugador i l'objecte de vida.

- **Menú principal:** per a l'inici del videojoc, a més de pantalla de joc superat , amb la possibilitat de tornar a començar el videojoc en tots dos casos.

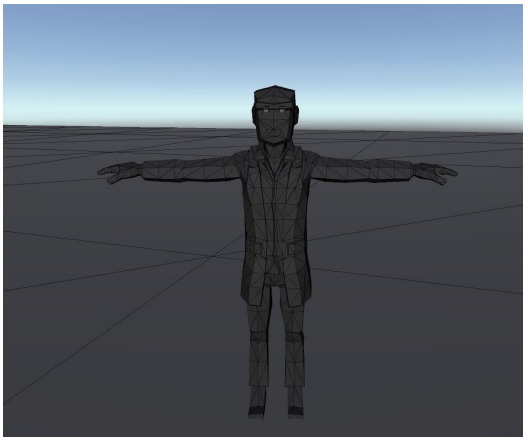
- **Control de puntuació** a l'encertar amb un projectil a l'enemic (es visualitza en pantalla al costat de l'arma de el personatge).

- **Dos controls**, a manera de joystick, per gestionar el videojoc desde la pantalla táctil d'un dispositiu Android. A més es controlarà en tot moment una segona pulsació, en qualsevol lloc de la pantalla, per al llançament de projectils.

- **Pantalla de pausa**, que permet l'usuari aturar el joc i continuar exactament en el mateix estat que abans de la seva pausa. També serà possible sortir del videojoc des d'aquest pantalla.

3.2 Moviment del personatge, animacions i armes

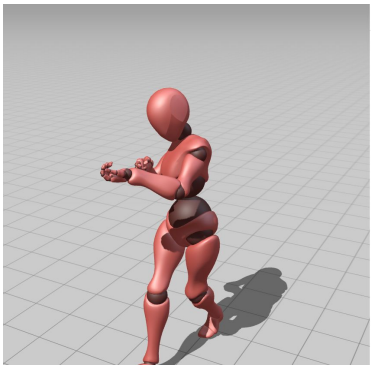
Una de les primeres coses que es va preparar és el moviment del personatge, per fer això, cal importar el seu respectiu model i assignar-li un script, aquest contindrà una estructura de control que s'encarrega de verificar que tecles s'estan fent clic, si són les fletxes el personatge es mourà cap a una direcció d'aquesta.



Model del personatge inicial

Un cop tenim el moviment, afegim les animacions perquè es mogui de manera més natural, per a això utilitzarem Mixamo.

Mixamo és una plataforma on es poden obtenir diverses animacions, des de les més bàsiques com caminar fins a tot tipus de balls.



Exemple de animació en mixamo

Després de triar les animacions que ens resulten més interessants, hem de afegir-les a el projecte, per gestionar les animacions **Unity** ens dota amb *AnimatorController*, un component que dins trobem una màquina d'estats amb cadascuna de les animacions

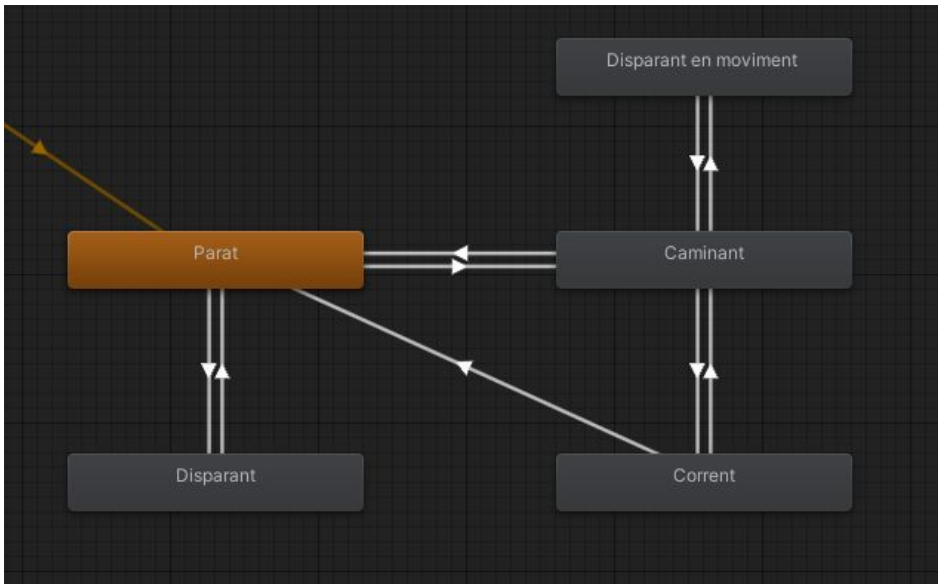


Diagrama de la màquina d'estats de les animacions

Com es veu al esquema, tenim diversos rectangles que corresponen a varis estats que pot prende el personatge, per cambiar entre estats tenim unes fletxes apuntant uns als altres, aquestes tenen les següents navegacions:

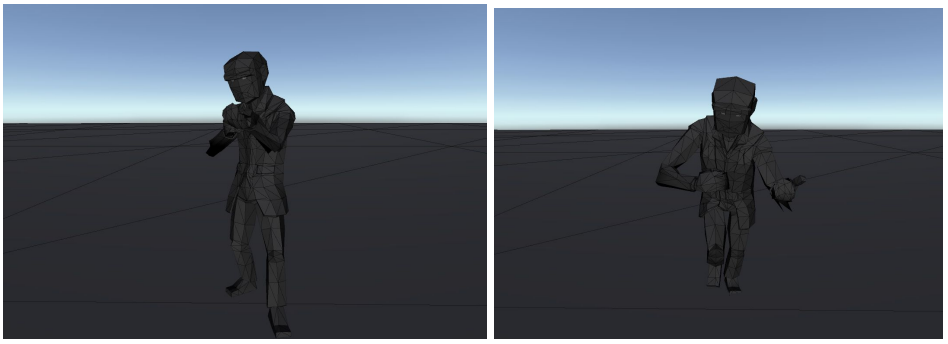
- **Parat:**

  El personatge es troba parat i pot passar a disparar o a caminar.
- **Disparant:**

  El personatge fa l'animació de disparar i torna a parat.
- **Caminant:**

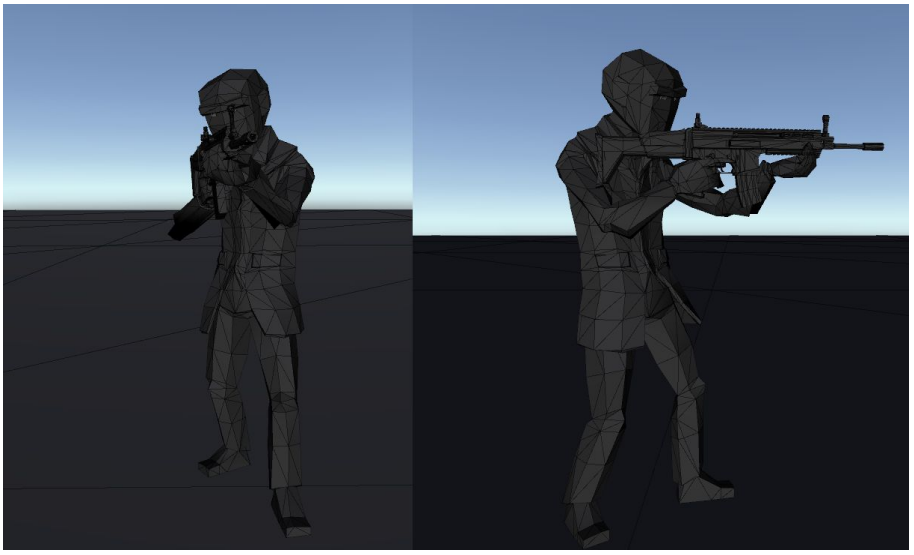
  El personatge es troba caminant, aquest pot pasa a corre, dispara mentres camina o pararse.
- **Corre:**

  El personatge comença a corre pero no pot disparar a menys que torni a caminar o aturat.



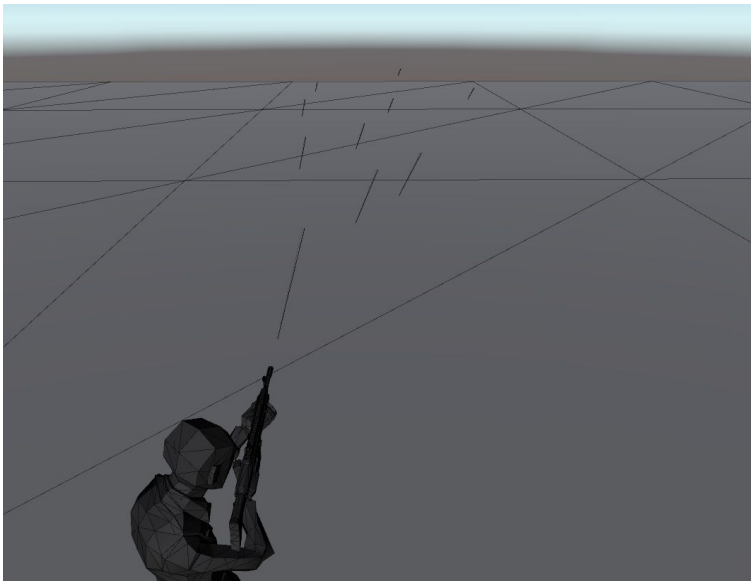
Personatge apuntant / corrent

Finalment per fer que el jugador pugui disparar primer haurem associat una arma, aquesta hauria de quedar ancorada a la mà del jugador.



Personatge empunyant una arma

Després, al script que controla el moviment, s'afegirà una condició que en cas que es premi la tecla de tret, vam crear una bala a la punta de l'arma i li apliquem força física perquè surti disparada de l'arma. Fent ús de timestamp controlem el temps que ha tornat a ser disparada per simular-hi la cadència.



Simulació de la dispersió dels trets

Un cop disparada, aquesta contindrà un script independent a el de tret, que en cas de no impactar en un cert temps sigui destruïda.

3.3 Creació del mapa, decoració i gràfics.

Per a la creació de mapa, prèviament vam adquirir uns assets d'acord amb les nostres necessitats, ja que fer-nous no és el nostre objectiu en el projecte, després vam anar començant a muntar el mapa.

Primeres versions de la zona de practica a l'escenari

Un cop vam adquirir mes pràctica amb l'editor, vam començar a millorar l'escenari afegint més elements.

Zona de pràctica amb més decoració

Com es pot veure en les imatges anteriors, l'escena no té il·luminació per a això **Unity** per defecte ve amb un objecte que simula la llum del sol, generant així, tota la il·luminació i ombres automàticament, però el resultat no ens van convencen així que vam decidir aplicar-li un plugin de **postprocessament**, aquest ens ajuda a millorar els gràfics de l'escena, donant resultat a una millor il·luminació, ombres i colors. El resultat és el següent:

Resultat de l'escena aplicant postprocessat

Imatge del personatge fent uns trets de prova

3.4 Interfície d'usuari (UI)

3.4.1 Creació de la pantalla d'inici (Menú Principal)

Per començar a desenvolupar es tindrà en compte les característiques que utilitza la plataforma **Unity**, com les relacions que té les escenes i entre els seus nodes.

Pas 1 - Creació canvas i botons.

1. Canvas:

El GameObject Canvas és un contenidor on podem incloure tots els objectes que componen el nostre interfície d'usuari (textos, imatges, menús, botons ...) que apareixen a la pantalla.

Per utilitzar el GameObject desde el propi editor d'**Unity**, només cal accedir al menú GameObject > UI > Canvas.

2. Botó:

El control del Button respon a un clic de l'usuari i és utilitzat per iniciar o confirmar una acció.

Pas 2 - On practiquem posar un fons de pantalla, els botons i també com navegar entre altres escenes.

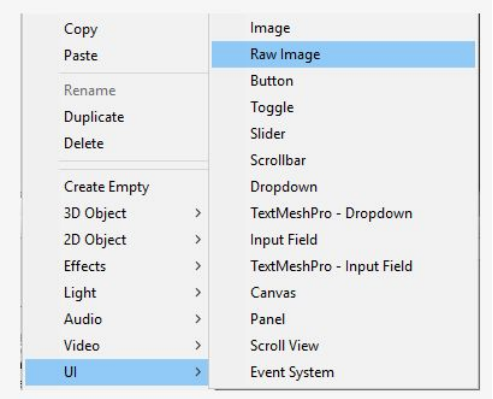
El que vam fer, és crear un canvas i dins d'aquest posar els botons, que en aquest cas són tres botons i cada botó té una funcionalitat:

1. El botó de l'jugar - Bàsicament canviarà de l'escena i portarà a una altra escena.
2. El botó d'opcions - Portarà en una altra escena on hi ha més opcions (Menús), com a opcions del gràfic de joc i volum en general.
3. El botó de sortida - Abandones el joc bàsicament.

Escena Menú Principal

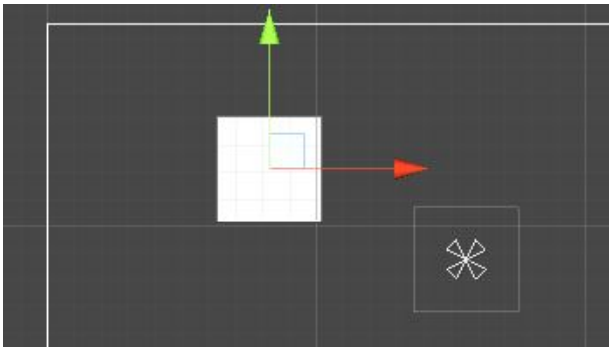
3.4.2 Creació del minimapa

Primer que res, anirem a la jerarquia de el projecte i allà crearem un objecte tipus Raw Image a la secció UI:



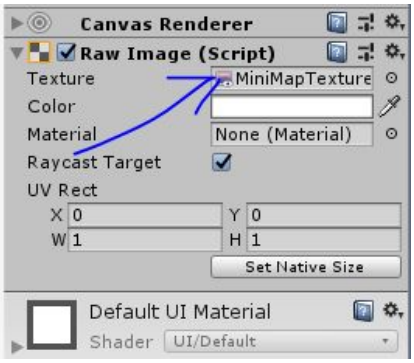
Anem a fer un parèntesi per explicar per què usem una Raw Image en comptes de la tradicional Image, la diferència és que la Imatge accepta com a paràmetre un Sprite, que acaba sent precisament una imatge solta, però nosaltres volem que la imatge s'actualitzi en viu, si veiem en el component raw image, ens accepta com a paràmetre un objecte tipus textura, i aquí és on ocorre la màgia.

Hi ha un asset en **Unity** que es diu Render Texture, aquesta textura permet ser actualitzada en temps real i llegida pel nostre objecte raw image, ara que tenim nuesta Raw Image posicionada en el nostre canvas es vora més o menys així:

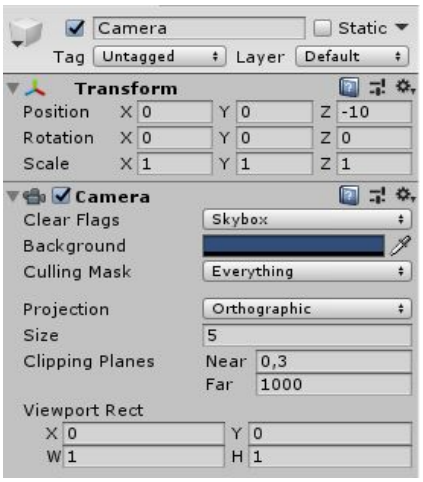


Ara la nostra Raw Image aquesta mostrant una imatge blanca per defecte, per això necessitem crear el nostre RenderTexture i arrossegar-lo a l'inspector per enllaçar-los, per això fem clic dret a la nostra secció de projecte> create> render texture.

Ara canviem de nom la nostra textura amb un nom descriptiu, i enllacem en l'inspector del RawImage:



En aquest punt ara ens preguntem, qui actualitza aquesta textura en temps real?, perquè una càmera, el minimapa, és en efecte una segona càmera, que veu de dalt a baix sense perspectiva al nostre personatge, llavors crearem una nova camara en nostra jerarquia, per a això fem clic dret en la jerarquia > càmera.



D'aquí canviarem diverses coses, primer que res necessitem que la posició X i Y siguin idèntiques al nostre personatge, en cas que el projecte sigui 3D, hem de modificar la rotació en X perquè miri cap avall, en el cas de 2D hem de disminuir la posició en Z de manera que la cambra quedi veient des de "Dalt al nostre personatge".

Després hem de canviar la projecció de la cambra de perspectiva a ortogràfica, amb el paràmetre *Size* que tenim ara per defecte en valor 5, podem modificar el Zoom del nostre minimapa, eliminem el component AudioListener de la nostra cambra i la nostra cambra que aquesta gairebé llesta hauria de quedar mes o menys així:

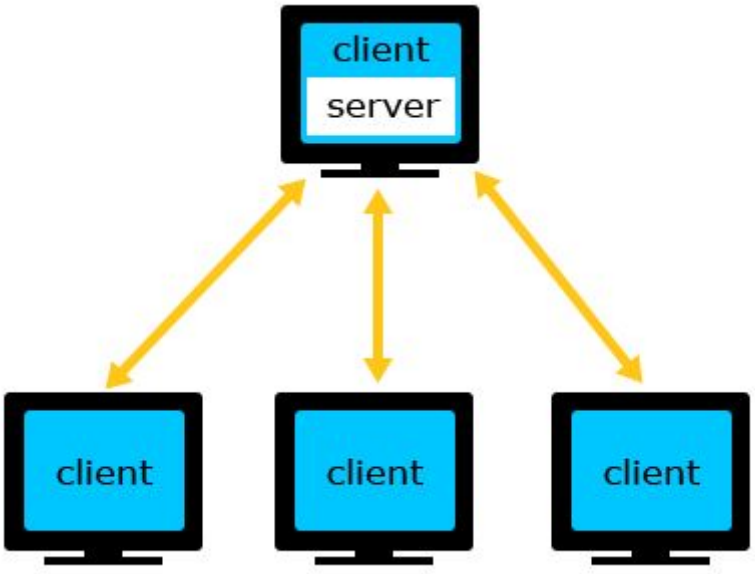
Resultat final del minimapa

3.6 Creació del servidor

Per a poder utilitzar el joc, és necessari que els clients estan connectats entre si per això es va requerir investigar sobre les diferents maneres d'implementar un servidor, protocols i quals són els seus avantatges i inconvenients.

3.6.1 Arquitectura Peer to peer

Aquest model consisteix en el fet que els clients que participen en la partida es connecten entre si i trien al jugador amb la millor connexió que farà de servidor.



Esquema arquitectura Client-Servidor

El principal avantatge d'aquesta és que els propietaris del joc no necessiten invertir en servidors per a allotjar les partides ja són els clients qui fan de servidor, estalviant costos en el pressupost.

Els majors problemes d'aquest model consisteixen en la capacitat dels clients d'aguantar la càrrega de xarxa, si un **client** experimenta una fallada de connexió o es desconnecta, tots els clients experimentessin problemes afectant l'experiència de joc.

Unity té un complement on ajuda als desenvolupadors a implementar aquesta arquitectura però al final es va decantar per un altre model degut als inconvenients d'aquesta.

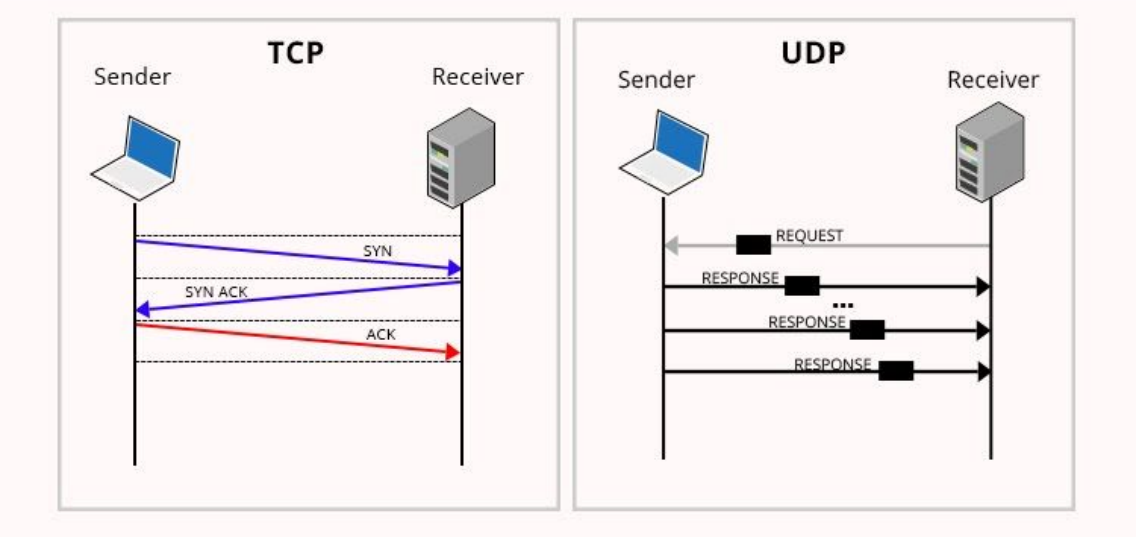
3.6.2 Arquitectura Client-Servidor

En aquest model els clients es connecten a un servidor dedicat on s'allotja la partida, aquest s'encarrega d'administrar la partida i enviar els diferents estats de la partida als jugadors.

Aquest és el model que s'ha utilitzat pel nostre projecte, ja que permet gestionar els estats dels jugadors més fàcilment a més d'oferir una major estabilitat en la connexió cap al servidor.

3.6.3 TCP i UDP

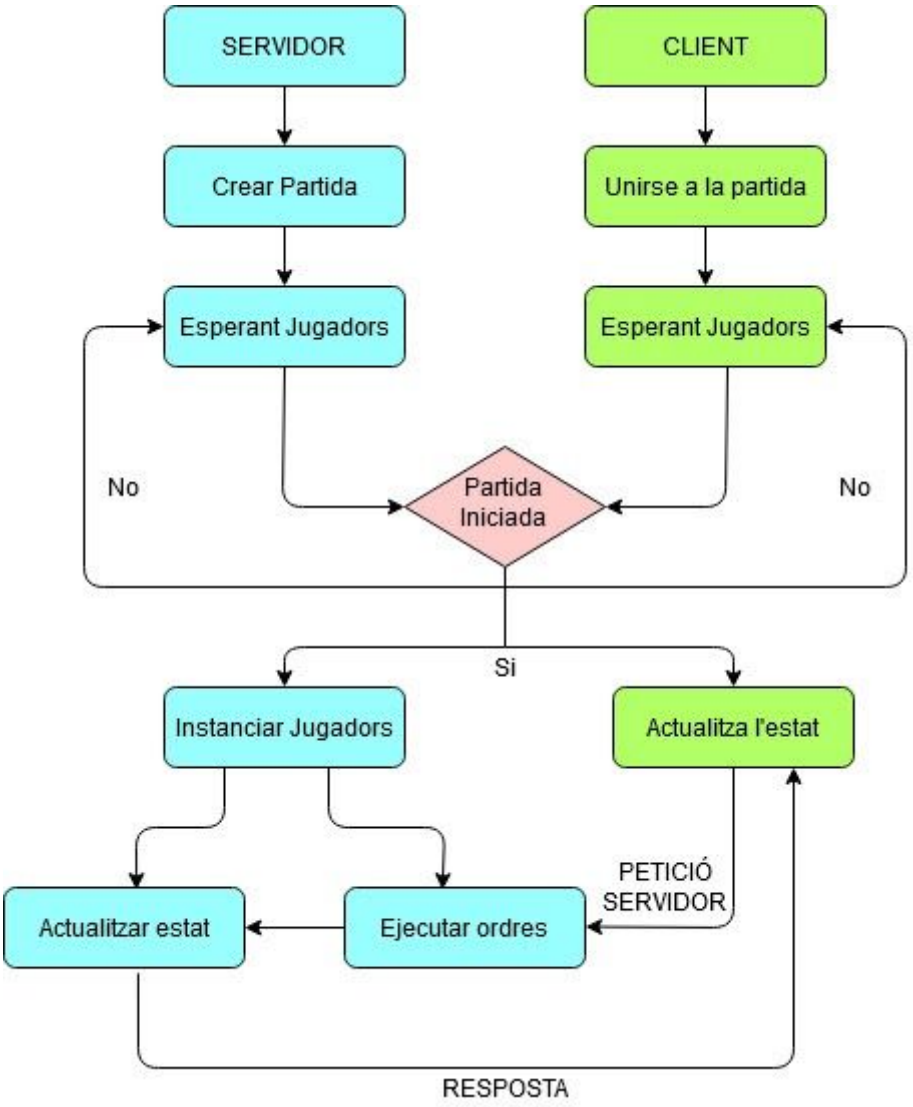
TCP i UDP són dos protocols per a realitzar comunicacions a través de la xarxa, les principals diferències són que TCP és orientat a connexió, la qual cosa ho torna més fiable a l'hora de transmetre dades en canvi UDP és més ràpid a l'hora d'intercanviar amb latències més baixes a costa de major pèrdua de dades.



Comparació dels protocols TCP i UDP

Al principi es va començar a construir el servidor mitjançant el protocol UDP, ja que és el més indicat per a videojocs però aquest té una gran dificultat d'implementació ja que requereix tenir en compte quan un **client** està connectat o les dades que es perden pel camí a més de que aquest és unidireccional i s'ha de controlar d'on provenen les dades.

Llavors es va considerar usar TCP, ja que la comunicació és bidireccional, és a dir que el client i el servidor poden comunicar-se al mateix temps i no han d'esperar a la resposta de l'altre, a més de ser més fàcil d'implementar, però a canvi es veu incrementada la latència. Finalment es va descartar el prototip de servidor en UDP i es va desenvolupar utilitzant TCP.



Esquema de funcionament del Client / Servidor

En resum, el servidor crea la partida i espera al fet que els jugadors s'uneixin a ella, una vegada connectats, el servidor envia el seu respectiu identificador a cadascun dels clients i seguidament repetidament els clients donen a conèixer les seves coordenades al servidor perquè aquest les re-enviï a la resta de jugadors, una vegada comença la partida els jugadors podran fer ús del tret i es veuen entre ells.

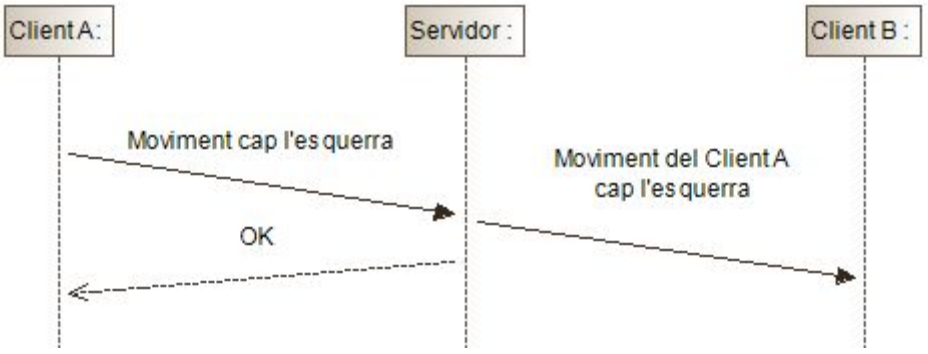
3.6.4 Mètode d'enviament de les dades

Una vegada definit el protocol, es requerirà analitzar quina és la manera més òptima d'enviar les diferents dades del jugador que consisteix en els següents:

- Identificador del jugador (ID)
- Posició i rotació
- Bales disparades
- Punts de salut
- Animació actual (Aturat, corrent, disparant, abatut).

Per a donar a conèixer tota aquesta informació un dels mètodes consistiria a **enviar les accions** que ha realitzat el jugador, si està prement la tecla per a moure's cap endavant o si ha disparat, el servidor calcula la trajectòria de la bala i confirma si el jugador enemic ha estat aconseguit o no.

Pel fet que el jugador sol enviar accions aquest mètode és menys susceptible a manipulacions ja que al no envia coordenades de posició, tindria més dificultats per a realitzar paranys. D'altra banda aquest sistema és més complicat d'implementar ja que es necessiten més dades per a fer conèixer la posició del personatge i consumeix més recursos per al servidor.



Exemple de enviament d'accions d'un client al servidor

El mètode alternatiu consisteix a **enviar l'estat actual** de cadascun dels elements del **client**, és a dir, enviem constantment la situació del **client** al servidor perquè aquest únicament la re-enviem als altres jugadors, aquest mètode és més senzill d'implementar però és susceptible a paranys.

Finalment s'ha realitzat un híbrid per a la manera d'enviar les dades, d'una banda, la posició del jugador i la seva rotació es s'envien les seves coordenades al servidor i aquest les transmet als altres clients, al igual que el seu estat d'animació, aquesta manera és la menys adequada però tenint en compte el temps necessari per a implementar el sistema d'accions era excessiu i es deixa apartat per a una futura actualització.

Per a realitzar l'enviament de la bala, el **client** envia al servidor l'acció de disparar i el servidor retorna als clients el punt on s'origina la bala, una vegada es rep aquesta informació el **client** genera una instància de la bala en aquest punt i li apliquen una força perquè faci la funció de tret, **Unity** calcula automàticament la trajectòria de la bala i on col·lideix evitant la necessitat d'enviar la posició de la bala constantment igual que passava amb el jugador.

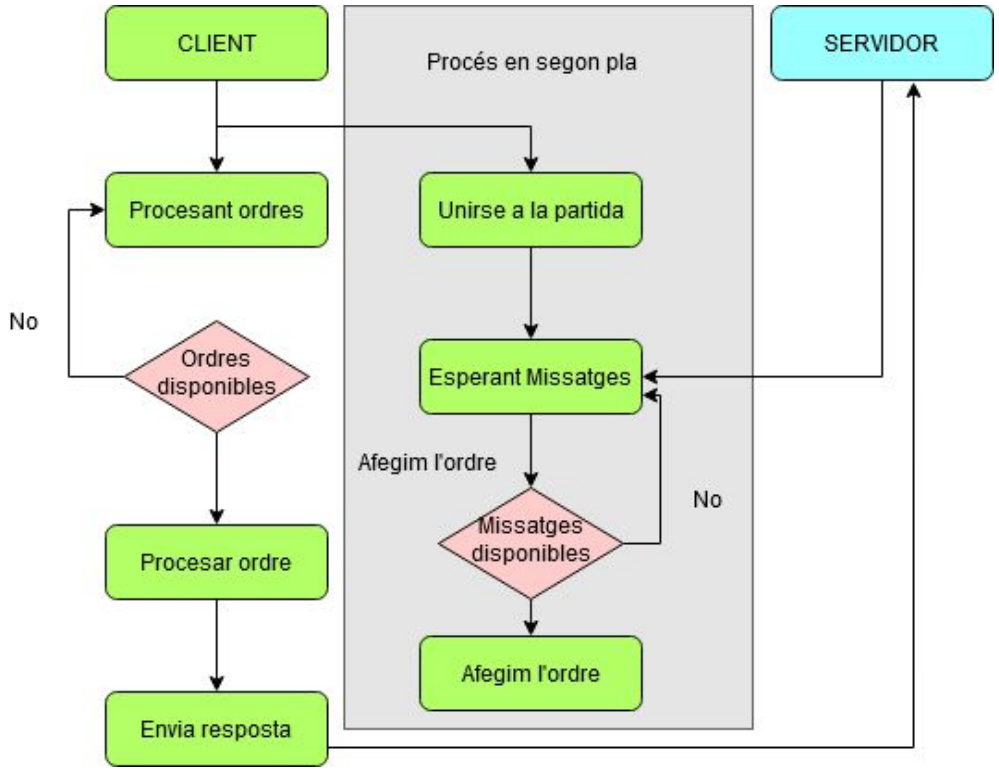
Els punts de vida es resten com a la bala col·lideix amb algun jugador, així que no és necessari enviar aquesta informació a través de la xarxa.

3.6.5 Estructura de les dades

Les dades que envien tant el **client** com el servidor es compon en una cadena de text composta per:

- Petició (Moviment, Rotació, Tret)
- Dades (Coordenades, ID d'usuari)
- Altres (Estat de l'animació)

Una vegada els clients reben la informació del servidor, per al seu processament, aquesta s'envia a una cua on un bucle executa les ordres pertinents en funció del rebut, això és així pel fet que el **client** s'executa en segon pla i **Unity** no permet anomenades al seu API per seguretat, és a dir, no és permès fer moure el personatge o disparar a una bala en un segon pla, llavors en col·locar-se en una cua d'execució es realitza en primer pla.



Esquema del funcionament del client

3.6 Exportació a Multiplataforma

Es conclou el desenvolupament del joc amb la seva exportació a diferents plataformes, els motors de desenvolupament de videojocs moderns com **Unity** permet fàcilment realitzar aquest procés únicament s'ha de tenir en compte els perifèrics que s'utilitzen en cada cas, per això, aquest joc té implementat l'ús de joysticks per al seu ús en dispositius mòbils i tauletes.

Unity té compatibilitat amb multitud de plataformes però la majoria d'aquestes (iOS, PS4, Switch) són privatives i requereixen una adquisició d'una llicència. Les plataformes on ha estat possible exportar el projecte són les següents:

- **Windows**
- **Android** (Móviles y tabletas)
- **iOS** (iPhone, iPad)

Imatges dels diferents dispositius fent ús del joc, de esquerra a dreta,

MacOS*, *iOS*, *Android*, *Windows

4. Conclusions

Finalment s'ha pogut finalitzar la gran majoria d'objectius com el fet de tenir una jugabilitat finalitzada i poder gaudir del joc en xarxa encara que amb alguns matisos dependent del dispositiu usat i la seva qualitat de connexió de la xarxa.

A causa del temps ha quedat un objectiu pendent i és el de publicar una versió de proves en la Google Play, ja que les proves realitzades no és factible llançar fins i tot el joc, per consegüent es queda pendent les següents qüestions:

1. **Optimització i millora de la comunicació en xarxa:**

   És necessari migrar el model de TCP cap a UDP per a millorar la latència en dispositius on la connexió de xarxa es realitzi mitjançant Wifi o Xarxes mòbils, ja a la mínima pèrdua de senyal el joc sofreix de lentitud.

   També és necessari un sistema d'aparellament (Matchmaking) perquè estiguin disponibles diversos servidors en línia, actualment només és suportat un.

2. **Millores en la jugabilitat:**

   S'havia plantejat afegir la possibilitat de recuperar vida, sistema de munició i zona de tempesta, aquest últim estava implementat en el joc mancant programar, aquest consisteix en un cilindre on al principi ocupa la totalitat del mapa i al llarg de la partida va minvant, els jugadors han de romandre en el, de no ser així van perdent vida constantment fins que tornin a entrar en el, així en cas que els jugadors estiguessin molt dispersats pel mapa s'acaben trobant.

*Vista preliminar de la zona segura des de l'editor de **Unity***

5. Glossari

Unity:

Motor de videojocs multiplataforma.

TCP i UDP:

Protocols a nivell de transport basat en que permet establir una connexió i l'intercanvi de dades entre dos amfitrions.

Client:

Component que comunica la aplicació amb el servidor.

GameObject:

Components dins de Unity que representen Personatges, Objectes, Música, dins del projecte.

UI:

Interfície d'usuari.

Timestamp:

Nombre de segons des de les 0 hores de la matinada de l'1 de gener de 1970 UTC

MacOS:

Sistema operatiu desenvolupat per Apple, es el software principal per els seus ordinadors.

iOS:

Sistema operatiu desenvolupat per Apple, es el software principal per els seus dispositius mòbils.

Android:

Sistema operatiu per a dispositius mòbils i tauletes desenvolupat per Google.

Windows:

Sistema operatiu desenvolupat per Microsoft, es el software principal per ordinadors.

6. Bibliografía

Principiant d'unitat Tutorials :
https://unity.com/learn/get-started

1. **Tutorial | Crear un menú principal en Unity | PC o Mòbil.**

   A. Part 1
   https://www.youtube.com/watch?v=XPcEcp9ZuW4&t=1760s
   B. Part 2
   https://www.youtube.com/watch?v=xXXNaIsMX7g&t=1240s
   C. Part 3
   https://www.youtube.com/watch?v=PW5zREl00Fo
   D. Part 4
   https://www.youtube.com/watch?v=aLb6k-uj3XI&list=PLTvi5WASMyz1Rrvw83mOVMNBxjJTgNrQo&index=4

2. **Com per fer un Minimap en Unitat.**
   https://www.youtube.com/watch?v=28JTTXqMvOU&t=297s

3. **Com per crear palanca de control mòbil en Unitat.**
   https://www.youtube.com/watch?v=8-X3BmvtXT0&t=29s

4. **Tacte Tercer Controlador de Caràcter de la Persona en Unitat.**
   https://www.youtube.com/watch?v=8ycgJbQegAo&t=158s

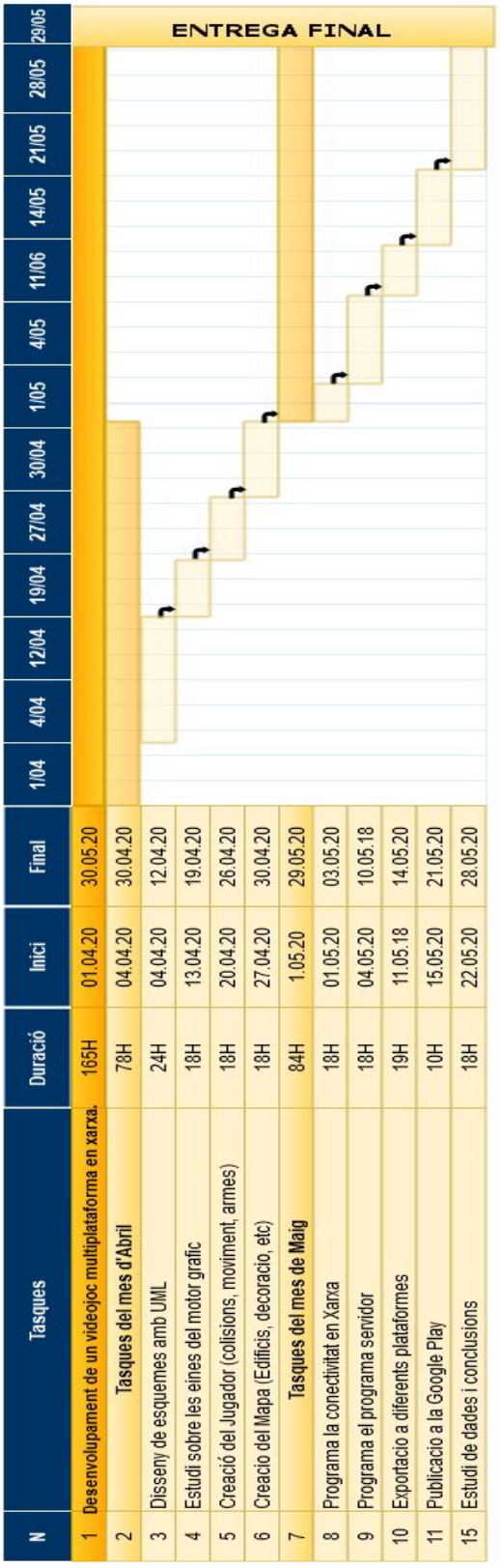
5. **Com crear la barra de vida.**
   https://www.youtube.com/watch?v=UKs1qO8w7qc&t=122s

6. **Com posar àudio i música en Unity.**
   https://www.youtube.com/watch?v=msSkbKb8CK4

7. **Connexió amb TCP**
   https://docs.microsoft.com/es-es/dotnet/api/system.net.sockets.tcpclient?view=netcore-3.1

7. Annexos

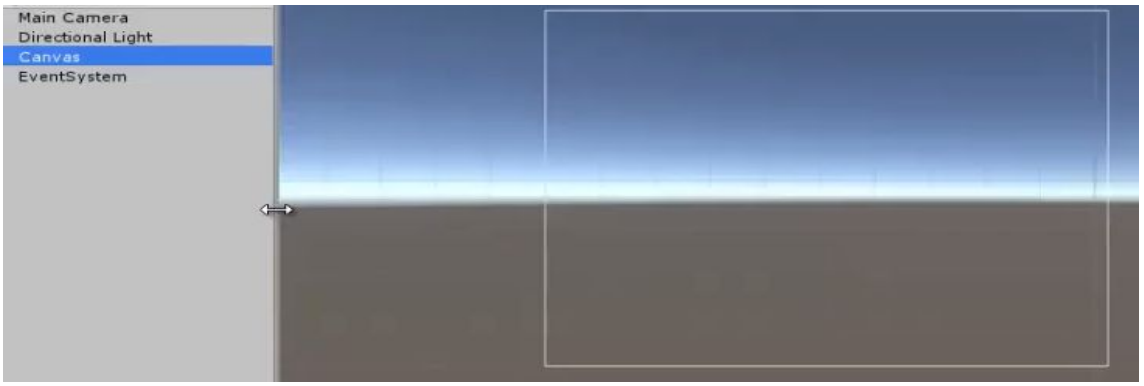
7.1 Diagrama de Gantt



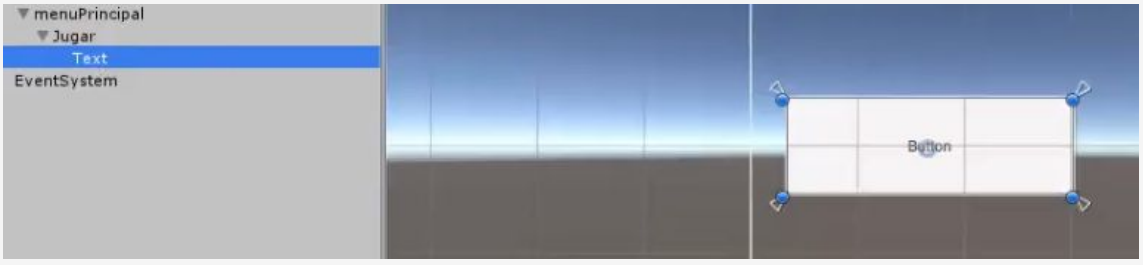
N	Tasques	Duració	Inici	Final
1	Desenvolupament de un videojoc multiplataforma en xarxa.	165H	01.04.20	30.05.20
2	Tasques del mes d'Abril	78H	04.04.20	30.04.20
3	Disseny de esquemes amb UML	24H	04.04.20	12.04.20
4	Estudi sobre les eines del motor grafic	18H	13.04.20	19.04.20
5	Creació del Jugador (colisions, moviment, armes)	18H	20.04.20	26.04.20
6	Creacio del Mapa (Edificis, decoracio, etc)	18H	27.04.20	30.04.20
7	Tasques del mes de Maig	84H	1.05.20	29.05.20
8	Programa la conectivitat en Xarxa	18H	01.05.20	03.05.20
9	Programa el programa servidor	18H	04.05.20	10.05.18
10	Exportacio a diferents plataformes	19H	11.05.18	14.05.20
11	Publicacio a la Google Play	10H	15.05.20	21.05.20
15	Estudi de dades i conclusions	18H	22.05.20	28.05.20

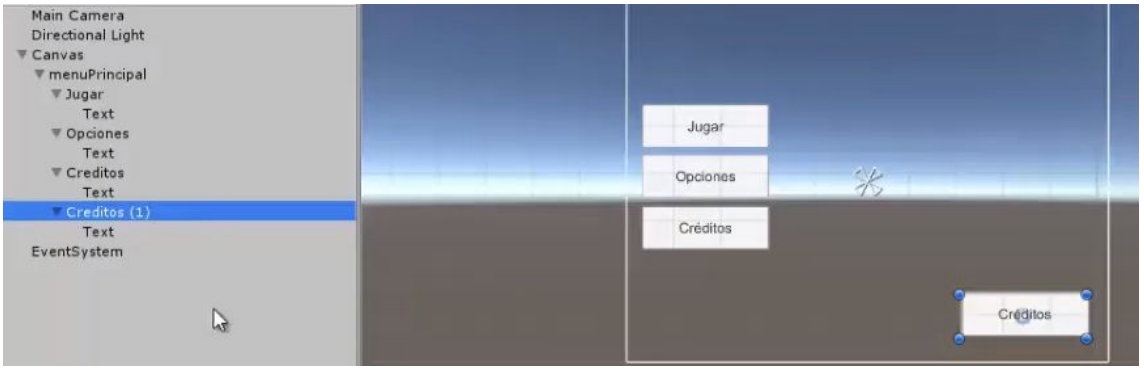
7.2 Creació del menú.

Pas 1 : Canvas.



Pas 2 : Panel i botons.





Pas 3 : Disseny.

7.3 Health Bar.