UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

Disseny i implementació d'un videojoc amb Unity

Treball de Fi de Grau
Grau en Enginyeria Informàtica
Enginyeria del Software

Àlex Humet Palomares

Directora: Claudia Patricia Ayala Martínez
Especialitat: Enginyeria del Software
Data: 20/12/2021

Resum

Aquest projecte consisteix a dissenyar i implementar un videojoc 2D dels gèneres exploració i supervivència, juntament amb el gènere *joc lliure*, conegut com a *sandbox*, amb el framework Unity i programat amb C# mitjançant l'entorn de desenvolupament Visual Studio. El videojoc s'anomena Lost Hero i l'objectiu d'aquest és explorar un món generat de manera procedural on el jugador haurà d'aconseguir recursos per fabricar-ne d'altres per així accedir a noves zones, i on podrà construir amb total llibertat i lluitar diferents enemics, entre d'altres.

En la memòria del projecte s'exposa la planificació del projecte, les diferents funcionalitats que componen el videojoc i els seus requisits, juntament amb una explicació detallada de l'arquitectura de software i del procés d'implementació.

Resumen

Este proyecto consiste en diseñar e implementar un videojuego 2D de los géneros exploración y supervivencia, junto al género *juego libre*, conocido como *sandbox*, mediante el framework Unity y programado con C# utilizando el entorno de desarrollo Visual Studio. El videojuego se llama Lost Hero y su objetivo es explorar un mundo generado de manera procedural donde el jugador tendrá que conseguir recursos para fabricar otros y acceder a nuevas zonas, y donde podrá construir con total libertad y luchar distintos enemigos, entre otras funcionalidades.

En la memoria del proyecto se expone la planificación del proyecto, las diferentes funcionalidades que componen el videojuego y sus requisitos, junto con una explicación detallada de la arquitectura de software y del proceso de implementación.

Abstract

This project involves designing and implementing a 2D exploration, survival and sandbox video game using the Unity framework and programmed with C # using the Visual Studio development environment. The video game is called Lost Hero and its goal is to explore a procedurally generated world where the player will have to get resources to access new areas and fight new enemies, and where he can build with complete freedom, among others.

The project document describes the planning, the different functionalities of the video game and its requirements, along with a detailed explanation of the software architecture and the implementation process.

Índex

Índex de figures

Índex de taules

1 Context

El projecte “Disseny i implementació d’un videojoc amb Unity” es tracta del Treball de Fi de Grau d’Enginyeria Informàtica de l’especialitat d’Enginyeria del Software realitzat en la Facultat d’Informàtica de Barcelona (FIB), Universitat Politècnica de Catalunya (UPC).

1.1 Introducció

Actualment, la indústria més gran d’entreteniment és la indústria dels videojocs, tant pels diners que mou com per la quantitat de persones que en fan ús, superant a la indústria cinematogràfica i la musical des de fa anys. Aquesta indústria està en continu creixement i genera milions d’euros al dia. El nombre de consumidors de videojocs en el món és aproximadament de 3 mil milions, és a dir, gairebé el 40% de la població total juga a videojocs, ja sigui en el seu dispositiu mòbil, en una consola o en l’ordinador [1].

Per entrar en context, un videojoc és un producte de software que implica la interacció amb una interfície d’usuari o un dispositiu d’entrada per generar una retroalimentació visual. Per la realització d’aquest és necessari disposar de dissenyadors, músics, desenvolupadors i modeladors, entre d’altres, però en aquest cas la intenció d’aquest projecte ha sigut dissenyar i implementar des de zero un videojoc en 2D adaptant tots els rols possibles en el procés. Els principals gèneres a destacar del videojoc son els de supervivència i exploració, juntament amb el gènere *sandbox*, un estil de videojoc caracteritzat per donar al jugador una llibertat total per ser creatiu. També abasta els gèneres d’aventura, acció, construcció, fabricació, 2D i píxel art.

El videojoc s’anomena “Lost Hero”, i el concepte principal d’aquest és la llibertat de poder explorar el món sense restriccions ni un ordre estricte a seguir per avançar. El jugador controla un personatge que es troba en un món sense explorar, i pot realitzar diverses accions amb les eines adequades per recol·lectar materials, construir noves eines i artefactes, descobrir i visitar nous biomes i coves o lluitar amb enemics que l’intentaran atacar, entre d’altres. L’objectiu del videojoc és sobreviure en el món en el qual es troba el personatge que es controla.

Lost Hero ha sigut desenvolupat mitjançant el motor Unity [2]. El que pretenc aconseguir amb aquest projecte no és trencar el mercat de videojocs amb una idea diferent i única, sinó aprendre una nova tecnologia, en aquest cas Unity, en el domini dels videojocs, ja que m’interessa aquest món i té molt potencial econòmic. Serà un repte per endinsar-me als requisits del mercat i poder posar en un futur el videojoc a la venda. Per aclarir, un motor de videojoc és un entorn de desenvolupament software dissenyat per a la creació de videojocs, i les principals funcionalitats que proporcionen solen incloure la renderització de gràfics 2D o 3D, el motor físic, la gestió d’animacions i del so, entre d’altres.

1.1.1 Unity

El motor Unity és un dels motors més populars. Segons el CEO d'Unity, John Riccitiello, és gairebé el motor de la meitat de tots els videojocs existents. Hi ha diferents quotes de mercat segons la plataforma, però més de la meitat de tots els videojocs per a mòbils estan creats amb Unity, entre un 60 i un 70 per cent per a màquines de realitat virtual o realitat augmentada, i aproximadament una mica més de la meitat de tots els jocs creats per a les plataformes de Nintendo, i una mica menys per Xbox i Sony, es desenvolupen amb Unity [3].

He escollit aquest motor pel gran potencial que té, per la integració multiplataforma que ofereix, la qual facilita el desenvolupament en diferents plataformes d'una manera ràpida i eficient, i pel fet que és un dels motors més utilitzats amb una àmplia documentació. El llenguatge de programació que he decidit utilitzar per desenvolupar el videojoc és C#, ja que és el llenguatge que més domino i el recomanat per Unity.

1.2 Problema

La situació actual de la COVID-19 en la qual ens trobem i els confinaments que hem viscut han afavorit al negoci dels videojocs, ja que aquesta situació ha fet augmentar encara més el consum d'entreteniment digital. Estar tancat a casa durant tot el dia genera avorriment, i una de les solucions és 'matar' el temps amb videojocs. I tot i que això sempre ha estat present, ja que tothom necessita desconnectar en algun moment del dia, aquests últims dos anys s'ha vist incrementat, deixant un 30% més de consumidors i un 40% més de guanys en la indústria [4].

El problema és que la majoria de videojocs són lineals, i a vegades el que un necessita per desconnectar és poder ser creatiu amb el que el joc li proporciona i no haver de pensar en quina missió es va quedar, en quina part de la història estava o quin nivell no aconseguia superar. Per això, aquest projecte busca crear un videojoc que satisfaci al jugador amb la llibertat de poder fer el que més li agradi dins del que el joc ofereix, i que sempre sigui una experiència única.

Per obtenir aquest resultat, s'implementarà una generació per procediments per la creació del món, és a dir, el contingut no estarà dissenyat per endavant, sinó que es crearà de manera aleatòria en base uns algoritmes. També s'implementarà un sistema d'artesania i de construcció, entre d'altres, donant així al jugador aquesta sensació de creativitat que es busca. I sobretot, es deixarà per lliure al jugador des del primer moment que comenci el joc.

1.3 Actors implicats

A continuació es descriuen els diferents actors implicats en el projecte:

- **Usuaris:** El producte anirà dirigit a qualsevol persona que disposi d'un ordinador, sigui portàtil o de taula. Si s'acaba implementant el requisit no funcional de multiplataforma, el qual té una prioritat baixa, el producte també anirà dirigit a persones amb dispositius mòbils o consoles. Els usuaris, en aquest cas, seran els mateixos jugadors de l'aplicació.

- **Directora del projecte:** La directora del projecte Claudia Patricia Ayala Martínez serà l'encarregada de dirigir i supervisar el projecte.
- **Desenvolupador:** El desenvolupador, en aquest cas l'estudiant Àlex Humet Palomares, serà l'encarregat de desenvolupar i documentar el projecte, a més de beneficiar-se en obtenir experiència per a un futur.

1.4 Limitacions legals implicades

El sistema que es desenvoluparà ha de seguir un seguit de normes legals llistades a continuació.

- Seguirà el sistema PEGI (Pan European Game Information), el qual és el mecanisme d'autoregulació dissenyat per la indústria per dotar els videojocs d'informació orientativa sobre l'edat adequada per al consum. Amb aquest sistema es mostren dos tipus d'icones descriptius, un relatiu a l'edat recomanada i l'altre al contingut específic susceptible d'anàlisis. En aquest cas, el videojoc entrarà en el marge d'edat de set anys (PEGI 7) pel sistema de combat amb diferents monstres, que entraria en l'apartat de violència d'aspecte no realista cap a personatges no humanitzats [5]. Aquest indicador es mostrarà en la portada del videojoc si aquest es posa a la venta en un futur en alguna plataforma de distribució de videojocs.
- Es tindran en compte els drets de Propietat Intel·lectual [6] dels autors dels dissenys gràfics del videojoc que no siguin creació pròpia i de la música i efectes de so utilitzats, sempre utilitzant aquells d'ús lliure personal i comercial i donant crèdit en la figura corresponent o en la bibliografia del projecte. Principalment s'utilitzaran llicències CC (Creative Commons), una eina legal de caràcter gratuït que permet als usuaris utilitzar les obres protegides per drets d'autor sense sol·licitar el permís de l'autor de l'obra.

2 Justificació

2.1 Productes alternatius

En l'actualitat existeixen més d'1 milió de videojocs, tenint en compte els que han sortit en físic o en plataformes digitals complint el marc normatiu, o més de 5 milions si afegim els que no surten de manera oficial [7]. Amb aquesta quantitat desorbitada és difícil crear un videojoc amb mecàniques úniques i un concepte totalment nou, fins i tot els videojocs més populars agafen idees de molts altres millorant-les i adaptant-les en el seu propi món. A continuació descriuré els videojocs de referència que he investigat i analitzat abans de començar aquest projecte:

- **Starbound** [8] és un joc d'exploració i construcció en el qual controlem a un personatge que ha abandonat el seu planeta natal, per estavellar-se en un altre i haver de començar de zero. Et pots trobar amb enemics, escenaris i armes tot generat aleatòriament, en un món que pots explorar per trobar objectes, fundar una pròpia colònia, construir instal·lacions, etc. Com podem observar en la *Figura 1*, és un joc 2D amb desplaçament lateral.

Figura 1: Imatge dins del videojoc Starbound
Font: *store.steampowered.com*

- **Astroneer** [9] és un videojoc on els jugadors poden construir bases personalitzades superficials o subterrànies, construir vehicles per explorar el sistema solar i utilitzar el terreny per crear qualsevol cosa que imaginin. A la *Figura 2* podem observar a tres jugadors explorant un planeta, el qual ha sigut generat aleatòriament, i a diferència de l'anterior videojoc, aquest és en 3D. L'enginy i la creativitat en Astroneer són clau per gaudir i esprémer tot el contingut que ofereix, cosa que es tindrà en compte amb el que es vol aconseguir en aquest projecte.

Figura 2: Imatge dins del videojoc Astroneer
Font: *store.steampowered.com*

- **Minecraft** [10] és un videojoc de món obert en primera persona que es centra en la col·locació i destrucció de blocs tridimensionalment cúbics. Els jugadors són lliures d'explorar el món, el qual es va generant infinitament i de manera aleatòria a mesura que vas avançant. En la *Figura 3* es pot observar una representació de com és el joc per dins.

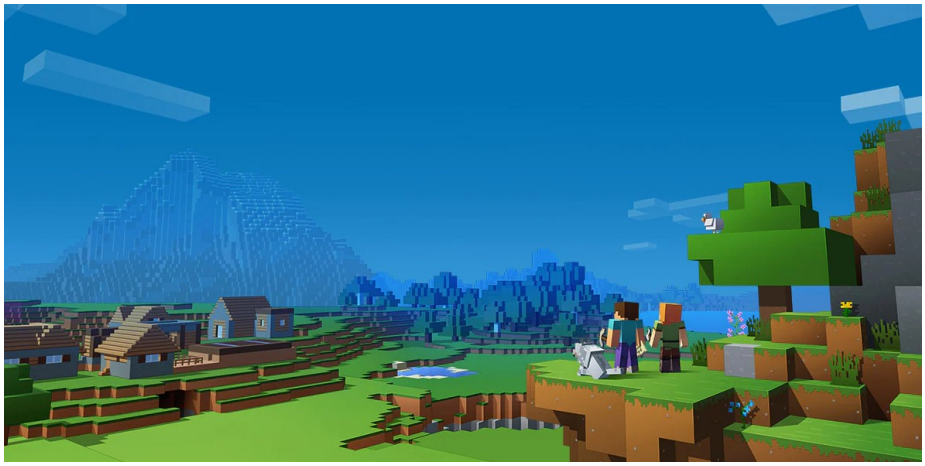


Figura 3: Imatge del videojoc Minecraft
Font: *minecraft.net*

2.2 Conclusió

Els videojocs descrits anteriorment tenen en comú el factor d'aleatorietat que es busca per aquest projecte, i sobretot la llibertat que ofereix als jugadors. Per això, es pretén desenvolupar el videojoc amb funcionalitats similars als descrits, adaptant-les en el motor Unity i en l'estil del joc propi. També es crearan noves funcionalitats que s'explicaran en el següent punt.

3 Abast

3.1 Objectius i subobjectius

L'objectiu principal del projecte és dissenyar i implementar un videojoc 2D, per així millorar els meus coneixements i la utilització del motor Unity pel desenvolupament de videojocs. Per aconseguir aquest objectiu, s'han de complir els següents subobjectius:

- Dissenyar una arquitectura de software robusta.
- Dissenyar el món del videojoc; jugador, enemics, escenaris, objectes, etc.
- Aprendre sobre el motor Unity amb la documentació i els fòrums disponibles.
- Crear una interfície d'usuari gràfica (GUI).
- Programar les funcionalitats; generació del terreny, sistema de combat, construcció, moviment, artesania, guardat del progrés, etc.
- Implementar música i efectes de so.
- Implementar animacions.
- Desenvolupar una IA pel comportament dels enemics.
- Implementar un sistema d'il·luminació global.

3.2 Obstacles i Riscos

En tot projecte software existeixen obstacles i riscos que s'han de tenir en compte pel correcte desenvolupament d'aquest, tenint solucions i alternatives preparades. A continuació es descriuen aquests possibles riscos i obstacles que poden afectar o endarrerir el projecte:

- **Errors de codi o 'bugs':** És inevitable no tenir errors de codi en algun moment del desenvolupament software, i aquests errors són el que més temps fan perdre. Una solució a aquest risc és classificar i solucionar primer els errors de codi més importants, i ajornar aquells que ja s'ha invertit una quantitat de temps considerable i que podrien afectar a la planificació del projecte.
- **Data de lliurament ajustada:** Hi ha una data fixa pel lliurament del projecte, cosa que significa que qualsevol contratemps en el desenvolupament del projecte pot afectar en la qualitat final del producte, ja sigui per acabar fent menys funcionalitats de les esperades, per errors de codi sense resoldre o simplement per acabar tenint un resultat més pobre de l'esperat. Per resoldre aquest obstacle es seguirà una planificació ben estructurada amb marges de temps per qualsevol problema que pugui ocórrer.

- **Poca experiència amb les eines utilitzades:** És possible que algunes funcionalitats acabin sent més complicades del pensat quan es va planificar el projecte i que portin més temps de l'esperat, ja que no tinc gaire experiència amb la plataforma de desenvolupament Unity i les eines que ofereix. Per sort, Unity compte amb una àmplia documentació que probablement ajudarà a solucionar aquest problema.
- **Pèrdua de dades:** Hi ha el risc que pugui haver-hi una pèrdua de dades per qualsevol causa durant el desenvolupament, per això s'utilitzarà *GitHub* [11], un servei de repositoris en el núvol, per tenir còpies de seguretat diàries del projecte i així minimitzar aquest risc a només un possible dia de pèrdues.

4 Planificació inicial

4.1 Metodologia de treball

La metodologia de treball que s'utilitzarà per al desenvolupament del projecte serà la metodologia *Agile* [12], concretament la *Kanban* [13]. L'objectiu principal d'aquesta metodologia és gestionar i organitzar el flux de treball mitjançant l'administració de la realització de les tasques fins a la seva finalització. Cada tasca estarà dins d'un taulell que comprendrà 5 estats diferents del flux de treball:

- **Per fer:** En aquest estat estan totes les tasques pendents per realitzar-se.
- **En progrés:** En aquest estat es troben totes les tasques que s'estan realitzant en el moment.
- **En avaluació:** En aquest estat estan totes les tasques fetes que s'estan comprovant que funcionin correctament.
- **Completat:** En aquest estat es troben totes les tasques finalitzades que funcionen correctament.
- **Rebutjat:** En aquest estat estan totes les tasques descartades.

4.2 Seguiment

Per tenir un seguiment de les tasques i poder aplicar la metodologia Kanban explicada anteriorment, s'utilitzarà Trello [14], un software d'administració de projectes amb interfície web on es poden crear taulells i tasques/targetes dins d'aquests. En la *Figura 4* podem observar un exemple de com seria un taulell de projecte realitzat amb Trello. També es faran reunions de seguiment cada dues setmanes aproximadament amb la directora del projecte per analitzar i discutir l'evolució del projecte.

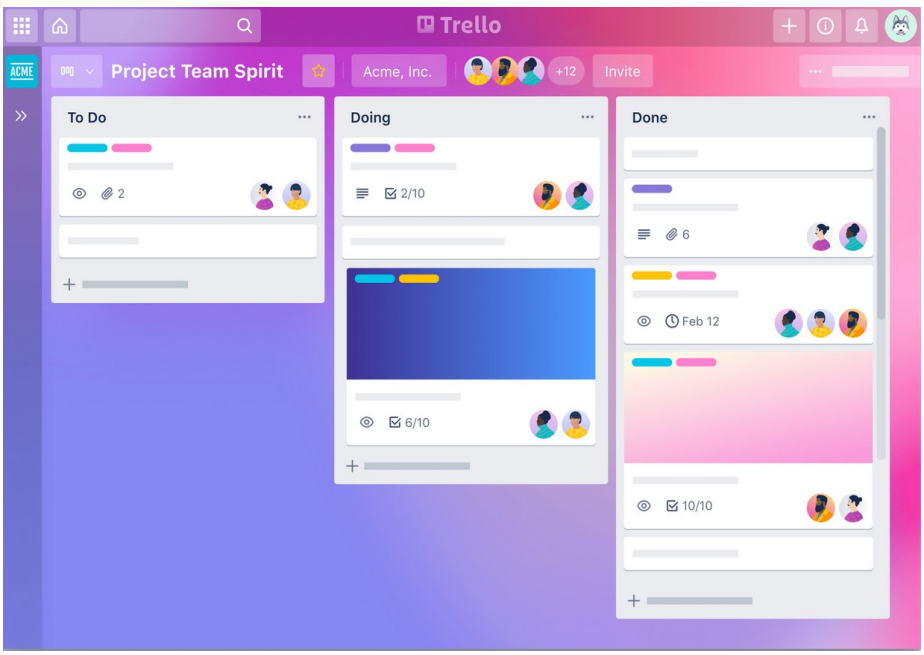


Figura 4: Exemple de taulell Trello
Font: *trello.com*

4.3 Descripció de les tasques

4.3.1 Introducció

La duració del projecte és aproximadament de cinc mesos, des de juliol fins a desembre, sense comptar el mes d'agost. Específicament, es va començar a treballar en el projecte el 16/07/2021, i es preveu la seva finalització la segona setmana de desembre. L'estimació d'hores totals de dedicació per aquest projecte és de 550 hores, argumentades en els següents apartats.

4.3.2 Tasques

A continuació, s'identifiquen totes les tasques que comprendrà aquest projecte. Per cada tasca es proporciona una breu descripció, una estimació en hores de la seva duració, les dependències que pugui tenir i els recursos necessaris per dur-la a terme. En la *Taula 1* es pot observar el resum de la informació de totes les tasques, i en la *Figura 5* el diagrama de Gantt amb l'horari de les tasques.

[T1] Gestió del projecte

En aquest grup de tasques es defineix el context, la planificació, la gestió econòmica i l'organització del projecte. També inclou les reunions al llarg del projecte.

- **[T1.1] Abast i contextualització (25 hores):** Es defineix l'objectiu general del projecte, la contextualització, la justificació de la temàtica, com es desenvoluparà i amb quins mitjans. Sense dependències.
- **[T1.2] Planificació temporal (15 hores):** Es defineixen les diferents tasques que comprenen el projecte i la planificació temporal d'aquestes. Dependències: **[T1.1]**
- **[T1.3] Gestió econòmica i sostenibilitat (15 hores):** Es defineix el pla econòmic del projecte i un informe de sostenibilitat. Dependències: **[T1.2]**
- **[T1.4] Definició del projecte final (20 hores):** S'agrupen les tres tasques anteriors per tenir la definició del projecte final, modificant i millorant les parts que estaven incorrectes. Dependències: **[T1.1]**, **[T1.2]** i **[T1.3]**
- **[T1.5] Gestió tècnica del projecte (10 hores):** Es duen a terme reunions amb la directora del projecte per analitzar i discutir l'evolució d'aquest, i per gestionar i administrar el desenvolupament del projecte. Sense dependències.

[T2] Familiarització amb l'entorn

En aquesta tasca s'amplien els coneixements de l'entorn en el qual es treballarà.

- **[T2.1] Aprendre sobre Unity (30 hores):** S'aprofundeix sobre el motor Unity amb la documentació i els fòrums disponibles. Sense dependències.

[T3] Anàlisi, especificació i disseny del videojoc

En aquest grup de tasques es realitza l'especificació de requisits i es dissenya l'arquitectura del videojoc i els elements gràfics d'aquest, incloent-hi el jugador, els materials, els escenaris i les mecàniques, entre d'altres.

- **[T3.1] Especificació de requisits (25 hores):** Es descriu el comportament del videojoc, incloent-hi el conjunt de casos d'ús, on es descriuen els requisits que ha de cumplir el sistema a desenvolupar, i el model conceptual. També es defineixen les diferents mecàniques del videojoc, és a dir, qualsevol acció realitzada pel jugador que modifiqui l'estat del joc, com el moviment, el combat, la construcció, etc.. Sense dependències.
- **[T3.2] Arquitectura i disseny del videojoc (30 hores):** Es defineix l'arquitectura del software del videojoc, incloent-hi el diagrama de classes i els diagrames de seqüència més importants. Dependències: **[T3.1]** i **[T2.1]**
- **[T3.3] Disseny gràfic de tots els elements (30 hores):** Es dissenyen gràficament tots els elements que es mostraran en el videojoc, com el jugador, els enemics, el terreny, el fons, etc. El joc tindrà un estil *Pixel Art* [15], s'utilitzarà *Pixlr* [16] per la creació dels elements i es buscaran dissenys d'aquest estil ja existents amb ús comercial gratuït. Sense dependències.

[T4] Programació del videojoc

En aquest grup de tasques es programen totes les funcionalitats que tindrà el videojoc.

- **[T4.1] Generació del terreny (30 hores):** Es programa la funcionalitat de generar amb procediments tot el terreny, incloent-hi la generació aleatòria de coves, minerals, estructures i biomes. Dependències: **[T3.2]** i **[T3.3]**
- **[T4.2] Funcionalitats del jugador (50 hores):** Es programen totes les funcionalitats relacionades amb el jugador, com el moviment, la càmera, el sistema de vida i dany, l'inventari, la construcció i destrucció del terreny amb les eines respectives i el combat, entre d'altres menys importants. Dependències: **[T3.3]**
- **[T4.3] Artesania (20 hores):** Es programa la funcionalitat d'artesania, és a dir, la possibilitat d'elaborar objectes del joc mitjançant altres. Dependències: **[T3.3]**
- **[T4.4] Interfície gràfica d'usuari (20 hores):** Es programen les interfícies gràfiques d'usuari necessàries per al videojoc, una per l'inventari, pel menú, per la salut i barra d'eines del jugador i per l'artesania. Dependències: **[T4.2]** i **[T4.3]**
- **[T4.5] Estructures (20 hores):** Es programen les estructures que es poden trobar en el videojoc, com per exemple una piràmide en el bioma del desert. Dependències: **[T4.1]**
- **[T4.6] IA dels enemics (30 hores):** Es programa la intel·ligència artificial per cada un dels enemics del videojoc. Dependències: **[T3.3]**
- **[T4.7] Música i efectes de so (20 hores):** Es busca i s'implementa la música i els efectes de so respectius en el videojoc. Dependències: **[T3.2]**

- **[T4.8] Animacions (15 hores):** Es programen les animacions del jugador i dels enemics segons les seves accions. Dependències: **[T4.2]** i **[T4.6]**
- **[T4.9] Sistema d'il·luminació global (15 hores):** S'implementa un sistema d'il·luminació global amb els *shaders* [17] respectius. Dependències: **[T4.1]**

[T5] Testeig i desplegament

En aquest grup de tasques es testegen les funcionalitats i l'integritat del codi, i es realitza el desplegament del videojocs

- **[T5.1] Testeig de les funcionalitats (30 hores):** Es posen a prova les funcionalitats programades. Dependències: **[T4]** (es farà de manera intermitent durant la programació del videojoc)
- **[T5.2] Testeig d'integritat i desplegament (10 hores):** Es verifica el correcte funcionament del sistema amb totes les funcionalitats implementades, i es fa el desplegament del videojoc perquè estigui disponible pel seu ús. Dependències: **[T5.1]**

[T6] Documentació del projecte

En aquest grup de tasques es documenta tot el projecte i es prepara per a la defensa davant el tribunal.

- **[T6.1] Documentació de la memòria (70 hores):** Es redacta el document final del projecte. Aquesta tasca es farà de manera intermitent durant el desenvolupament del projecte. És dependent de totes les tasques anteriors.
- **[T6.2] Defensa del projecte (20 hores):** Es prepara una presentació per la defensa del projecte davant el tribunal. Dependències: **[T6.1]**

4.3.3 Recursos

A continuació, s'especifiquen i es descriuen els recursos necessaris per dur a terme les tasques descrites anteriorment. Aquests recursos s'han dividit en 3 grups diferents. Entre claudàtors es troben les sigles utilitzades en els recursos de la *Taula 1*.

Recursos humans

- El principal recurs humà és el desenvolupador [DEV], l'encarregat de documentar i desenvolupar el projecte.

- La directora del projecte [D] és l'encarregada de dirigir i supervisar el projecte.

- El tutor de GEP [T] és l'encarregat d'ajudar al desenvolupador a fer la planificació de la gestió del projecte correctament.

Recursos materials

Els recursos imprescindibles per fer el projecte són un ordinador on poder treballar tant amb la programació com amb la redacció del projecte, i un espai de treball, juntament amb els gastos generals d'internet i electricitat [PC]. Pel que fa al software:

- Unity [U]: Plataforma on es desenvolupa el videojoc.

- Visual Studio 2019 [VS]: Entorn de desenvolupament integrat (IDE) per la plataforma Unity.

- Trello [TR]: Pàgina web per administrar el projecte amb la metodologia Kanban.

- GitHub [GH]: Servei de repositoris per tenir còpies de seguretat del projecte.

- GanttProject [GP]: Programa per crear el diagrama de Gantt.

- Apache OpenOffice Writer [W]: Processador de text on es documenta la memòria del projecte.

- Google Meet [GM]: Servei on es realitzen les reunions amb la directora.

- Diagrams.net [DI]: Software per realitzar tota mena de diagrames.

- Pixlr [P]: Editor de fotos en línia, en aquest cas per realitzar el *Pixel Art* esmentat anteriorment.

4.4 Estimacions i diagrama de Gantt

En la següent taula es mostra un resum de la informació de totes les tasques mencionades anteriorment amb el temps estimat, les dependències i els recursos.

CODI	TASCA	TEMPS (HORES)	DEPENDÈNCIES	RECURSOS
T1	**Gestió del projecte**	**85**	-	-
T1.1	Abast i contextualització	25	-	PC, W, T, D, I
T1.2	Planificació temporal	15	T1.1	PC, W, GP, T, D, I
T1.3	Gestió econòmica i sostenibilitat	15	T1.2	PC, W, T, D, I
T1.4	Definició del projecte final	20	T1.1, T1.2, T1.3	PC, W, T, D, I
T1.5	Gestió tècnica del projecte	10	-	PC, GM, D, I
T2	**Familiarització amb l'entorn**	**30**	-	-
T2.1	Aprendre sobre Unity	30	-	PC, U, I
T3	**Anàlisi, especificació i disseny**	**85**	-	-
T3.1	Especificació de requisits	25	-	PC, DI, I
T3.2	Arquitectura i disseny del videojoc	30	T3.1, T2.1	PC, DI, I
T3.3	Disseny gràfic de tots els elements	30	-	PC, P, I
T4	**Programació del videojoc**	**220**	-	-
T4.1	Generació del terreny	30	T3.2, T3.4	PC, U, VS, I
T4.2	Funcionalitats del jugador	50	T3.3, T3.4	PC, U, VS, I
T4.3	Artesania	20	T3.3, T3.4	PC, U, VS, I
T4.4	Interfície gràfica d'usuari	20	T4.2, T4.3	PC, U, VS, I
T4.5	Estructures	20	T4.1	PC, U, VS, I
T4.6	IA dels enemics	30	T3.3, T3.4	PC, U, VS, I
T4.7	Música i efectes de so	20	T3.2	PC, U, VS, I
T4.8	Animacions	15	T4.2, T4.6	PC, U, VS, I
T4.9	Sistema d'il·luminació global	15	T4.1	PC, U, VS, I
T5	**Testeig i desplegament**	**40**	-	-
T5.1	Testeig de les funcionalitats	30	T4	PC, U, VS, I
T5.2	Testeig d'integritat i desplegament	10	T5.1	PC, U, VS, I
T6	**Documentació del projecte**	**90**	-	-
T6.1	Documentació de la memòria	70	T1, T2, T3, T4, T5	PC, TR, W, D, I
T6.2	Defensa del projecte	20	T6.1	PC, W, I
TOTAL	-	**550**	-	-

Taula 1: Resum de la informació de les tasques
Font: *Creació pròpia*

4.4.1 Diagrama de Gantt

A continuació es mostra el diagrama de Gantt de totes les tasques amb la planificació temporal del projecte.

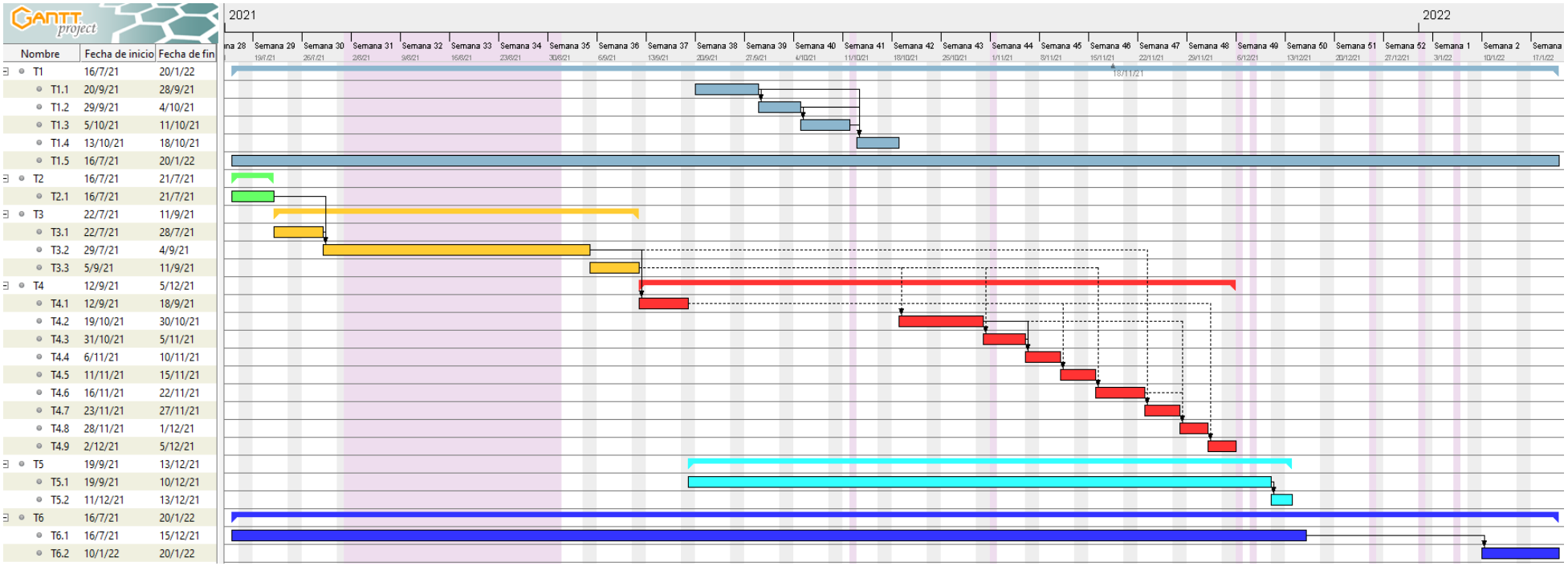


Figura 5: Diagrama de Gantt amb totes les tasques
Font: *Creació pròpia amb el programa GanttProject*

4.5 Gestió del risc: Plans alternatius i obstacles

En l'apartat 3.2 s'han descrit tots els obstacles i riscos que s'han de tenir en compte durant aquest projecte. A continuació es proposen algunes solucions a aquests obstacles:

- **Errors de codi o 'bugs':** El que es pretén per minimitzar aquest risc és classificar i solucionar primer els errors de codi més importants, i ajornar aquells que ja s'ha invertit una quantitat de temps considerable i que podrien afectar a la planificació del projecte. També es pretén anar fent la tasca de "Testeig de les funcionalitats" per cada funcionalitat acabada, per més petita que sigui, o durant el procés d'algunes funcionalitats llargues, per així poder detectar el més abans possible un potencial error de codi i així no endarrerir la planificació pel simple fet de localitzar l'error. Addicionalment es tindran en compte els fòrums disponibles de Unity per preguntar a la comunitat qualsevol problema que estigui portant més temps del previst.
- **Data de lliurament ajustada:** Per resoldre aquest obstacle s'intentarà seguir la planificació descrita en el diagrama de Gantt, i com es pot observar, es deixaran unes tres setmanes de marge per qualsevol problema que pugui endarrerir el projecte. Si així i tot es necessita més temps, s'augmentaran les hores de treball diàries.
- **Poca experiència amb les eines utilitzades:** Unity compte amb una àmplia documentació que ajudarà a solucionar aquest problema, i en tot cas es dedicarien més hores en la tasca de "Aprendre sobre Unity", ja que hi ha unes setmanes de marge per fer-ho.
- **Pèrdua de dades:** S'utilitzarà *GitHub*, un servei de repositoris en el núvol, per tenir còpies de seguretat diàries del projecte i així minimitzar aquest risc a només un possible dia de pèrdues.

Aquests obstacles poden afectar a la duració total del projecte, és per això que s'han establert unes setmanes de marge, com es pot veure en el diagrama de Gantt. També pot afectar amb el consum total de recursos, els quals ja s'han tingut en compte en el següent punt.

4.6 Pressupost

En aquesta secció es defineix el pla econòmic del projecte, identificant i estimant els costos, i definint un control de gestió per controlar les desviacions que poden aparèixer durant el projecte.

4.6.1 Identificació i estimació dels costos

Per identificar els costos del projecte, hem de tenir en compte diversos factors; els recursos humans, el hardware, el software, les despeses generals i els impostos, juntament amb les contingències i imprevistos que puguin ocórrer. A continuació hi ha una estimació dels costos per cada factor.

Recursos humans

Abans d'estimar els costos dels recursos humans, s'han de definir els rols d'aquests. Per començar, tenim el cap de projecte, que és el responsable de planejar i desenvolupar el projecte. També tenim l'analista, que s'encarrega d'experimentar i analitzar resultats, com documentar tot el que implica el desenvolupament i resultats del projecte. Després tenim el programador, que és l'encarregat de programar el codi del videojoc, seguit del provador, el qual verifica en tot moment el correcte funcionament de les funcionalitats implementades. I per acabar està el rol del dissenyador gràfic, que és qui té la funció de dissenyar artísticament l'aparença i la forma del videojoc, incloent-hi els escenaris i les textures, entre d'altres.

En la *Taula 2* podem observar el salari estimat de tots els rols implicats en el desenvolupament del projecte.

Rol	Salari brut (€/any)	Salari brut (€/hora)	Salari brut (€/hora) + Seguretat Social (x1,3)	Realitzat per
Cap de projecte	35.875 [18]	17,25	22,43	T, D, DEV
Analista	28.514 [19]	13,71	17,82	DEV
Arquitecte de software	45.956 [20]	22,09	28,72	DEV
Programador	29.152 [21]	14,02	18,23	DEV
Provador	20.347 [22]	9,78	12,71	DEV
Dissenyador gràfic	22.178 [23]	10,66	13,86	DEV

Taula 2: Salari estimat dels rols del projecte
DEV: Desenvolupador
T: Tutor de GEP
D: Directora del projecte
***Font:** Creació pròpia*

En la *Taula 3* es mostren tots els rols amb les hores que ha de dedicar per realitzar les diferents tasques, i el cost final de l'execució d'aquestes, amb el cost total dels recursos humans.

Codi tasca	Hores	Cap de projecte	Analista	Arquitecte de software	Programador	Provador	Dissenyador gràfic	Cost (€)
T1	**85**	**85**	**0**	**0**	**0**	**0**	**0**	**1.906,55**
T1.1	25	25	0	0	0	0	0	560,75
T1.2	15	15	0	0	0	0	0	336,45
T1.3	15	15	0	0	0	0	0	336,45
T1.4	20	20	0	0	0	0	0	448,6
T1.5	10	10	0	0	0	0	0	224,3
T2	**30**	**0**	**5**	**0**	**25**	**0**	**0**	**544,85**
T2.1	30	0	5	0	25	0	0	544,85
T3	**85**	**0**	**25**	**30**	**0**	**0**	**30**	**1.722,9**
T3.1	25	0	25	0	0	0	0	445,5
T3.2	30	0	0	30	0	0	0	861,6
T3.3	30	0	0	0	0	0	30	415,8
T4	**260**	**0**	**0**	**0**	**220**	**0**	**0**	**4.010,6**
T4.1	30	0	0	0	30	0	0	546,9
T4.2	50	0	0	0	50	0	0	911,5
T4.3	20	0	0	0	20	0	0	364,6
T4.4	20	0	0	0	20	0	0	364,6
T4.5	20	0	0	0	20	0	0	364,6
T4.6	30	0	0	0	30	0	0	546,9
T4.7	20	0	0	0	20	0	0	364,6
T4.8	15	0	0	0	15	0	0	273,45
T4.9	15	0	0	0	15	0	0	273,45
T5	**40**	**0**	**0**	**0**	**0**	**40**	**0**	**508,4**
T5.1	30	0	0	0	0	30	0	381,3
T5.2	10	0	0	0	0	10	0	127,1
T6	**90**	**20**	**70**	**0**	**0**	**0**	**0**	**1.696**
T6.1	70	0	70	0	0	0	0	1.247,4
T6.2	20	20	0	0	0	0	0	448,6
TOTAL	**550**	**105**	**100**	**30**	**245**	**40**	**30**	**10.389,3**

Taula 3: Rols amb les hores i cost per cada tasca
Font: *Creació pròpia*

El cost presentat és una estimació, ja que les hores d'algunes tasques poden variar al llarg del projecte a causa d'imprevistos i obstacles comentats en apartats anteriors, afectant el cost total.

Hardware

En la *Taula 4* es troben els costos estimats en hardware, amb l'amortització corresponent. Aquesta amortització s'ha calculat aplicant la següent fórmula, on *Vida útil estimada en dies* és igual a 4x365 dies, i *Dies d'ús* és igual a 152, els dies que s'espera treballar en el projecte:

*Amortització = (Valor de compra / Vida útil estimada en dies) * Dies d'ús*

Hardware	**Cost (€)**	**Amortització (€)**
Ordinador de sobretaula	1.200	124,93
Monitor	150	15,62
Teclat	50	5,2
Ratolí	20	2,08
TOTAL	**1.420**	**147,84**

Taula 4: Costos estimats en hardware
Font: *Creació pròpia*

Les contingències sobre aquest factor que puguin generar algun problema són gairebé nul·les, ja que hauria d'haver-hi un mal funcionament d'algun component hardware esmentat, i no és gaire probable.

Software

En la *Taula 5* podem observar el software utilitzat en el projecte, el qual és tot d'ús gratuït i no comporta cap cost a tenir en compte.

Software	**Cost (€)**	**Amortització (€)**
Unity	0	0
Visual Studio 2019	0	0
Trello	0	0
GitHub	0	0
GanttProject	0	0
Apache OpenOffice Writer	0	0
Google Meet	0	0
Diagrams.net	0	0
Pixlr	0	0
TOTAL	**0**	**0**

Taula 5: Costos estimats en software
Font: *Creació pròpia*

Despeses generals

A continuació, tenim les despeses generals, o costos indirectes, mostrats a la *Taula 6*. L'amortització s'ha calculat amb la següent fórmula, on *Mesos d'ús del servei* és igual a 5:

*Amortització = Preu mensual * Mesos d'ús del servei*

Recurs	Cost	Amortització (€)
Internet	44 €/mes	220
Electricitat	0,17394 €/kWh	19,85
Entorn de treball	30 €/mes	150
TOTAL	-	**389,85**

Taula 6: Costos estimats en despeses generals
Font: *Creació pròpia*

Imprevistos

Per acabar amb els costos del projecte, a la *Taula 7* tenim els costos dels imprevistos que puguin ocórrer.

Risc	Probabilitat	Hores	Cost (€)	Rol implicat	Cost amb probabilitat (€)
Errors de codi	60%	30	546,9	Programador	328,14
Data ajustada	10%	20	448,6	Cap de projecte	44,86
Poca experiència	20%	10	181,62	Programador	36,32
Pèrdua de dades	5%	5	89,1	Analista	4,46
TOTAL	-	-	-	-	**413,78**

Taula 7: Costos estimats en imprevistos i contingències
Font: *Creació pròpia*

Resumint la informació dels costos, en la *Taula 8* es mostra el cost estimat total del projecte, tenint en compte les contingències d'aquest tipus de projecte, que són aproximadament del 20%, i l'IVA.

Factor	Cost (€)
Recursos humans	10.389,3
Hardware	147,84
Software	0
Despeses generals	389,85
Imprevistos	413,78
Contingències	2.077,86
TOTAL	**13.418,63 + IVA(21%) = 16.236,54**

Taula 8: Costos estimats totals del projecte
Font: *Creació pròpia*

4.6.2 Control de gestió

En aquest punt es defineixen una sèrie fórmules indicatives per controlar les desviacions que puguin aparèixer en relació amb el pressupost, tenint en compte els obstacles i riscos presents en aquest projecte i la metodologia que s'utilitza.

A continuació es mostren les desviacions relacionades amb els recursos humans, les quals són causades per canvis en els salaris o per la realització de les tasques amb més o menys temps que l'esperat:

*Desviació del cost = (Cost estimat - Cost real) * Hores reals*

*Desviació d'hores = (Hores estimades - Hores reals) * Cost real per hora*

Per tenir un seguiment de les desviacions d'hores, quan una tasca estigui completada s'actualitzarà el pressupost tenint en compte les hores emprades en realitzar-la i els imprevistos que hagin ocorregut.

Sobre el hardware, cal tenir en compte les desviacions sobre l'amortització, ja que un recurs es pot utilitzar més o menys i generar així una variació en el pressupost, i sobre les despeses generals també s'han de tenir en compte les hores d'ús d'electricitat i d'internet. Podem aplicar la següent fòrmula en els dos casos:

*Desviació d'amortització = (Hores d'ús estimades – Hores d'ús reals) * Cost real per hora*

Els imprevistos amb les seves respectives probabilitats i les contingències d'aquest tipus de projecte ja s'han tingut en compte en el càlcul del cost estimat total del projecte. Amb aquestes fórmules indicatives podrem saber on estan les desviacions del pressupost i anar comparant-ho amb el cost estimat total per tenir un control del pla econòmic.

4.7 Informe de sostenibilitat

4.7.1 Autoavaluació

Tot projecte, per molt curt que sigui, té implicacions negatives que s'han de tenir en compte i minimitzar el màxim possible. Després de realitzar l'enquesta donada per la UPC, he pres més consciència sobre el tema, el qual estava deixant de costat, i m'he adonat que desconec bastant sobre la sostenibilitat en un projecte software com aquest, així que de cada camp he extret unes conclusions que m'ajudaran a tenir present i millorar l'impacte econòmic, ambiental i social.

- Pel que fa al camp econòmic, conec com realitzar un pla econòmic i saber si és viable o no, com calcular i analitzar els costos, i com entendre l'impacte econòmic que tindrà el projecte. Però durant la realització del pla econòmic he intentat minimitzar el cost total sense tenir en compte la relació amb els altres camps, els quals són igual d'importants que aquest.
- Pel que fa al camp ambiental, tinc pocs coneixements sobre la majoria d'indicadors, i per tant conec poc les conseqüències d'utilitzar hardware i dissenyar nou software. En el projecte s'intentarà reduir l'ús del hardware i tenir un rendiment molt bo per l'ús del producte software resultant, per així minimitzar la petjada ecològica d'aquest.
- Pel que fa al camp social, conec algunes mètriques per mesurar l'impacte social del producte software que es vol realitzar, però desconec com utilitzar-les per millorar el projecte. Comprenc els conceptes relacionats amb la salut, seguretat i justícia social i intento millorar els conceptes relacionats amb el meu projecte, com l'ergonomia, l'accessibilitat i l'experiència d'usuari.

4.7.2 Dimensió Econòmica

Per estimar el cost total del projecte s'han tingut en compte els recursos humans i materials, incloent-hi el hardware, el software, les despeses generals i els imprevistos. També s'han contemplat les possibles desviacions amb les fórmules indicatives a aplicar-se per tenir un control de la gestió econòmica.

Per minimitzar els aspectes de costos s'utilitzarà en tot moment software d'ús lliure, com s'ha pogut veure en l'apartat del pressupost, i també s'utilitzaran alguns dissenys gràfics d'ús lliure comercial.

Una de les coses en què aquest projecte pot millorar econòmicament respecte a videojocs ja existents és un millor rendiment per una reducció del consum d'electricitat, i la utilització d'eines ja existents, com l'Animator de Unity [26], per així reduir les hores de treball dels recursos humans.

Tot i no ser un objectiu principal d'aquest projecte, l'opció de treure el videojoc al mercat un cop acabat no està descartada, i per tant, això comportaria un benefici econòmic posterior a la realització d'aquest.

4.7.3 Dimensió Ambiental

L'impacte ambiental d'aquest projecte es basa en la utilització dels recursos hardware, ja que aquests consumeixen energia. L'únic recurs hardware que s'utilitzarà és un ordinador de sobretaula, amb els components necessaris perquè aquest faci la seva funció, com el monitor, el ratolí i el teclat, i per tant no es podran reutilitzar recursos per reduir aquest impacte. Però això no implica que sigui negatiu, ja que significa que el projecte té un impacte ambiental bastant baix.

Un cop dissenyat i implementat el videojoc, estarà disponible en una plataforma digital, és a dir, aquest ja no consumirà més recursos per si sol, a part de l'energia necessària per executar i utilitzar el software. Per tant, l'únic que es pot millorar ambientalment respecte a altres videojocs és el rendiment perquè el videojoc no faci un ús excessiu dels components on s'executi, i això provoqui un consum més elevat d'electricitat o malmeti aquests components.

Això sí, el fet que el videojoc estigui publicat en una plataforma de distribució de videojocs també tindrà un impacte ambiental, per molt petit que sigui, ja que la gestió per mantenir-lo en línia perquè els usuaris el puguin comprar, descarregar i jugar, a la vegada que guardar el progrés de manera local, que en aquest cas no afectaria l'impacte ambiental, o guardar-lo al núvol de la plataforma específica, afectarà el consum d'energia dels servidors de la plataforma en concret.

4.7.4 Dimensió Social

Aquest projecte m'aportarà experiència i coneixements sobre el motor de Unity i sobre el desenvolupament de videojocs i tot el que comporta.

El que es busca amb aquest videojoc és donar una llibertat completa al jugador, per així millorar la qualitat de vida respecte a videojocs ja existents. També es busca recompensar a aquells jugadors que vulguin un videojoc on puguin ser creatius i no tinguin restriccions.

Aquest projecte va dirigit a la indústria de l'entreteniment, i per aquest motiu no crec que existeixi una necessitat real per realitzar-lo, a part del coneixement i experiència que guanyaré. Però sí que pot ajudar a la qualitat de vida d'algunes persones que necessitin desconnectar jugant a videojocs. Actualment encara estem en la pandèmia de la COVID-19, i l'oci digital s'ha vist incrementat per diverses raons, com el confinament o l'obligació de treballar des de casa. Aquest videojoc pot ajudar a satisfer aquest oci digital que avui en dia es consumeix en grans proporcions.

5 Especificació de requisits

En aquesta secció s'especifiquen amb detall els requisits funcionals i no funcionals del videojoc. També es mostra el diagrama de casos d'ús juntament amb una especificació completa per cada cas d'ús de l'aplicació, i el model conceptual del videojoc seguit dels diagrames de seqüència de les funcionalitats més importants. Finalment s'exposa el document de disseny del videojoc, mostrant i explicant els diferents elements gràfics amb les mecàniques que han de tenir dins del videojoc.

5.1 Requisits

A continuació es descriuen els requisits funcionals i no funcionals de l'aplicació, donant a conèixer en profunditat les funcionalitats que es desenvoluparan i la informació necessària sobre el funcionament del videojoc. A causa de la metodologia de treball que s'utilitza, alguns requisits poden variar al llarg del projecte. La prioritat indica la importància del requisit, i s'especifica com baixa, mitjana o alta.

La majoria d'aquests requisits s'han obtingut després d'analitzar diversos videojocs i entendre el funcionament d'aquests amb les idees que presenten inicialment, d'altres han sigut idees que han sorgit d'algunes mecàniques de videojocs existents, però millorades o adaptades a aquest videojoc, i alguns requisits han sigut idees pròpies que ja tenia pensades en implementar.

Aquests requisits s'han prioritzat mitjançant la importància que tenen dins del videojoc, tenint en compte que algunes funcionalitats són essencials per poder jugar o es necessiten per complir els objectius principals, i per tant tenen una prioritat alta, i algunes són millores de la qualitat de vida del videojoc o mecàniques que no són imprescindibles, i per tant tenen una prioritat baixa o mitjana.

5.1.1 Requisits funcionals

Codi - Nom	**RF01 - Moviment del personatge**
Descripció	El sistema ha de poder moure el personatge que controla el jugador en qualsevol direcció, ja sigui sobre l'eix horitzontal (caminar i córrer) o sobre l'eix vertical (saltar i caure).
Dependències	-
Prioritat	Alta
Informació extra	El jugador pot saltar i moure el personatge cap a l'esquerra i dreta amb les tecles respectives. Si el jugador vol augmentar la velocitat del personatge, ho pot fer mantenint pressionada la tecla respectiva. Si el personatge no està sobre el terreny, aquest caurà fins a estar en col·lisió amb el terreny de nou.

Taula 9: RF01 – Moviment del personatge

Codi - Nom	RF02 – Seguiment del personatge
Descripció	El sistema ha de seguir el personatge de manera suau amb la càmera principal.
Dependències	-
Prioritat	Alta
Informació extra	-

Taula 10: RF02 – Seguiment del personatge

Codi - Nom	RF03 – Vida del personatge
Descripció	El sistema ha de tenir una barra de salut pel personatge que disminuirà segons el dany dels enemics i s'anirà regenerant amb el temps o amb alguns objectes específics.
Dependències	-
Prioritat	Alta
Informació extra	El personatge ha de tenir 40 punts de vida distribuïts en 10 cors, és a dir, cada cor comprèn 4 punts de vida. El jugador pot saber la vida actual del personatge mitjançant la interfície gràfica dels cors, els quals han de tenir 5 fases; cor complet, tres quarts de cor, mig cor, un quart de cor i cor buit. La vida s'ha de calcular amb decimals i s'ha de representar amb els cors, truncant el decimal a la unitat, és a dir, si el jugador rep 2,5 de dany, els cors totals que tindria serien 9 cors complets més mig cor. Si rebés de nou 2,5 de dany, el personatge tindria 8 cors més tres quarts de cor, acumulant el 0,5 anterior.

Taula 11: RF03 – Vida del personatge

Codi - Nom	RF04 – Regeneració de vida del personatge
Descripció	El sistema ha de permetre al personatge regenerar vida amb el temps i amb alguns objectes específics.
Dependències	RF03
Prioritat	Mitjana
Informació extra	La regeneració de vida només ha d'estar activa quan el personatge no rep dany durant una determinada quantitat de temps. Els objectes que regeneren salut l'han de regenerar instantàniament.

Taula 12: RF04 – Regeneració de vida del personatge

Codi - Nom	RF05 – Temperatura del personatge
Descripció	El sistema ha de tenir un indicador de temperatura pel personatge que augmentarà o disminuirà depenent del bioma en el qual es trobi.
Dependències	-
Prioritat	Mitjana
Informació extra	En el bioma de neu el personatge ha de tenir la temperatura mínima, i en el bioma de desert ha de tenir la temperatura màxima. Aquesta temperatura pot equilibrar-se amb objectes específics.

Taula 13: RF05 – Temperatura del personatge

Codi - Nom	RF06 – Dany del personatge
Descripció	El sistema ha de permetre al personatge rebre dany de caiguda, d'enemics i de temperatura.
Dependències	-
Prioritat	Alta
Informació extra	El dany de caiguda s'ha de calcular amb la velocitat vertical del personatge. El dany dels enemics varia segons l'enemic. El dany de temperatura ha de ser constant. Si el personatge té la temperatura mínima, ha de rebre 2 de dany per segon. Si el personatge té la temperatura màxima, ha de rebre 4 de dany per segon.

Taula 14: RF06 – Dany del personatge

Codi - Nom	RF07 – Mort i reaparició del personatge
Descripció	El sistema ha de permetre al personatge morir, perdent així alguns objectes, i reaparèixer de nou en el punt de reaparició establert.
Dependències	RF03 i RF06
Prioritat	Alta
Informació extra	El personatge ha de morir quan la salut és inferior o igual a 0. El personatge ha de perdre tots els objectes menys els de la barra principal. Si el personatge arriba en el lloc on ha mort, ha de poder recuperar aquests objectes. El punt de reaparició ha de ser el mateix que el punt on comença el joc. Aquest punt de reaparició ha de poder canviar-se amb algun objecte específic.

Taula 15: RF07 – Mort i reaparició del personatge

Codi - Nom	RF08 – Inventari
Descripció	El sistema ha de tenir un inventari pels materials, objectes i eines del personatge.
Dependències	-
Prioritat	Alta
Informació extra	L'inventari s'ha de poder obrir amb la tecla respectiva. L'inventari ha de tenir 50 espais disponibles, distribuïts en 5 files. Els elements de l'inventari s'han de poder moure i organitzar per aquest. La primera fila ha de ser la barra principal del jugador, la qual ha d'estar visible sense l'inventari obert i cada espai ha de tenir una drecera amb una tecla per utilitzar el material, objecte o eina que contingui aquell espai.

Taula 16: RF08 - Inventari

Codi - Nom	RF09 - Eines
Descripció	El sistema ha de tenir un conjunt d'eines que han de servir per realitzar diferents accions.
Dependències	-
Prioritat	Alta (martells prioritat baixa)
Informació extra	Espases: Han de servir per danyar als enemics. Pics: Han de servir per destruir el terreny, com pedra, terra, minerals, entre d'altres. Destrals: Han de servir per destruir arbres i fusta. Martells: Han de servir per destruir parets. Les parets han d'estar en el fons, darrere del personatge, per així crear una sensació de profunditat. El jugador ha de poder utilitzar una eina amb la tecla respectiva, si aquesta està en la barra principal de l'inventari i està seleccionada.

Taula 17: RF09 - Eines

Codi - Nom	RF10 – Construir i destruir
Descripció	El sistema ha de permetre al jugador construir amb materials del món i eliminar-los de l'inventari, i destruir amb les eines adequades i afegir-los a l'inventari.
Dependències	RF08 i RF09 (destruir)
Prioritat	Alta
Informació extra	El jugador no ha de poder construir a l'aire, ha d'haver-hi un bloc adjacent o una paret per poder construir. Els blocs s'han de deixar caure un cop destruïts, i s'han de poder recollir automàticament quan el personatge s'apropa suficient, afegint-los a l'inventari.

Taula 18: RF10 – Construir i destruir

Codi - Nom	RF11 – Fabricació
Descripció	El sistema ha de permetre al jugador obrir un menú de fabricació per elaborar diferents objectes i eines amb altres materials i objectes.
Dependències	-
Prioritat	Mitjana
Informació extra	En el menú de fabricació han de sortir totes les receptes disponibles amb els materials i objectes que el jugador té a l'inventari. Els materials que s'utilitzin per a l'elaboració d'un objecte o eina s'han d'eliminar de l'inventari.

Taula 19: RF11 – Fabricació

Codi - Nom	RF12 – Interfície d'usuari gràfica
Descripció	El sistema ha de tenir una interfície d'usuari gràfica pel menú, per la salut, per la temperatura, per la mort del personatge, per l'inventari, per la barra principal del jugador i pel menú de fabricació.
Dependències	RF03, RF05, RF07, RF08 i RF11
Prioritat	Alta
Informació extra	El jugador ha de poder interactuar amb aquestes interfícies gràfiques.

Taula 20: RF12 – Interfície d'usuari gràfica

Codi - Nom	RF13 – Generació del terreny
Descripció	El sistema ha de poder generar amb procediments tot el terreny quan el jugador crea un nou món.
Dependències	-
Prioritat	Alta
Informació extra	El terreny és tot el món on el jugador jugarà al videojoc. Per aquesta raó, està previst que el terreny trigui un temps a generar-se. En aquest temps, el sistema ha de mostrar una pantalla de càrrega. Aquest requisit es pot dividir en els subrequisits següents: • Generació aleatòria de coves. • Generació aleatòria de minerals segons la profunditat. • Generació aleatòria d'estructures. • Generació aleatòria de biomes. Tot el terreny ha d'estar dividit en trossos de 16 blocs d'amplada, els quals s'han d'anar activant i desactivant depenent de la distància amb el personatge per millorar el rendiment del videojoc.

Taula 21: RF13 – Generació del terreny

Codi - Nom	RF14 – Món en bucle
Descripció	El sistema ha de connectar el final del món amb el principi, i viceversa, de manera que simuli el món en bucle.
Dependències	RF13
Prioritat	Baixa
Informació extra	Quan el personatge arriba al final del món, el tros de 16 blocs d'amplada del principi del món ha d'aparèixer al final, i així successivament, simulant el bucle. Ha de passar el mateix a la inversa si el jugador va al principi del món.

Taula 22: RF14 – Món en bucle

Codi - Nom	RF15 – Col·lisió d'objectes
Descripció	El sistema ha de tenir una gestió de col·lisions entre el personatge, els enemics i el terreny.
Dependències	-
Prioritat	Alta
Informació extra	La col·lisió entre un enemic i el personatge ha de generar dany al personatge. El personatge i els enemics s'han de poder moure pel terreny.

Taula 23: RF15 – Col·lisió d'objectes

Codi - Nom	RF16 – Enemics
Descripció	El sistema ha de tenir diversos enemics amb vida, dany i disseny gràfic diferents.
Dependències	-
Prioritat	Mitjana
Informació extra	-

Taula 24: RF16 – Enemics

Codi - Nom	RF17 – Animacions
Descripció	El sistema ha de tenir una gestió d'animacions de les diferents accions del personatge i dels enemics.
Dependències	RF01, RF06 i RF16
Prioritat	Mitjana
Informació extra	Les accions del jugador que han de tenir animacions són córrer, caminar, saltar, caure, utilitzar una eina i fer-se dany. Les accions dels enemics que han de tenir animacions són fer dany, caminar, saltar i fer-se dany.

Taula 25: RF17 – Animacions

Codi - Nom	RF18 – Aparició d'enemics
Descripció	El sistema ha de poder generar enemics fora del rang de la càmera i depenent de la zona en la qual es trobi el jugador.
Dependències	RF16
Prioritat	Mitjana
Informació extra	Els enemics s'han d'eliminar si el tros de 16 blocs d'amplada on es troben es desactiva. No han de reaparèixer més enemics si ja hi ha 20 enemics en pantalla.

Taula 26: RF18 – Aparició d'enemics

Codi - Nom	RF19 – Intel·ligència artificial per cada enemic
Descripció	El sistema ha de tenir una intel·ligència artificial per cada enemic.
Dependències	RF16
Prioritat	Mitjana
Informació extra	Tots els enemics s'han de moure en direcció el jugador.

Taula 27: RF19 – Intel·ligència artificial per cada enemic

Codi - Nom	RF20 – Reproducció de música i efectes de so
Descripció	El sistema ha de poder reproduir música de fons corresponent al bioma en el qual es trobi el personatge i efectes de so corresponents al que estigui passant, com danyar a un enemic en concret, picar un bloc o rebre dany.
Dependències	-
Prioritat	Alta
Informació extra	-

Taula 28: RF20 – Reproducció de música i efectes de so

Codi - Nom	RF21 – Gestió de música i efectes de so
Descripció	El sistema ha de tenir l'opció d'activar, desactivar i canviar els valors del volum de la música i els efectes de so.
Dependències	RF20
Prioritat	Baixa
Informació extra	-

Taula 29: RF21 – Gestió de música i efectes de so

Codi - Nom	RF22 – Il·luminació global
Descripció	El sistema ha de gestionar automàticament la il·luminació del joc.
Dependències	-
Prioritat	Mitjana
Informació extra	-

Taula 30: RF22 – Il·luminació global

Codi - Nom	RF23 – Pausar el videojoc
Descripció	El sistema ha de tenir l'opció d'aturar el videojoc en qualsevol moment.
Dependències	-
Prioritat	Mitjana
Informació extra	El jugador ha de poder aturar el joc obrint l'inventari, el menú de fabricació o el menú general.

Taula 31: RF23 – Pausar el videojoc

Codi - Nom	RF24 – Guardar el progrés
Descripció	El sistema ha de permetre al jugador guardar el progrés en qualsevol moment.
Dependències	-
Prioritat	Mitjana
Informació extra	-

Taula 32: RF24 – Guardar el progrés

Codi - Nom	RF25– Sortir de l'aplicació
Descripció	El sistema ha de permetre al jugador sortir de l'aplicació en qualsevol moment.
Dependències	-
Prioritat	Alta
Informació extra	-

Taula 33: RF25 – Sortir de l'aplicació

Codi - Nom	RF26– Parallax Scrolling
Descripció	El sistema ha de poder moure les imatges del fons més lentament per crear una il·lusió de profunditat en l'escena 2D.
Dependències	-
Prioritat	Baixa
Informació extra	-

Taula 34: RF26 – Parallax Scrolling

5.1.2 Requisits no funcionals

Codi - Nom	RNF01– Rendiment
Descripció	El sistema ha d'oferir un bon rendiment, carregant i responent a les entrades del jugador en el mínim temps possible i mantenint 60 fotogrames per segon (fps) estables. Els requisits mínims de hardware s'han de situar al voltant de 256 MB de RAM.
Justificació	És important que el jugador noti la fluïdesa quan estigui jugant i tingui resposta ràpida a les accions que realitzi per gaudir al màxim del videojoc.
Condició de satisfacció	Es deixarà provar el videojoc acabat a diversos usuaris i el 80% o més no tindrà cap problema de rendiment.
Prioritat	Alta

Taula 35: RNF01 – Rendiment

Codi - Nom	RNF02 – Interfície gràfica atractiva
Descripció	El sistema ha d'oferir interfícies agradables i intuïtives pel jugador, i un disseny artístic general del videojoc atractiu.
Justificació	Una de les principals atraccions d'un videojoc és la part artística i visual d'aquest.
Condició de satisfacció	Es mostraran les diverses interfícies gràfiques i l'art del videojoc a diversos usuaris i el 80% o més les trobaran atractives i intuïtives.
Prioritat	Alta

Taula 36: RNF02 – Interfície gràfica atractiva

Codi - Nom	RNF03 – Emmagatzematge
Descripció	El sistema necessita un mínim de 200 MB d'espai lliure al disc dur.
Justificació	Com menys espai ocupi el videojoc, millor per l'usuari.
Condició de satisfacció	La instal·lació completa del videojoc no supera els 200 MB.
Prioritat	Alta

Taula 37: RNF03 – Emmagatzematge

Codi - Nom	RNF04 – Multiplataforma
Descripció	El sistema ha de ser compatible entre els sistemes operatius Windows, Linux i macOS, i la resolució de la pantalla s'ha de poder ajustar.
Justificació	Per arribar a un públic més gran d'usuaris.
Condició de satisfacció	El videojoc es pot executar en els diferents sistemes operatius i s'adapta a diferents resolucions de pantalla.
Prioritat	Mitjana

Taula 38: RNF04 – Escalabilitat

Codi - Nom	RNF05 – Multidioma
Descripció	El sistema ha d'oferir el videojoc en els idiomes anglès i espanyol.
Justificació	Per arribar a un públic més gran d'usuaris.
Condició de satisfacció	El videojoc té disponibles els idiomes anglès i espanyol.
Prioritat	Mitjana

Taula 39: RNF05 – Multidioma

5.2 Casos d’ús

En aquest apartat es descriuen les interaccions del sistema amb l’usuari, en aquest cas el jugador, per aconseguir comportaments específics.

A continuació es mostra el diagrama de casos d’ús, el qual especifica la comunicació i el comportament del sistema mitjançant la interacció de l’usuari, i seguidament s’especifiquen aquests casos d’ús amb detall.

5.2.1 Diagrama de casos d’ús

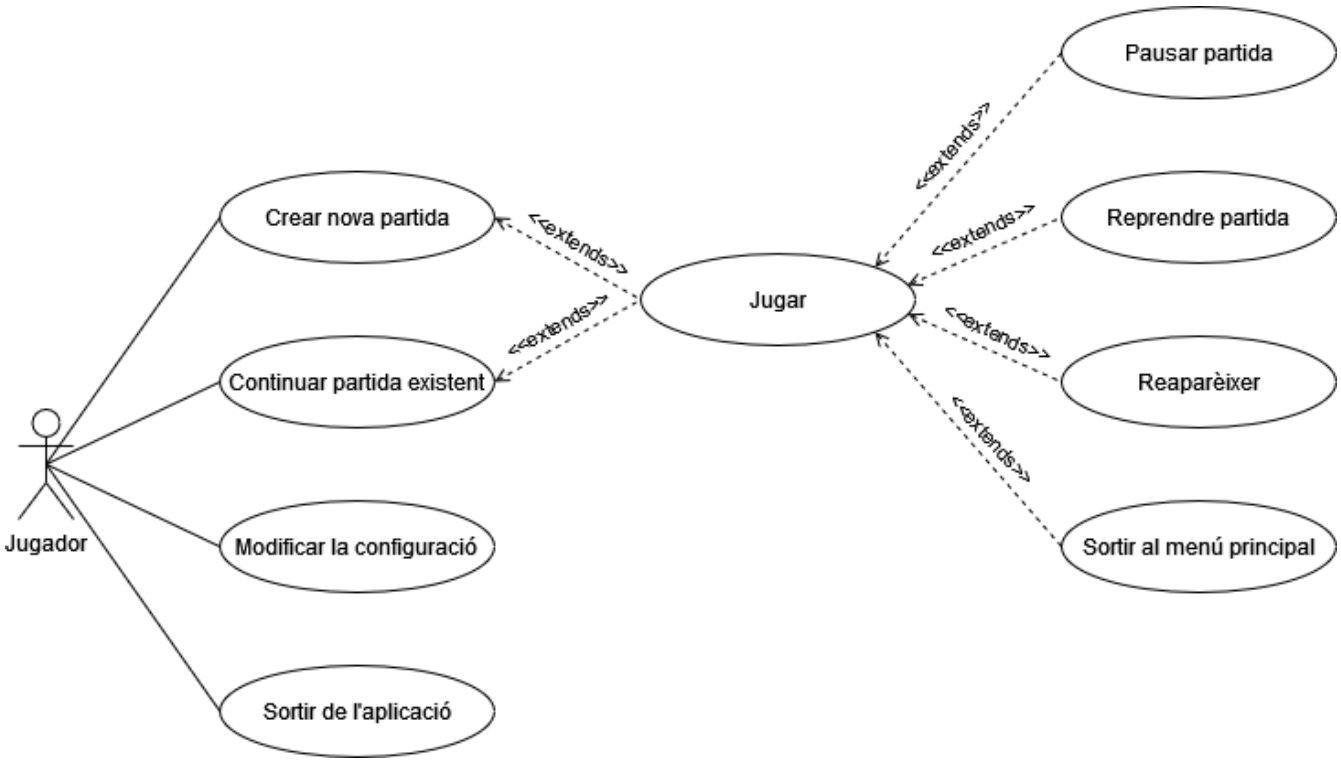


Figura 6: Diagrama de casos d’ús
Font: *Creació pròpia*

5.2.2 Especificació completa de casos d'ús

Codi - Cas d'ús	**CU01 – Crear nova partida**
Actor principal	Jugador
Precondició	El jugador està en el menú principal del joc
Disparador	El jugador vol crear una nova partida.
Escenari principal d'èxit	
1. El jugador selecciona l'opció de crear una nova partida en el menú principal. 2. El sistema mostra els espais de guardat disponibles. 3. El jugador selecciona un espai de guardat per la creació de la partida. 4. El sistema mostra tres opcions per la creació del món: gran, mitjà i petit. 5. El jugador selecciona una de les tres opcions. 6. El sistema genera el terreny. 7. El jugador ja pot començar a jugar.	
Extensions	
3a. El jugador selecciona un espai de guardat que conté una partida existent. 3a1. El sistema notifica al jugador si vol sobreescriure l'espai de guardat amb una nova partida, eliminant la partida existent. 3a2. Si el jugador accepta, va al punt 4. Si cancel·la, torna al punt 3. 3b, 4a. El jugador cancel·la la creació de la partida. 3b1, 4a1. El sistema dirigeix al jugador al menú principal. Fi del cas d'ús.	

Taula 40: CU01 – Crear nova partida

Codi - Cas d'ús	**CU02 – Continuar partida existent**
Actor principal	Jugador
Precondició	El jugador es troba en el menú principal del joc
Disparador	El jugador vol continuar una partida existent
Escenari principal d'èxit	
1. El jugador selecciona l'opció de continuar una partida existent en el menú principal. 2. El sistema mostra els espais de guardat disponibles. 3. El jugador selecciona l'espai de guardat on es troba la partida existent. 4. El sistema carrega la partida. 5. El jugador ja pot començar a jugar.	
Extensions	
3a. El jugador selecciona un espai de guardat que no conté una partida existent. 3a1. El sistema notifica al jugador si vol crear una nova partida en l'espai de guardat seleccionat. 3a2. Si el jugador accepta, el jugador va al cas d'ús CU01 al punt 3. Si cancel·la el jugador torna al punt 2. 3b. El jugador cancel·la la continuació d'una partida existent. 3b1. El sistema dirigeix al jugador al menú principal. Fi del cas d'ús.	

Taula 41: CU02 – Continuar partida existent

Codi - Cas d'ús	CU03 – Jugar
Actor principal	Jugador
Precondició	El jugador ha continuat una partida existent o ha creat una nova partida
Disparador	El jugador vol jugar
Escenari principal d'èxit	
1. El sistema carrega la partida. 2. El jugador ja pot jugar, realitzant accions com moure el personatge, saltar, utilitzar les eines, construir i destruir el terreny, etc.	
Extensions	
-	

Taula 42: CU03 – Jugar

Codi - Cas d'ús	CU04 – Pausar partida
Actor principal	Jugador
Precondició	El jugador està dins d'una partida
Disparador	El jugador vol pausar la partida
Escenari principal d'èxit	
1. El jugador selecciona l'opció de pausar la partida amb la tecla corresponent, obrint l'inventari o obrint el menú del joc. 2. El sistema estableix l'escala del temps a 0, s'atura la partida i s'obre l'inventari o el menú.	
Extensions	
2a. El jugador intenta pausar una partida que ja està pausada amb la tecla corresponent, és a dir, el jugador ja es trobava en el menú o en l'inventari. 2a1. El jugador va al cas d'ús CU05 al punt 2.	

Taula 43: CU04 – Pausar partida

Codi - Cas d'ús	CU05 – Reprendre partida
Actor principal	Jugador
Precondició	El jugador es troba dins d'una partida
Disparador	El jugador vol reprendre la partida
Escenari principal d'èxit	
1. El jugador selecciona l'opció de reprendre la partida amb la tecla corresponent, tancant l'inventari o tancant el menú del joc. 2. El sistema estableix l'escala del temps a 1 i es reprèn la partida, tancant l'inventari o menú.	
Extensions	
2a. El jugador intenta reprendre una partida que no està pausada amb la tecla corresponent. 2a1. El jugador va al cas d'ús CU04 al punt 2.	

Taula 44: CU05 – Reprendre partida

Codi - Cas d'ús	CU06 – Reaparèixer
Actor principal	Jugador
Precondició	El jugador es troba dins d'una partida i el personatge que controla el jugador no té punts de salut
Disparador	El jugador vol fer reaparèixer el personatge
Escenari principal d'èxit	
1. El jugador selecciona l'opció de reaparèixer en el menú que li ha sortit quan el personatge que controla ha mort. 2. El sistema fa aparèixer el personatge en el punt de reaparició establert amb la meitat de punts màxims de salut i el jugador pot continuar jugant.	
Extensions	
1a. El jugador selecciona l'opció de guardar i sortir en el menú que apareix en morir. 1a1. El sistema retorna al jugador al menú principal.	

Taula 45: CU06 – Reaparèixer

Codi - Cas d'ús	CU07 – Modificar la configuració
Actor principal	Jugador
Precondició	El jugador es troba en el menú principal o en el menú de dins del joc
Disparador	El jugador vol modificar la configuració
Escenari principal d'èxit	
1. El jugador selecciona l'opció de la configuració en el menú principal o en el menú de dins del joc. 2. El sistema mostra les diferents opcions de configuració d'idioma, de música i efectes de so. 3. El jugador modifica els valors que vol canviar. 4. El sistema aplica els canvis realitzats.	
Extensions	
3a. El jugador selecciona l'opció de restablir les opcions predeterminades. 3a1. El sistema restableix la configuració de l'aplicació. 3a2. El jugador torna al punt 2.	

Taula 46: CU07 – Modificar la configuració

Codi - Cas d'ús	CU08 – Sortir al menú principal
Actor principal	Jugador
Precondició	El jugador està dins d'una partida
Disparador	El jugador vol sortir al menú principal
Escenari principal d'èxit	
1. El jugador presiona la tecla corresponent per obrir el menú. 2. El sistema mostra el menú de la partida. 3. El jugador selecciona l'opció del menú de sortir de la partida. 4. El sistema guarda el progrés i retorna al jugador al menú principal.	
Extensions	
-	

Taula 47: CU08 – Sortir al menú principal

Codi - Cas d'ús	CU09 – Sortir de l'aplicació
Actor principal	Jugador
Precondició	El jugador es troba en el menú principal
Disparador	El jugador vol sortir de l'aplicació
Escenari principal d'èxit	
1. El jugador selecciona l'opció de sortir de l'aplicació en el menú principal. 2. El sistema tanca l'aplicació.	
Extensions	
-	

Taula 48: CU09 – Sortir de l'aplicació

5.3 Model conceptual del videojoc

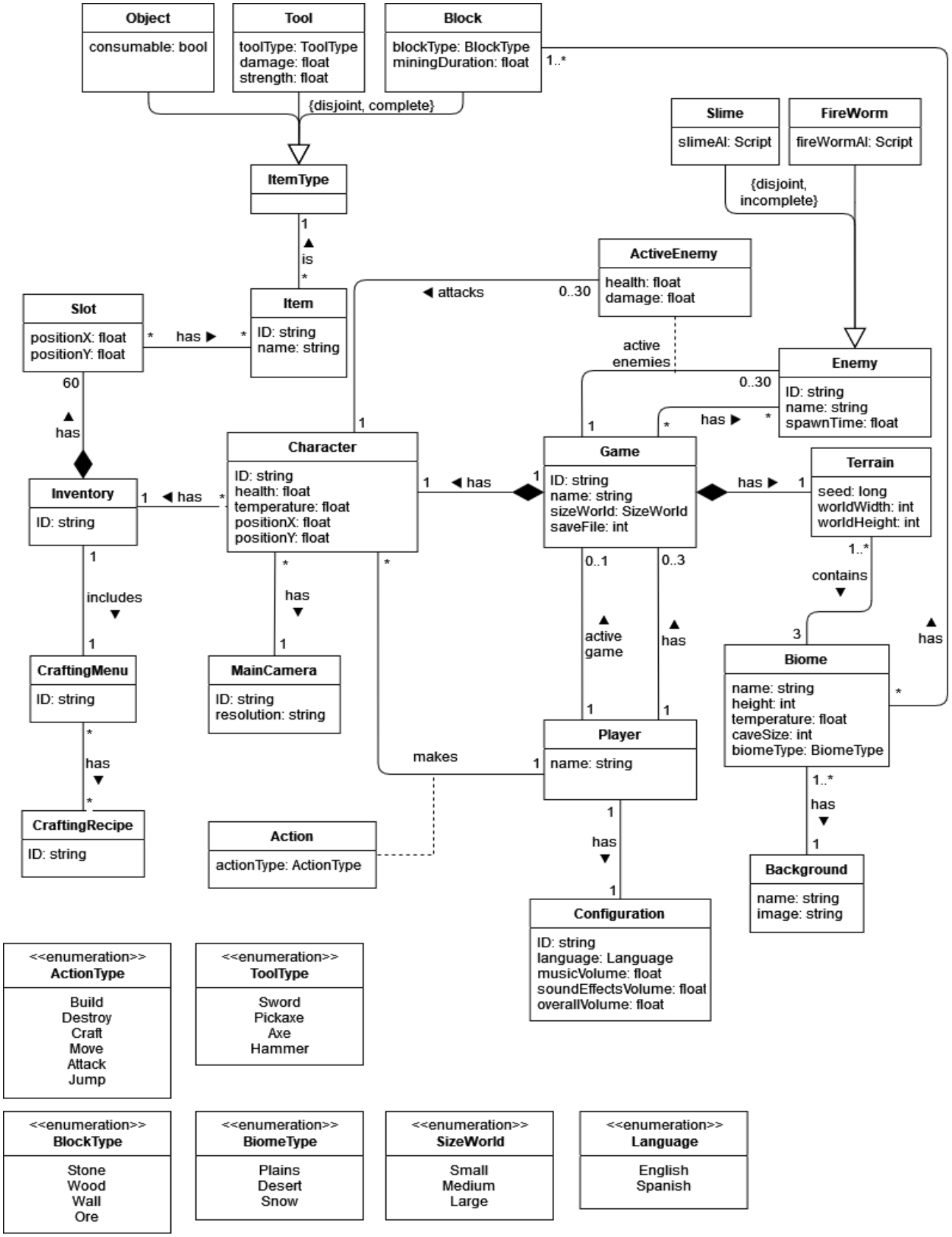


Figura 7: Model conceptual del videojoc
Font: *Creació pròpia*

5.3.1 Restriccions textuals

RT1 – Claus externes: (Player: name), (Game: ID), (Character: ID), (MainCamera: ID), (Background: name), (Configuration: ID), (CraftingMenu: ID), (Item: ID), (Enemy: name), (Terrain: seed), (Biome: name), (Inventory: ID), (Slot: positionX, positionY).

RT2 – El valor de l'atribut *health* de Player i d'Enemy han de ser majors o iguals que zero.

RT3 – El valor de l'atribut *miningDuration* de Block ha de ser major o igual que zero.

RT4 – Els valors dels atributs *damage* i *strength* de Tool han de ser majors o iguals que zero.

RT5 – Els valors dels atributs *worldWidth* i *worldHeight* han de ser 256, 512 o 1024.

RT6 – La relació *attacks* d'un ActiveEnemy a un Character ha de ser del Game del Character.

RT7 – La relació *active game* 0..1 de Player i Game està inclosa en la relació *has* 0..3 de Player i Game.

RT8 – Els Items d'un Slot han de ser del mateix tipus. Si l'Item d'un Slot es una Tool, no es pot tenir més d'1.

RT9 – Un Player només pot controlar un Character del seu *active game.*

5.3.2 Descripció de les entitats

Player

Representa el jugador del videojoc.

Atribut	Tipus	Descripció
name	String	Identificador únic en el sistema.

Taula 49: Player

Game

Representa una partida del videojoc.

Atribut	Tipus	Descripció
ID	String	Identificador únic de la partida en el sistema.
name	String	Nom complet de la partida.
sizeWorld	SizeWorld	Tamany del terreny de la partida.
saveFile	Integer	Número de l'espai de guardat de la partida.

Taula 50: Game

Configuration

Representa la configuració de l'idioma i els volums de música i efectes de so del videojoc.

Atribut	Tipus	Descripció
ID	Animator	Identificador únic de la configuració en el sistema.
language	Language	Idioma del sistema.
musicVolume	Float	Volum de la música.
soundEffectsVolume	Float	Volum dels efectes de so.
overallVolume	Float	Volum general del sistema.

Taula 51: Configuration

Character

Representa el personatge que el jugador controla.

Atribut	Tipus	Descripció
ID	String	Identificador únic del personatge en el sistema.
health	Float	Salut del personatge.
temperature	Float	Temperatura del personatge.
positionX	Float	Posició en l'eix X del personatge.
positionX	Float	Posició en l'eix Y del personatge.

Taula 52: Character

Action

Representa una acció realitzada pel jugador amb el personatge.

Atribut	Tipus	Descripció
actionType	String	Tipus d'acció.

Taula 53: Action

MainCamera

Representa la càmera principal del sistema.

Atribut	Tipus	Descripció
ID	String	Identificador únic de la càmera principal en el sistema.
resolution	String	Resolució de la càmera principal.

Taula 54: MainCamera

Enemy

Representa un enemic del videojoc.

Atribut	Tipus	Descripció
ID	String	Identificador únic de l'enemic en el sistema.
name	String	Nom de l'enemic.
spawnTime	Float	Temps per l'aparició automàtica de l'enemic.

Taula 55: Enemy

ActiveEnemy

Representa un enemic actiu en una partida determinada.

Atribut	Tipus	Descripció
health	String	Salut de l'enemic.
damage	Float	Dany de l'enemic.

Taula 56: ActiveEnemy

Slime

Representa un enemic del tipus Slime o Llim del videojoc.

Atribut	Tipus	Descripció
slimeAI	Script	Script per la intel·ligència artificial del Slime.

Taula 57: Slime

FireWorm

Representa un enemic del tipus FireWorm o Cuc de foc del videojoc.

Atribut	Tipus	Descripció
fireWormAI	Script	Script per la intel·ligència artificial del FireWorm.

Taula 58: FireWorm

Terrain

Representa el món o terreny d'una partida.

Atribut	Tipus	Descripció
seed	Long	Identificador únic del terreny en el sistema.
worldWidth	Integer	Amplada màxima del món.
worldHeight	Integer	Altura màxima del món.

Taula 59: Terrain

Biome

Representa un bioma del terreny de la partida.

Atribut	Tipus	Descripció
name	String	Nom del bioma.
height	Integer	Altura màxima del bioma.
temperature	Float	Temperatura mitjana del bioma.
biomeType	BiomeType	Tipus de bioma.

Taula 60: Biome

Background

Representa un fons pel videojoc.

Atribut	Tipus	Descripció
name	String	Nom del fons.
image	String	Imatge del fons.

Taula 61: Background

Inventory

Representa l'inventari del jugador.

Atribut	Tipus	Descripció
ID	String	Identificador únic de l'inventari en el sistema.

Taula 62: Inventory

Slot

Representa una ranura de l'inventari.

Atribut	Tipus	Descripció
number	Integer	Número de la ranura.
positionX	Float	Posició X de la ranura en l'inventari.
positionY	Float	Posició Y de la ranura en l'inventari.

Taula 63: Slot

CraftingMenu

Representa el menú de fabricació.

Atribut	Tipus	Descripció
ID	String	Identificador únic del menú de fabricació en el sistema.

Taula 64: CraftingMenu

CraftingRecipe

Representa una recepta de fabricació.

Atribut	Tipus	Descripció
ID	String	Identificador únic de la recepta en el sistema.

Taula 65: CraftingRecipe

Item

Representa un element del videojoc.

Atribut	Tipus	Descripció
ID	String	Identificador únic de l'element en el sistema.
name	String	Nom de l'objecte.
quantity	Integer	Quantitat total de l'objecte.

Taula 66: Item

ItemType

Representa el tipus d'un element.

Block

Representa un material, un tipus d'element del videojoc que es pot col·locar i destruir.

Atribut	Tipus	Descripció
blockType	BlockType	Tipus de material.
miningDuration	Float	Duració total que es triga a destruir.

Taula 67: Block

Tool

Representa una eina, un tipus d'element del videojoc que es pot fer servir per danyar o destruir.

Atribut	Tipus	Descripció
toolType	ToolType	Tipus d'eina.
damage	Float	Dany de l'eina.
strength	Float	Força per destruir de l'eina, la qual es calcula amb l'atribut *miningDuration* del material que es vol destruir.

Taula 68: Tool

Object

Representa un objecte, un tipus d'element del videojoc que es pot utilitzar.

Atribut	Tipus	Descripció
consumable	Boolean	Indica si l'objecte es pot consumir.

Taula 69: Object

5.4 Esquema del comportament

5.4.1 Especificacions amb diagrames de seqüència

CU01 – Crear nova partida

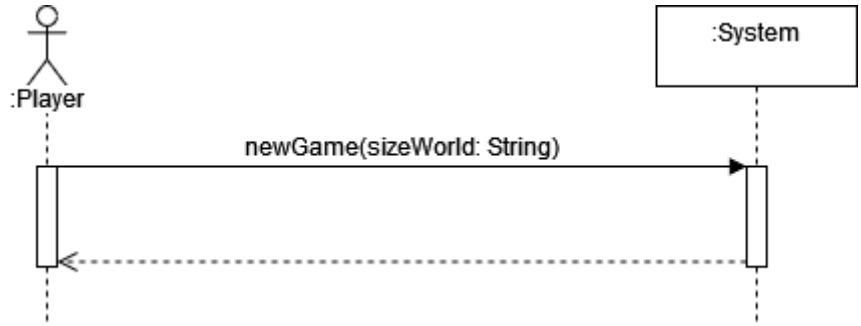


Figura 8: Diagrama de seqüència CU01
Font: *Creació pròpia*

Context	System::newGame(sizeWorld: String)
Pre	El jugador està en el menú principal
	sizeWorld té un valor vàlid en el sistema (*Small*, *Medium* o *Large*)
Post	El sistema crea la partida on el jugador pot començar a jugar

CU02 – Continuar partida existent

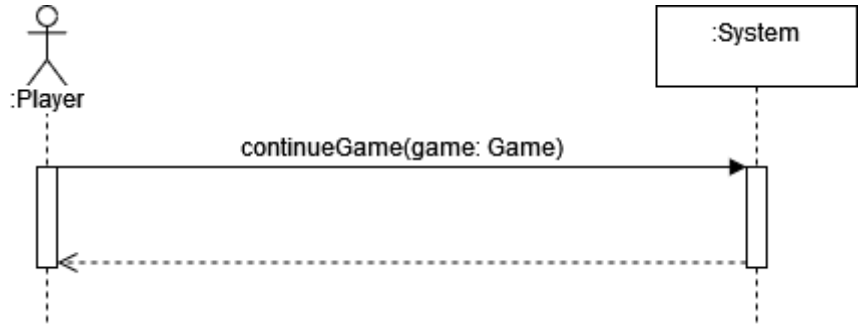


Figura 9: Diagrama de seqüència CU02
Font: *Creació pròpia*

Context	System::continueGame(game: Game)
Pre	El jugador es troba en el menú principal
	game és una partida existent en el sistema
Post	El jugador es troba dins de la partida i pot continuar jugant

CU03 – Jugar

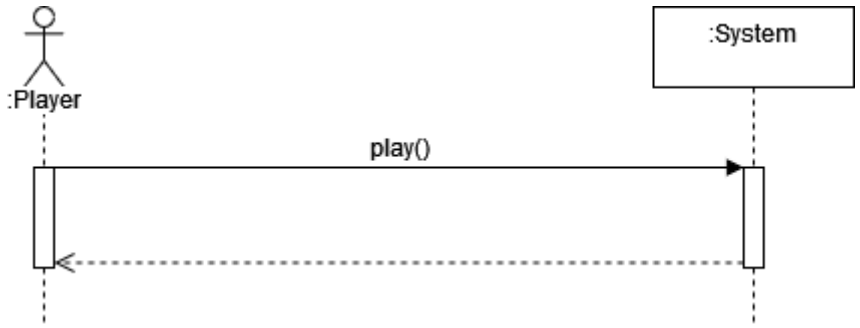


Figura 10: Diagrama de seqüència CU03
***Font:** Creació pròpia*

Context	System::play()
Pre	El jugador ha creat una nova partida o continuat una partida existent
Post	El jugador pot realitzar accions com moure el personatge, saltar, utilitzar les eines, construir i destruir el terreny, etc.

CU04 – Pausar partida

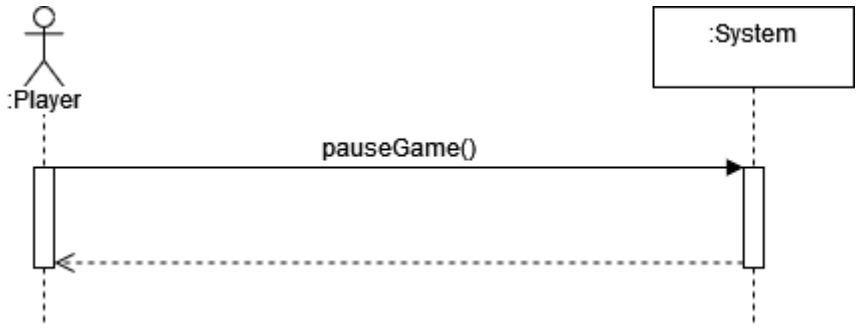


Figura 11: Diagrama de seqüència CU04
***Font:** Creació pròpia*

Context	System::pauseGame()
Pre	El jugador està dins d'una partida La partida no està pausada
Post	El jugador té la partida pausada

CU05 – Reprendre partida

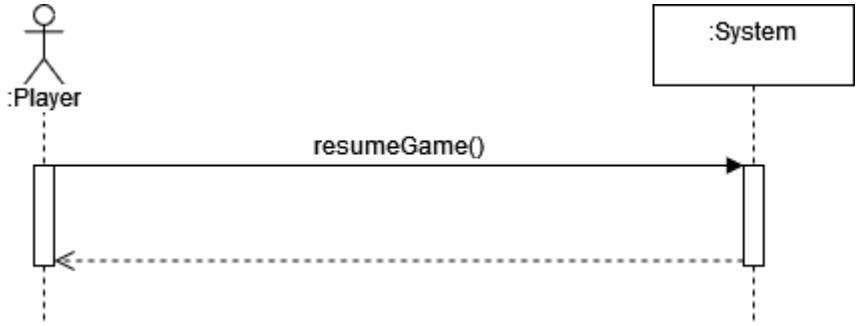


Figura 12: Diagrama de seqüència CU05
***Font:** Creació pròpia*

Context	System::resumeGame()
Pre	El jugador està dins d'una partida
	La partida està pausada
Post	El jugador té la partida sense pausar

CU06 – Reaparèixer

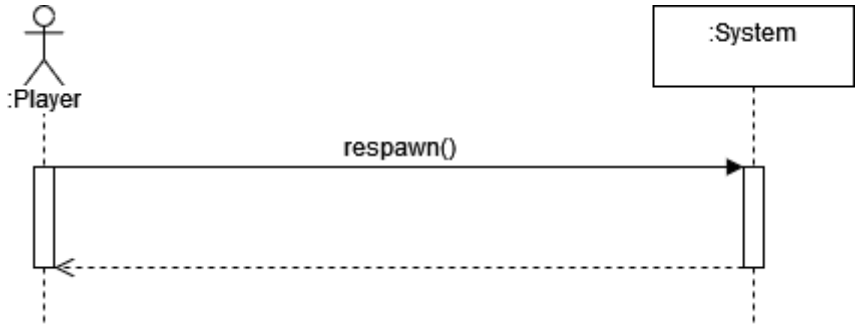


Figura 13: Diagrama de seqüència CU06
***Font:** Creació pròpia*

Context	System::respawn()
Pre	El jugador es troba dins d'una partida
	El personatge que controla el jugador no té punts de salut
Post	El personatge que controla el jugador es troba en el punt de reaparició establert amb la meitat dels punts màxims de salut
	El jugador ha perdut objectes de l'inventari

CU07 – Modificar la configuració

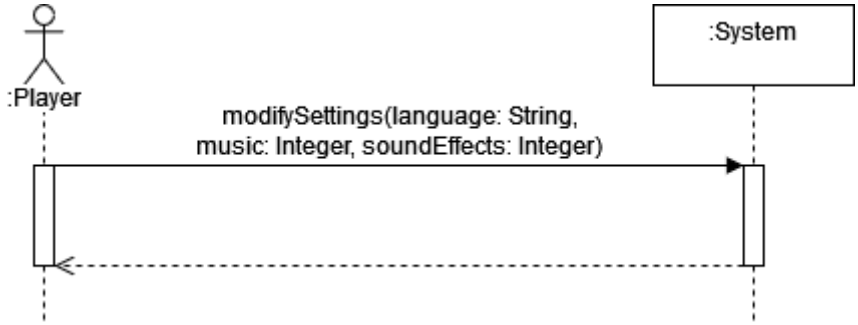


Figura 14: Diagrama de seqüència CU07
***Font:** Creació pròpia*

Context	System::modifySettings(language: String, music: Integer, soundEffects: Integer)
Pre	El jugador està en el menú principal o en el menú dins d'una partida *language* és un idioma vàlid en el sistema (*English* o *Spanish*) *music* i *soundEffects* són enters entre 0 i 10 incluits
Post	El jugador veu els canvis aplicats en el sistema

CU08 – Sortir al menú principal

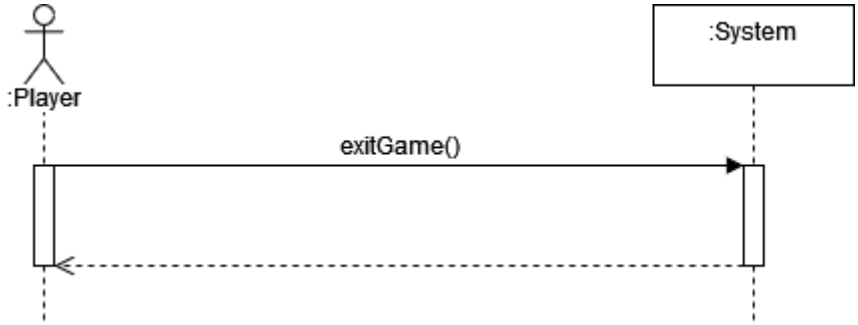


Figura 15: Diagrama de seqüència CU08
***Font:** Creació pròpia*

Context	System::exitGame()
Pre	El jugador es troba en el menú dins d'una partida
Post	El jugador es troba en el menú principal

CU09 – Sortir de l'aplicació

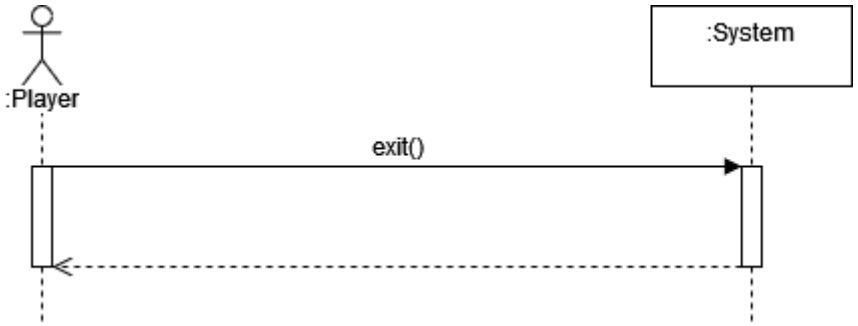


Figura 16: Diagrama de seqüència CU09
Font: *Creació pròpia*

Context	System::exit()
Pre	El jugador es troba en el menú principal
Post	L'aplicació s'ha tancat

5.5 Document de disseny del videojoc

El document de disseny del videojoc, conegut com a GDD (Game Design Document), es tracta d'una síntesi de tot el que comprendrà el videojoc, com el propi nom, el concepte, el gènere, l'art, la jugabilitat i els objectius, entre d'altres.

El títol del videojoc és "Lost Hero" i abasta els gèneres de supervivència en món obert, exploració, sandbox, aventures, acció, construcció, fabricació, 2D i píxel art. El concepte principal de Lost Hero és la llibertat de poder explorar, destruir i construir en el món sense restriccions. Per definir millor els conceptes, en el GDD es parlarà amb els controls de teclat i ratolí.

5.5.1 Jugador

En el videojoc Lost Hero el jugador controla al següent personatge, el qual representa el títol del videojoc, un heroi perdut, amb l'objectiu d'explorar el nou món on es troba.

Figura 17: Sprite del personatge
Font: *rvros.itch.io*

El jugador pot moure el personatge cap a l'esquerra amb la tecla 'a', i cap a la dreta amb la tecla 'd', i augmentar la seva velocitat en desplaçar-se mantenint pressionada la tecla 'shift'. Per saltar ho pot fer amb la barra espaiadora, mantenint-la pressionada fins al final per un salt amb l'altura màxima o deixant-la anar en qualsevol moment per deixar d'incrementar l'altura del salt.

Amb el clic esquerre el jugador pot realitzar l'acció de l'eina que tingui seleccionada. Aquesta acció pot ser destruir un bloc del terreny si l'eina és un pic o una destral, o realitzar un atac cos a cos si l'eina és una espasa. Qualsevol altre objecte seleccionat no té cap efecte realitzar un clic esquerre, tret d'alguns blocs que es poden destruir sense tenir cap eina seleccionada. Amb el clic dret el jugador pot col·locar blocs en el terreny, ja sigui per construir una casa, per arribar a llocs alts o pel que el jugador trobi convenient. El clic dret no té cap efecte si el bloc que es vol col·locar no està seleccionat o el cursor està situat fora del rang de col·locació del personatge. A la figura següent es mostren les diferents animacions del personatge depenent de l'acció que estigui realitzant.

Figura 18: Sprites de les animacions del personatge
Font: *rvros.itch.io*

En la primera fila de la figura es mostra l'animació de quan el personatge està quiet, en la segona fila quan el personatge està caminant, en la tercera fila quan està corrent i en l'última fila es mostra quan està saltant en els quatre primers sprites, i quan està caient en els dos últims sprites.

El personatge, com s'ha comentat en apartats anteriors, compte amb una barra de salut, com es pot veure en la figura següent, juntament amb un indicador de la temperatura actual.

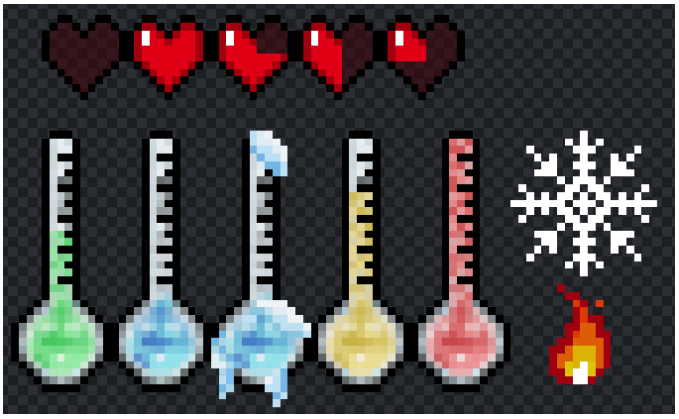


Figura 19: Sprites de la salut i la temperatura
Font: *Creació pròpia amb pixlr.com*

A continuació es mostra un exemple del personatge en el desert i en la neu, per poder observar la barra de salut i l'indicador de temperatura, la qual varia depenent del bioma. S'ha de tenir en compte que s'ha aplicat zoom a la imatge per poder apreciar millor la interfície i no és la resolució final del videojoc.

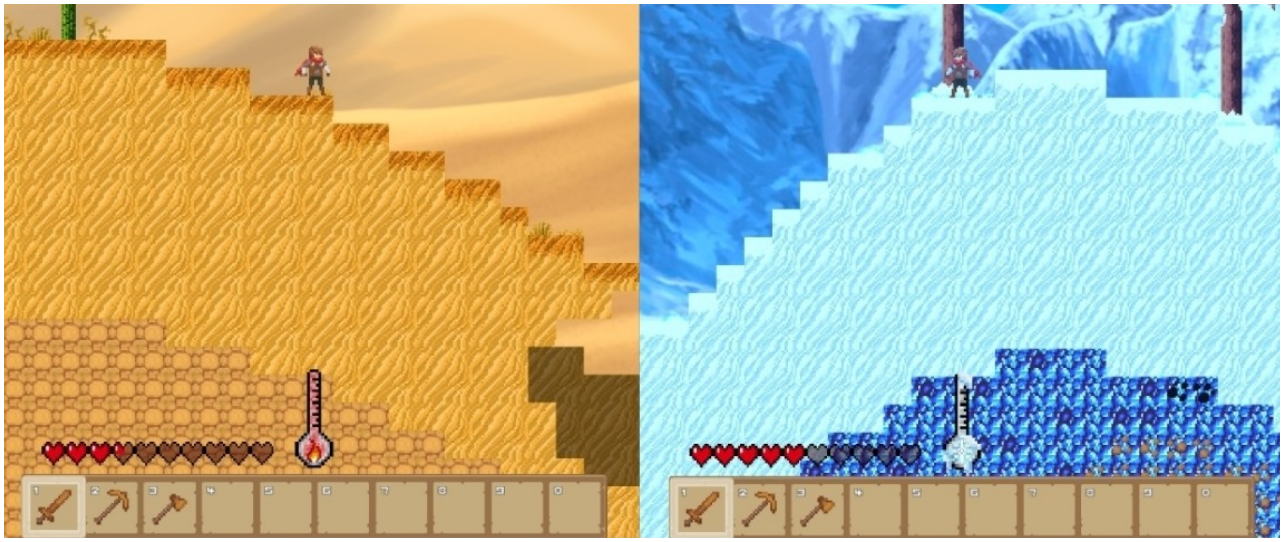


Figura 20: Captura de pantalla dins del videojoc en diferents biomes
Font: *Creació pròpia amb Unity*

5.5.2 Blocs

El món de Lost Hero està format per un conjunt de diferents blocs, cadascun amb la seva funció, ja sigui per fabricar altres materials, per construir o per decorar. A continuació es mostren els diferents tipus de blocs.

Terreny

A continuació podem veure en la primera fila els sprites que representen la terra, la terra amb herba de la superfície i la pedra, que pertanyen al bioma inicial, o de les planures. En la segona fila estan els sprites que representen la sorra, els blocs de sorra de la superfície i l'arenisca del bioma del desert, i per últim, en la tercera fila es troben els sprites de la neu, la neu a la superfície i el gel del bioma de neu.

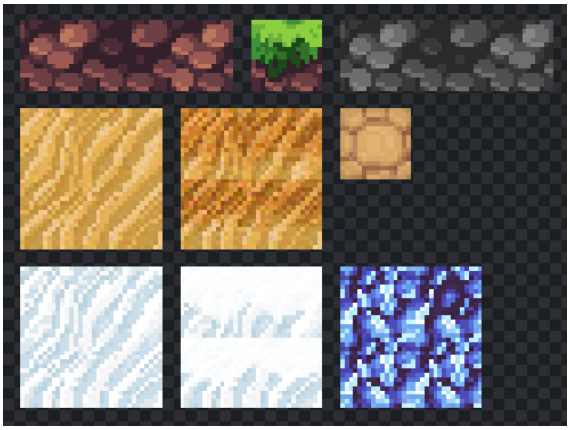


Figura 21: Sprites dels blocs del terreny
Font: *Creació pròpia amb pixlr.com*

Minerals

Els blocs que no es troben en la superfície, és a dir, la pedra i les variants dels diferents biomes, com l'arenisca i el gel, poden contenir diversos minerals depenent de la profunditat en la qual es trobin. Després de picar alguns d'aquests minerals, aquests deixen anar un tros del mineral en qüestió sense refinar, el qual es pot processar a lingot amb carbó, mitjançant el menú de fabricació. Els minerals que es poden trobar en el món de Lost Hero són el carbó, el coure, el ferro, l'or, el platí i el diamant. A la figura següent es mostren aquests minerals amb algunes variants d'exemple, i els ítems corresponents que obtens quan els mines.

Figura 22: Sprites dels minerals amb els drops corresponents
Font: *Creació pròpia amb pixlr.com*

Vegetació

Cada bioma té la seva pròpia vegetació. En el bioma de les planures, el jugador es pot trobar amb arbres de diferents altures, vegetació per la superfície, com arbustos i herba, fruites en alguns arbres, les quals es poden consumir per regenerar salut del personatge, i diferents bolets. Els arbres es poden talar amb la destral seleccionant un dels blocs del tronc i destruint-lo, fent caure així tot l'arbre i deixant anar una quantitat de fusta que varia depenent de l'altura d'aquest, i una fruita si en tenia.

En el bioma de neu, el jugador es pot trobar amb els mateixos arbres però nevats, o amb arbres morts i congelats. També hi ha herba congelada. I en el desert, el jugador es pot trobar amb cactus, els quals es destrueixen a partir del bloc del cactus que el jugador realitza l'acció de picar cap amunt, i vegetació morta. En la figura següent es poden veure els diferents blocs descrits en aquest apartat.

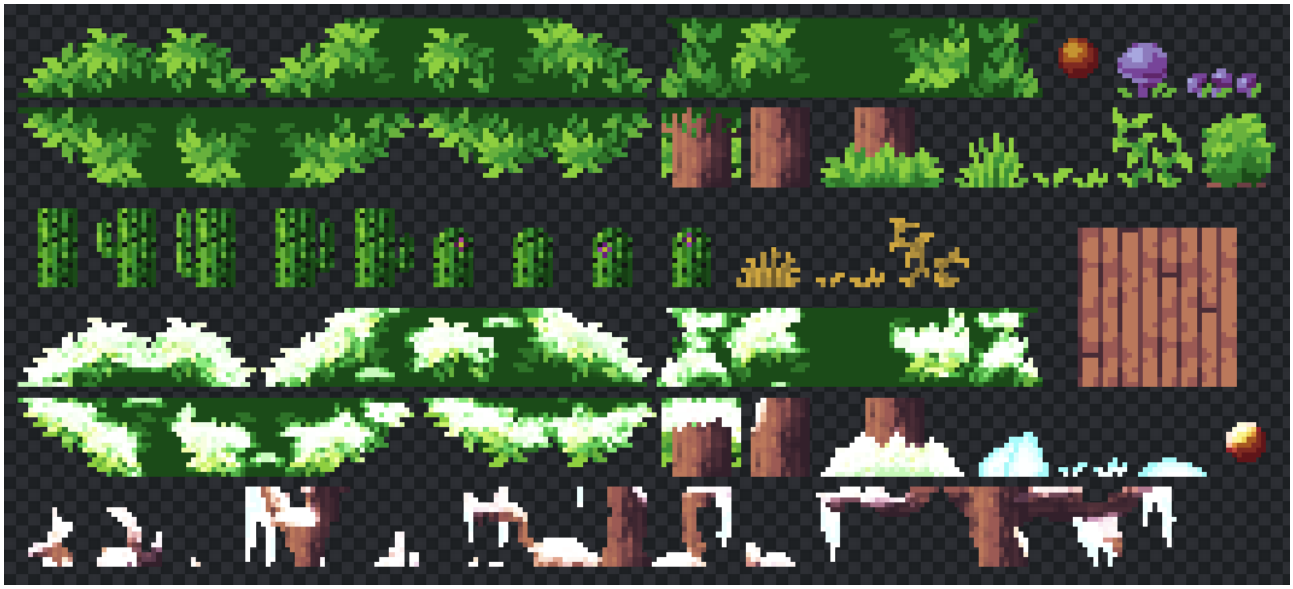


Figura 23: Sprites de la vegetació de diferents biomes i de la fusta dels arbres
Font: *Creació pròpia amb pixlr.com*

5.5.3 Eines

A continuació es poden veure les diferents eines fabricades amb els minerals comentats anteriorment. Les eines de coure, ferro i or es fabriquen amb fusta i el mineral corresponent, i les eines de platí i diamant amb ferro i el mineral corresponent.

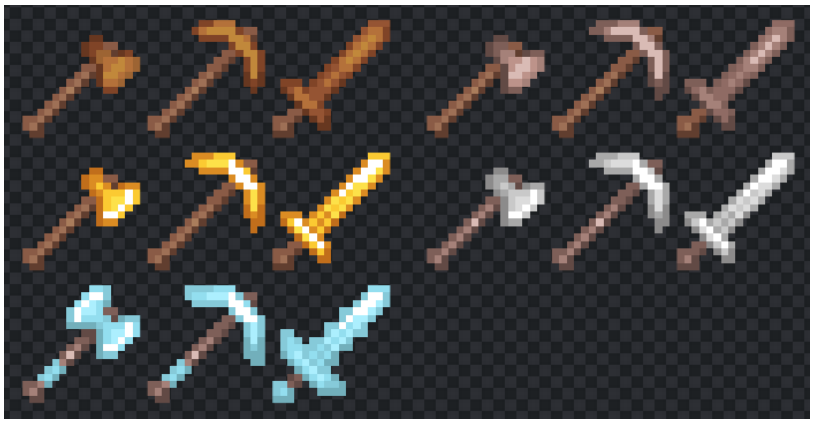


Figura 24: Sprites de les eines
Font: *Creació pròpia amb pixlr.com*

Cada eina té una potència diferent. En el cas de les destrals i pics, aquesta potència determina el temps que es triga a picar un bloc en concret, tenint en compte que cada bloc té una duració diferent i que alguns blocs necessiten unes eines específiques per picar-se, com és el cas del diamant en mineral, que necessites un pic de platí o de diamant. En el cas de les espases, la potència defineix el dany que causen als enemics. Quan el jugador utilitza una eina, es realitza una animació del sprite de l'eina amb la mà del personatge que va de 90 graus a 0 graus, per així donar a entendre a l'usuari que està utilitzant l'eina en concret. Aquesta animació també es realitza en col·locar blocs amb el sprite determinat del bloc.

5.5.4 Terreny

A continuació es mostra un exemple de generació del terreny de Lost Hero des de l'editor de Unity. Aquest terreny es genera aleatòriament en cada partida, tenint una *seed* única que l'identifica, és a dir, un número que s'utilitza per crear procedimentalment el món del videojoc. Encara que el terreny estigui generat de manera aleatòria, hi ha unes pautes establertes, com l'altura del terreny o la freqüència dels minerals, entre d'altres.

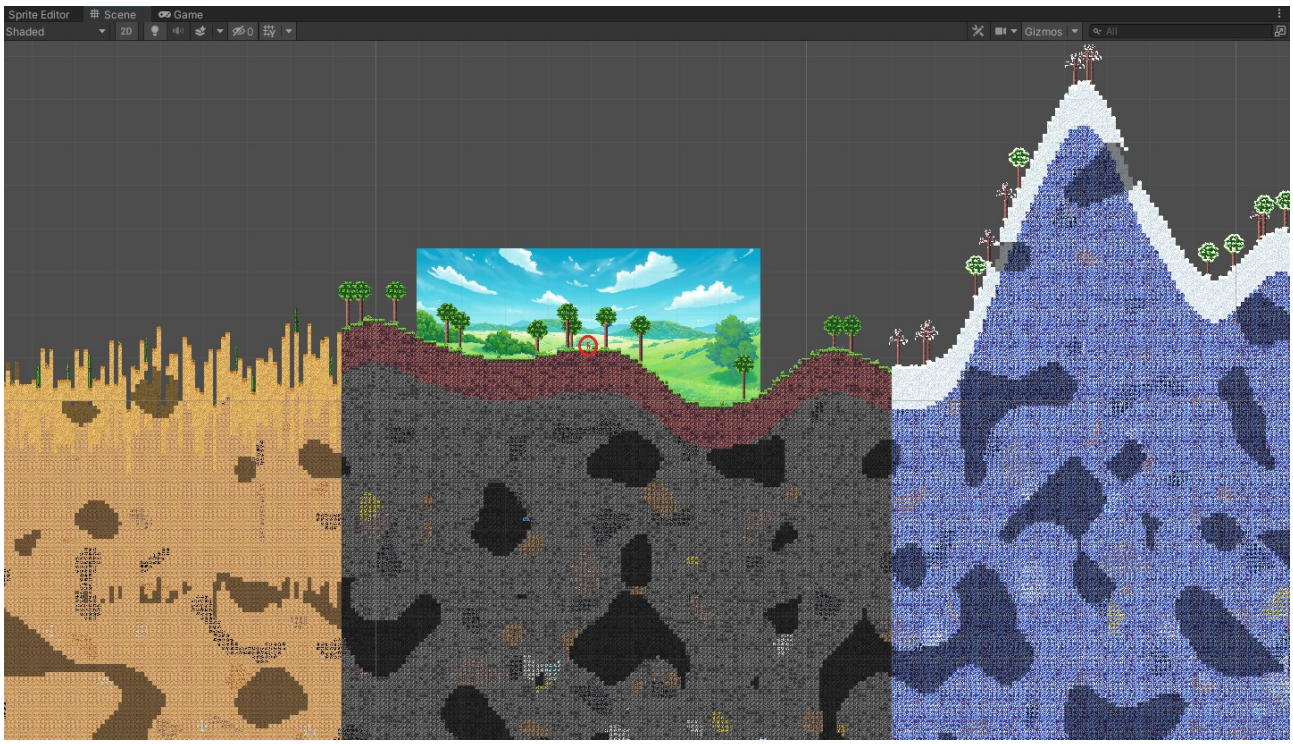


Figura 25: Exemple de terreny
Font: *Creació pròpia amb Unity*

El terreny que es mostra és d'una partida amb la selecció de món Mitjà, i no representa el món complet, només una part d'aquest per tenir una idea general de la generació i per poder diferenciar els diferents elements. El personatge que el jugador controla està dins del cercle vermell que es pot veure en la imatge, i el fons que es veu és el que englobaria tota la pantalla del dispositiu on estigui jugant l'usuari.

5.5.5 Enemics

Cada bioma té enemics propis del bioma que van apareixent fora del camp de visió del jugador, fins a un màxim de 30 enemics actius per un bon rendiment. Cada enemic té una intel·ligència artificial pròpia del tipus d'enemic que pertanyi, per exemple, l'enemic Slime representa un llim el qual es mou sempre en direcció al jugador i té la característica única que salta aproximadament el doble de l'altura del salt del jugador, realitzant aquesta acció tenint en compte diverses variables, com és la posició del jugador o si hi ha un mur davant del llim. A la figura següent es pot veure un exemple d'enemic Slime amb el seu conjunt de sprites per representar les seves animacions, com estar quiet, caminar, saltar, fer dany, rebre dany i morir.

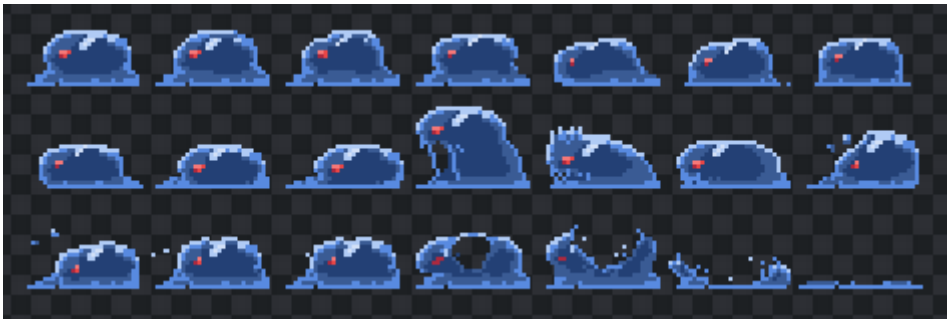


Figura 26: Sprites d'un Slime
Font: *rvros.itch.io*

5.5.6 Interfícies

L'inventari és la interfície més important que el jugador té dins del joc, la qual mostra tots els ítems dels quals disposa en la partida actual, i que pot organitzar per tenir-los ordenats en la barra d'ús principal. Aquesta interfície també té un botó per mostrar la interfície de fabricació, on el jugador pot fabricar altres ítems a partir dels que posseeix. A continuació es mostren aquestes dues interfícies just després de clicar el botó de fabricació, i la interfície del menú dins del joc, on l'usuari pot resumir el joc pausat, entrar a la configuració per ajustar el volum de la música i els efectes de so, i sortir al menú principal, és a dir, sortir de la partida actual. Quan una interfície és oberta, ja sigui l'inventari o el menú, el joc és pausat.

Figura 27: Interfícies d'inventari i fabricació
Font: *Creació pròpia amb Unity*

Figura 28: Interfície del menú dins d'una partida
Font: *Creació pròpia amb Unity*

La figura següent mostra l'interfície del menú principal del joc, on l'usuari pot crear una partida, continuar una partida existent, modificar la configuració del joc i sortir de l'aplicació.

Figura 29: Interfície del menú principal
Font: *Creació pròpia amb Unity*

6 Arquitectura del software

En aquest apartat es defineix l'arquitectura del software del videojoc, juntament amb els diagrames de seqüència més importants, i s'exposa el document de disseny del videojoc, mostrant i explicant els diferents elements gràfics i mecàniques del videojoc.

6.1 Arquitectura del videojoc

Després d'aprofundir sobre el motor Unity amb la documentació disponible i d'investigar sobre el funcionament intern de Unity, he decidit adaptar l'arquitectura del software que s'utilitzarà pel desenvolupament del videojoc per una adient al *framework* que s'utilitzarà. Unity té un disseny d'arquitectura basat en components, i la forma natural en què està dissenyat per treballar és tractar cada objecte del joc com un sistema independent.

Inicialment, el patró que s'anava a seguir pel desenvolupament del videojoc era el patró MVC (Model-Vista-Controlador), separant la interfície d'usuari de la lògica i les dades del videojoc. Resumint, en la capa Model es trobarien els elements d'emmagatzematge de les dades necessàries perquè el sistema executi l'aplicació i pugui guardar el progrés en una base de dades local. En la capa Controlador es trobaria tota la lògica del videojoc, és a dir, contindria tots els *scripts* amb les funcionalitats i els diferents mòduls de gestió, com podria ser la gestió d'animacions, la gestió de la IA, la gestió de les físiques, etc. I finalment en la capa Vista, que seria la presentació, es trobaria la interfície d'usuari encarregada de mostrar els diferents estats i successos del videojoc per així guiar correctament l'usuari amb la interacció d'aquests.

Però coneixent l'arquitectura de Unity i com està dissenyat el framework per utilitzar-lo, aquest projecte seguirà el model d'arquitectura basada en components, seguint el patró arquitectònic d'**Entity Component System** (ECS), o Sistema Entitat-Component, un patró que s'utilitza majoritàriament en el desenvolupament de videojocs i que Unity ofereix un paquet per aplicar-lo al projecte. Com es pot veure en la següent figura, l'arquitectura ECS separa la identitat (entitat), les dades (components) i el comportament (sistema), i l'arquitectura es centra principalment en un disseny orientat a les dades. El sistema llegeix els fluxos de dades dels components i els transforma d'un estat d'entrada a un estat de sortida, que després les entitats indexen.

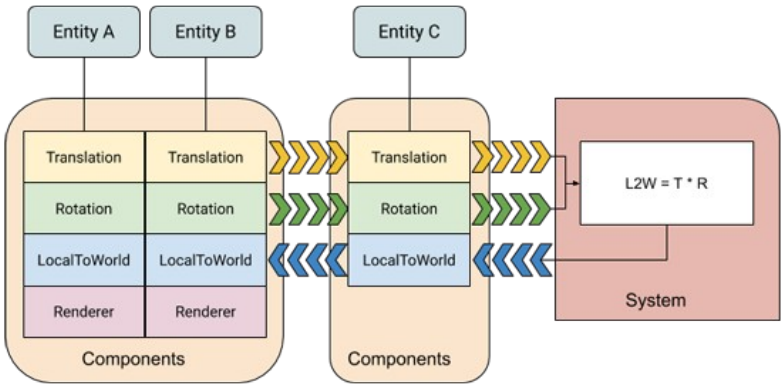


Figura 30: Diagrama d'exemple d'ECS
Font: *[24]*

En el diagrama anterior tenim un sistema que llegeix els components de translació i rotació oferits per Unity, els multiplica i després actualitza els components LocalToWorld corresponents (L2W = T*R).

Com s'ha comentat en la pàgina anterior, el patró ECS té tres parts principals:

- Entitats: els objectes que omplen el videojoc.
- Components: les dades associades a les entitats, però organitzades per les mateixes dades en lloc de per les entitats.
- Sistemes: la lògica que transforma les dades dels components del seu estat actual al seu estat següent.

La principal raó per la qual s'utilitza aquest patró en el desenvolupament del videojoc és perquè permet processar els diferents estats d'una manera flexible i extensible. En un moment donat del videojoc s'han de processar molts estats diferents, com per exemple els diferents enemics amb les seves accions, l'estat del personatge actual, l'estat de tots els milers de blocs del terreny i dels ítems, entre d'altres. Cada objecte en l'ambient té un estat i un comportament que s'ha de gestionar, i ECS ofereix solucions que faciliten aquesta gestió dels estats amb eficiència. A continuació s'expliquen en detall les parts principals d'aquest patró arquitectònic per entendre millor l'aplicació d'aquestes.

6.1.1 Entitats

Les entitats representen totes les "coses" individuals del joc. Una entitat no té ni dades ni un comportament, però sí que identifica quines dades li pertanyen. Essencialment, una entitat és un identificador, és a dir, un GameObject que ni tan sols té un nom per defecte, i una característica important és que l'identificador de les entitats són estables, i per tant, es poden utilitzar per emmagatzemar una referència a un altre component o entitat. Per exemple, una entitat secundària o *child* d'una jerarquia pot necessitar referenciar la seva entitat principal o *parent*.

La manera més senzilla de crear una entitat és amb l'editor de Unity, i es pot configurar que tots els GameObjects i *Prefabs* d'un projecte existent es converteixin en entitats. El videojoc té un *EntityManager*, o gestor d'entitats, que s'encarrega de gestionar totes les entitats existents, tenint una llista completa i organitzant les dades associades a les entitats per un rendiment òptim.

Encara que una entitat no tingui un tipus, els grups d'entitats es categoritzen mitjançant els tipus de dades dels components associats amb ells. El gestor d'entitats fa un seguiment de totes les combinacions úniques de components de les entitats existents, i les agrupa pel que s'anomena *Archetype*, o arquetip. Es poden crear noves entitats que s'ajustin a arquetips existents, o utilitzar aquests arquetips per crear entitats. A la figura següent es mostra un exemple.

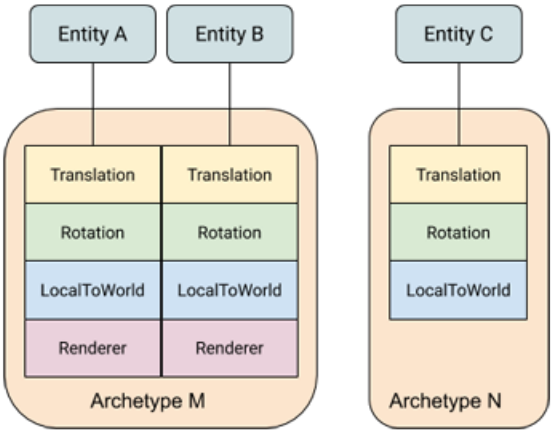


Figura 31: Diagrama d'exemple d'arquetip
Font: *[24]*

Unity ofereix la possibilitat de serialitzar els diferents objectes d'un projecte. La serialització és el procés automàtic de transformació d'estructures de dades o estats d'objectes en un format que es pugui emmagatzemar i reconstruir a posteriori, i Unity utilitza aquesta serialització per carregar i desar diferents objectes cap i des del disc dur de l'ordinador, com podrien ser components MonoBehaviour o objectes amb scripts. Amb el patró ECS, la millor opció amb un impacte mínim en el rendiment del videojoc és serialitzar les diferents entitats o l'EntityManager que conté aquestes entitats.

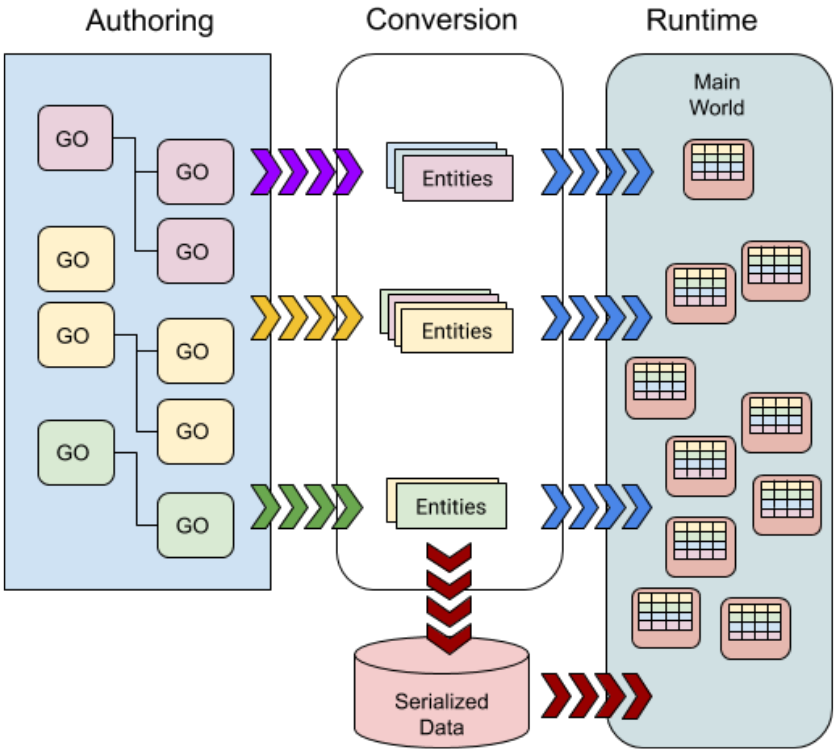


Figura 32: Diagrama de dades serialitzades d'ECS
Font: *[24]*

6.1.2 Sistemes

Els sistemes proporcionen la lògica que modifica les dades dels components que tenen les entitats d'un estat actual a un estat pròxim. Un exemple seria l'actualització de les posicions de totes les entitats en moviment multiplicant la seva velocitat per l'interval de temps de l'actualització anterior. A continuació es pot veure el funcionament bàsic d'un sistema.

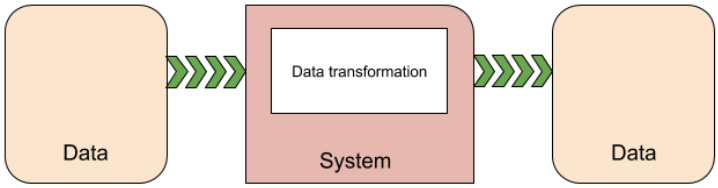


Figura 33: Procés de transformació de dades d'un sistema
Font: *[27]*

L'ECS que proporciona Unity detecta automàticament les classes de sistema del projecte, les crea en temps d'execució, i afegeix cada sistema a un dels grups de sistemes predeterminats. Es poden utilitzar atributs d'un sistema per especificar el grup original del sistema i l'ordre del sistema dins del grup.

En aquest projecte s'aplicarà el que s'anomena **Hybrid ECS** o ECS híbrid, per poder utilitzar així la classe MonoBehaviour que Unity implementa. En el següent apartat de components s'explica en profunditat aquesta classe i la importància que té a l'hora d'implementar i programar els components Script en els GameObjects.

Una implementació híbrida d'ECS conté totes les característiques d'una implementació pura d'ECS juntament amb classes d'ajuda que converteixen les classes MonoBehaviour en components d'ECS. És a dir, la classe ComponentSystem, que és la classe que fa servir el paquet d'ECS que proporciona Unity per crear els diferents sistemes, s'utilitza per actualitzar les dades dels diferents objectes amb la classe MonoBehaviour inclosa.

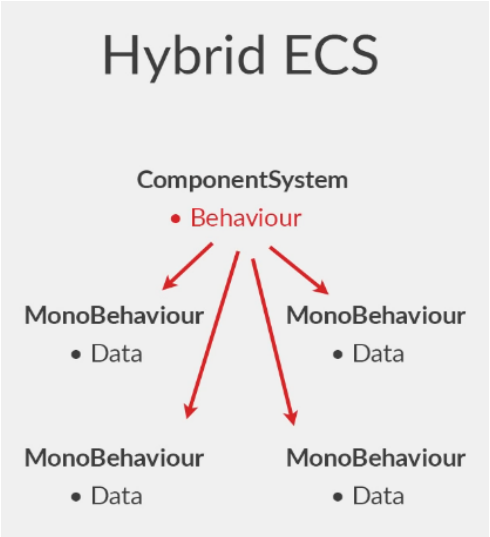


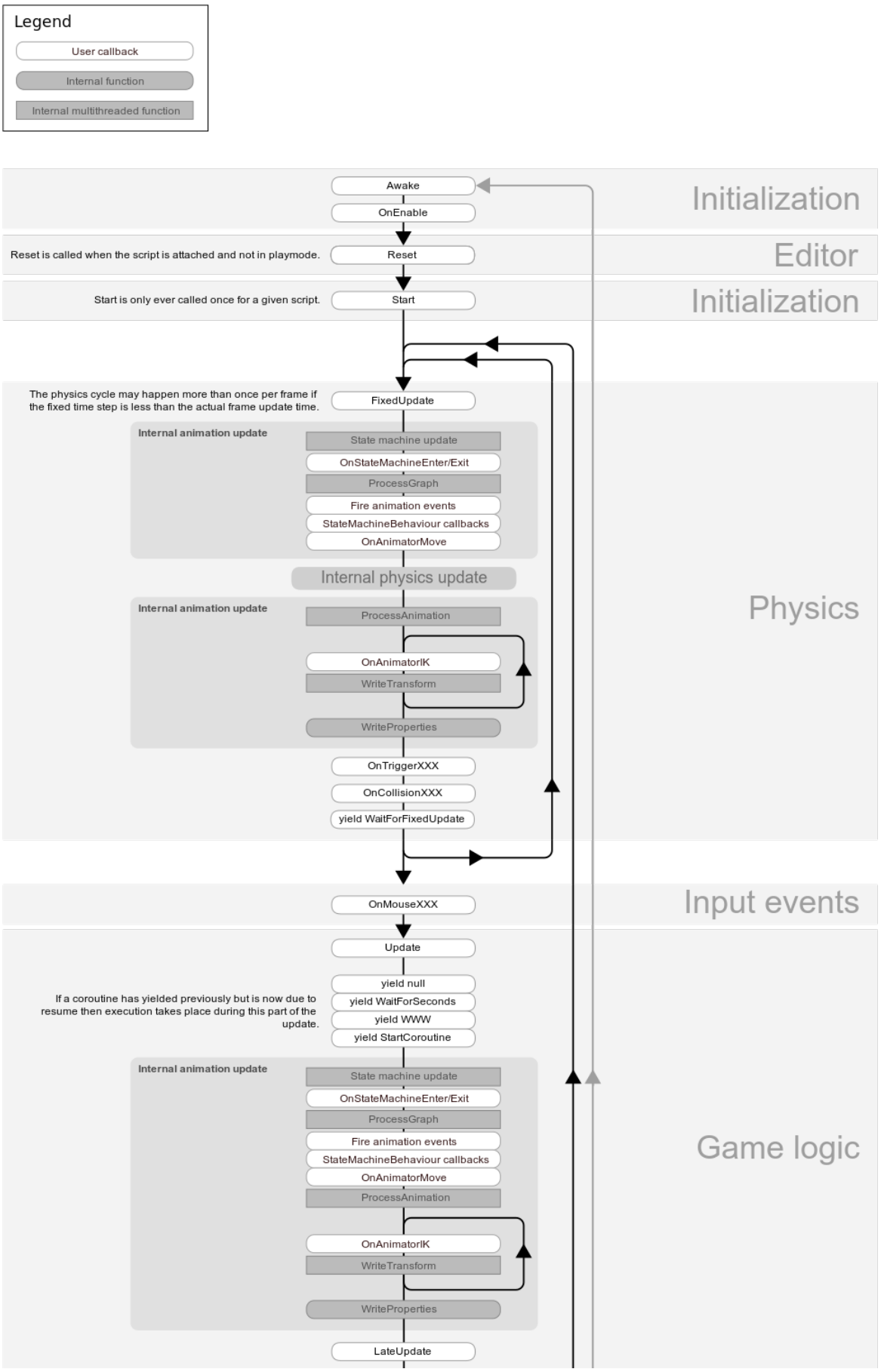
Figura 34: Sistema d'un ECS híbrid
Font: *Creació pròpia*

6.1.3 Components

Un component és un objecte software destinat a interactuar amb altres components, i aquest es pot desplegar de manera independent i està subjecte a composicions de tercers. Aquest tipus d'arquitectura facilita el testeig, i per tant, facilita el manteniment i l'ampliació de les diferents entitats programades amb els seus components. També és molt flexible, evita problemes amb la denominació de mètodes i variables, i redueix la complexitat dels *tracebacks* per arribar a l'objecte original.

En l'enfocament de programació tradicional, l'herència s'aprofita dins de la programació orientada a objectes principalment per generar una jerarquia de classes en una aplicació en concret i així poder reutilitzar codi. En canvi amb Unity és millor utilitzar un enfocament basat en la composició per com funciona el framework, ja que en tot moment s'utilitza la composició per sobre de l'herència, és a dir, les funcionalitats es dissenyen afegint components mitjançant la composició. Tot el que compon una escena de Unity són *GameObjects*, és a dir, objectes del joc sense ser subclasses d'alguna altre entitat, i la majoria de funcionalitats són implementades com a *MonoBehaviours* dins d'aquests GameObjects que actuen com a contenidors de components. Per exemple, el GameObject Personatge estaria format per una ID del GameObject, i diferents components amb els seus propis comportaments, com podrien ser Vida, Posició, Atacar, Velocitat, Animador, etc. A l'hora de dissenyar i implementar funcionalitats, és més eficient i més fàcil de mantenir afegir directament els components que compleixen aquestes funcionalitats en el GameObject respectiu en comptes d'heretar-los d'altres objectes.

Per entrar en context, MonoBehaviour és la classe principal de la qual deriven tots els scripts de Unity, i la figura següent resumeix com Unity ordena i repeteix els diferents esdeveniments al llarg de la vida d'un script. En la figura podem veure funcions com *Awake()*, que es crida quan s'inicialitza l'script, *Start()*, que es crida un sol cop en activar-se l'script i sempre després de la funció Awake, o *Update()*, que es crida una vegada per frame, sent així la funció més important pel funcionament dels components. La funció *FixedUpdate()* vindria a ser una variant d'Update que només s'utilitza per a les físiques del GameObject, i que es deixa de cridar a cada frame si l'escala de temps és 0, i la funció *LateUpdate()* funciona idènticament que la funció Update, amb la diferència que es crida just després d'Update. Un ús comú de LateUpdate seria implementar en aquesta funció el moviment d'una càmera seguint el personatge per assegurar que el personatge s'hagi mogut completament, tenint en compte que el moviment del personatge està en Update, on es trobarien els inputs i altres funcions, i FixedUpdate, on es trobarien els càlculs de les físiques. En la figura també podem veure la lògica del sistema, amb els diferents estats de les animacions i de les corutines, i el funcionament de la renderització dels diferents elements de l'escena.


Legend
User callback
Internal function
Internal multithreaded function
Initialization
Awake
OnEnable
Editor
Reset is called when the script is attached and not in playmode.
Reset
Initialization
Start is only ever called once for a given script.
Start
Physics
The physics cycle may happen more than once per frame if the fixed time step is less than the actual frame update time.
FixedUpdate
Internal animation update
State machine update
OnStateMachineEnter/Exit
ProcessGraph
Fire animation events
StateMachineBehaviour callbacks
OnAnimatorMove
Internal physics update
Internal animation update
ProcessAnimation
OnAnimatorIK
WriteTransform
WriteProperties
OnTriggerXXX
OnCollisionXXX
yield WaitForFixedUpdate
Input events
OnMouseXXX
Game logic
Update
If a coroutine has yielded previously but is now due to resume then execution takes place during this part of the update.
yield null
yield WaitForSeconds
yield WWW
yield StartCoroutine
Internal animation update
State machine update
OnStateMachineEnter/Exit
ProcessGraph
Fire animation events
StateMachineBehaviour callbacks
OnAnimatorMove
ProcessAnimation
OnAnimatorIK
WriteTransform
WriteProperties
LateUpdate

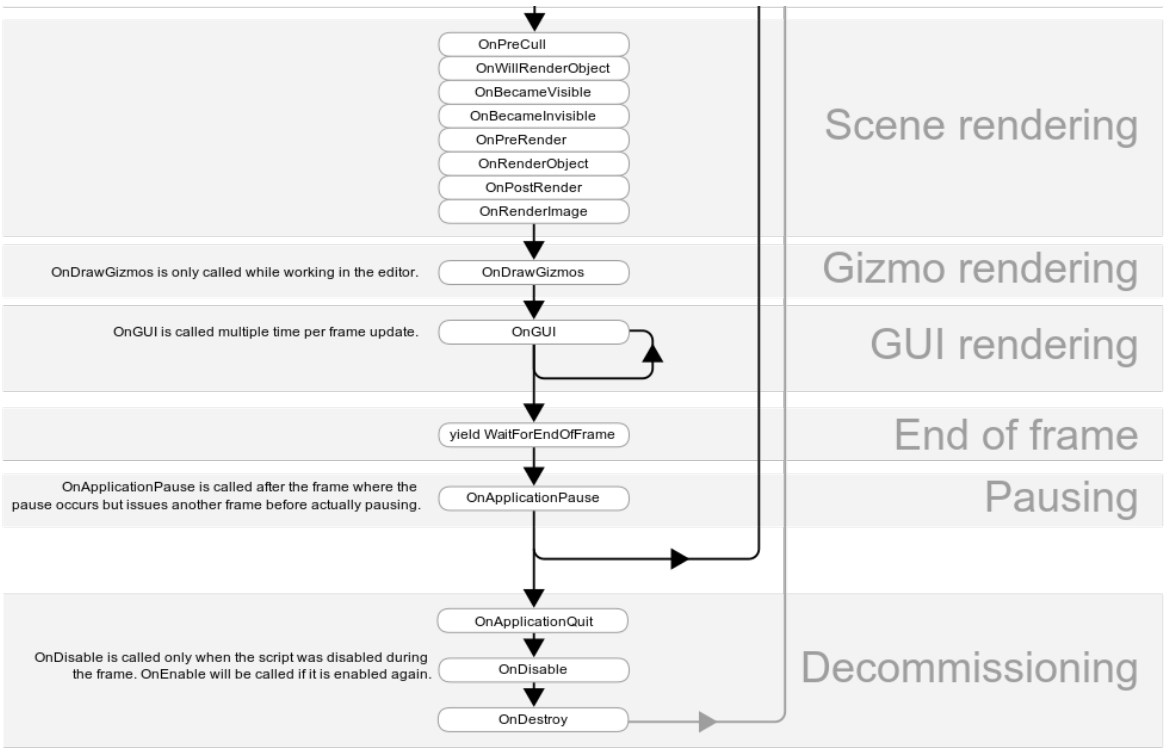


Figura 35: Cicle de vida de MonoBehaviour
Font: *[25]*

A continuació tenim una figura que mostra un exemple d'un GameObject amb els seus components, en aquest cas, el GameObject que genera la llum de l'escena, amb els components *Transform*, on s'indica la posició, la rotació i l'escala d'aquest, i el component *Light*, que en aquest cas és un component existent de Unity amb una sèrie d'atributs predefinits.

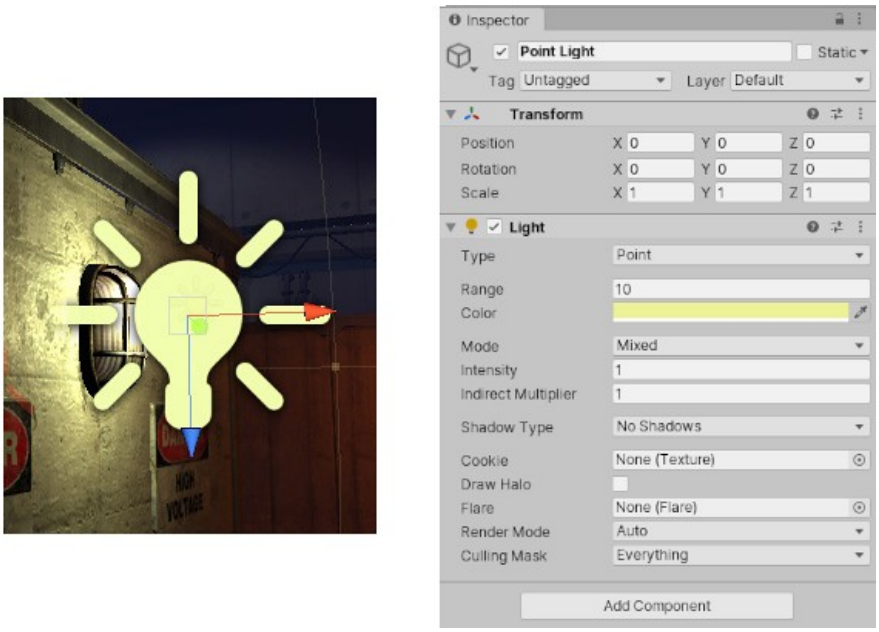


Figura 36: GameObject d'una llum
Font: *[28]*

Una característica dels GameObjects a tenir en compte és que sempre han de tenir el component Transform adjunt, indicant la posició, rotació i escala en l'escena.

Com s'ha comentat en l'apartat d'entitats, el gestor d'entitats organitza combinacions úniques dels components en arquetips. Emmagatzema els components de totes les entitats amb el mateix arquetip juntament amb els blocs de memòria anomenats *chunks*. Totes les entitats d'un chunk determinat tenen el mateix tipus d'arquetip. En la següent figura es mostra com l'ECS emmagatzema les dades dels components mitjançant els arquetips.

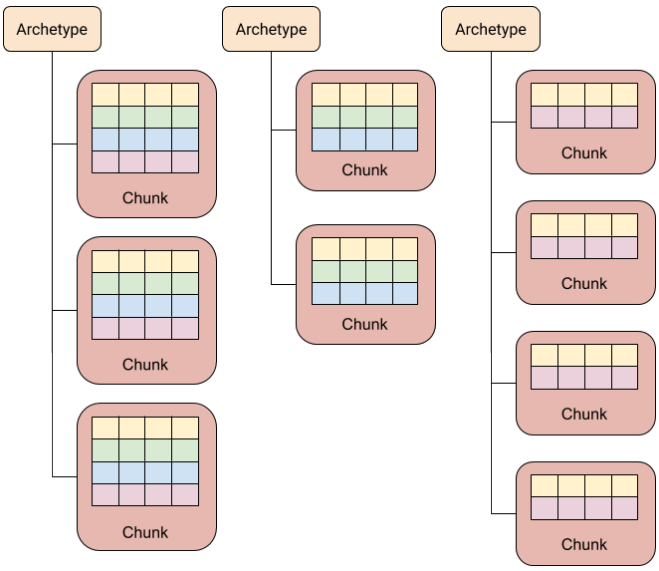


Figura 37: Diagrama del funcionament d'arquetips
Font: *[29]*

Les característiques principals dels components són les següents:

- Reutilitzabilitat: els components solen estar dissenyats per ser reutilitzats en diferents situacions en diferents aplicacions. Tanmateix, alguns components poden estar dissenyats només per a una tasca específica.
- Substituïble: els components es poden substituir lliurement per altres components similars.
- No específic: els components estan dissenyats per funcionar en diferents entorns i contextos.
- Extensible: un component es pot estendre a partir de components existents per proporcionar un comportament nou.
- Encapsulat: un component té la seva pròpia interfície, cosa que permet que s'utilitzin les seves funcionalitats sense exposar els detalls dels processos interns ni cap variable o estat intern.
- Independent: els components estan dissenyats per tenir dependències mínimes d'altres components.

Els components principals i més importants que s'utilitzaran en aquest projecte són els següents:

- Script: permet afegir diferents funcionalitats i comportaments a l'objecte.
- Rigidbody2D: permet a l'objecte interactuar amb les físiques del joc. Sense aquest component no es poden afegir d'altres, com per exemple el component Physics 2D.
- Camera: converteix l'objecte en una càmera, capturant i mostrant el món al jugador.
- Physics2D: permet a l'objecte col·lisionar amb altres i definir els contorns de col·lisió de l'objecte, entre d'altres. Engloba tots els components Collider2D.
- Animator: permet a l'objecte organitzar i mantenir un conjunt de clips d'animació i transicions d'animació associades.
- SpriteRenderer: renderitza un objecte a un *sprite* per gràfics 2D.
- Canvas: permet crear elements GUI dins d'aquest component.
- AudioSource: afegeix a l'objecte una font d'àudio. Es necessita el component AudioListener per poder escoltar aquest àudio, el qual sol estar en l'objecte que té el component Camera.

6.1.4 Diagrama ECS del videojoc

A continuació es mostra l'arquitectura ECS amb els elements del videojoc. Algunes entitats no es poden representar per tot el sistema, com és el cas de les entitats dels enemics, ja que es van instanciant amb el temps a partir d'una entitat prefabricada. En aquest cas, Unity agrupa les diferents entitats d'enemics d'un únic tipus en un arquetip amb els components d'aquestes entitats. El següent diagrama s'ha dissenyat tenint en compte un sistema amb dos entitats d'enemics del tipus Slime actius, i un número *n* de blocs totals del terreny. Les escenes del diagrama, explicades posteriorment en l'apartat d'implementació, funcionen com a contenidors de les diferents entitats.

S'ha de tenir en compte que els components mostrats a continuació els proporciona Unity amb la possibilitat de poder modificar els valors d'aquests i estendre'ls amb el component Script, el qual és el component per escriure codi, en aquest cas en C# amb VisualStudio. Totes les funcionalitats del projecte han sigut programades utilitzant els components que ofereix Unity. Per exemple, la intel·ligència artificial dels enemics o els estats ocasionats per les col·lisions s'han programat des de zero, però el component Camera o el component Animator han servit com a base per programar les funcionalitats de la càmera i de les animacions, fent servir les interfícies d'aquests components i estenent el necessari amb el component Script. A l'apartat d'implementació s'expliquen en detall els components més importants i com s'han implementat en el videojoc.

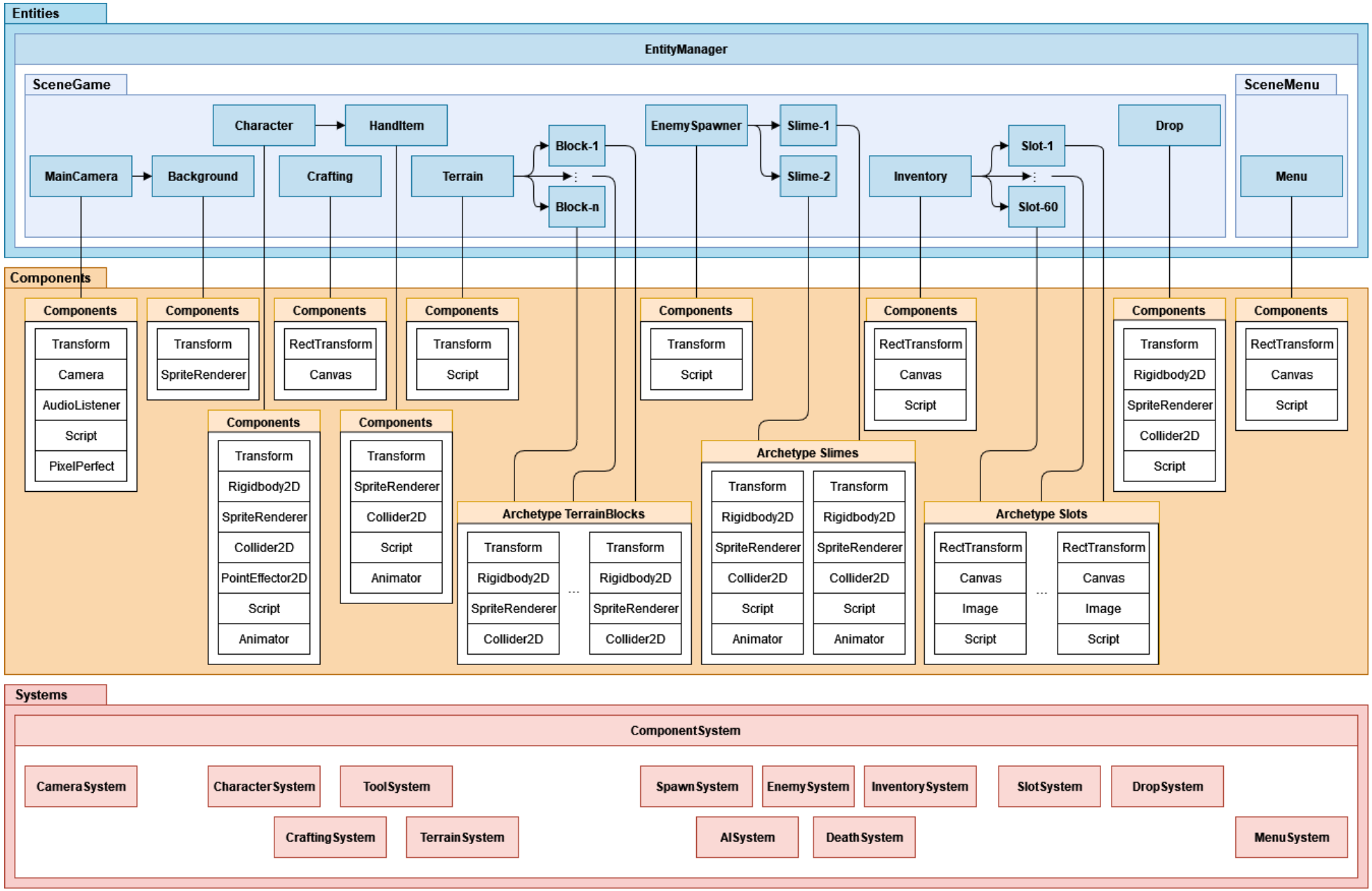


Figura 38: Diagrama ECS del videojoc
Font: *Creació pròpia*

6.2 Diagrames de seqüència

A continuació es mostren els diagrames de seqüència més rellevants del videojoc per tenir una representació més clara del funcionament del sistema utilitzat.

6.2.1 Creació d'una partida nova

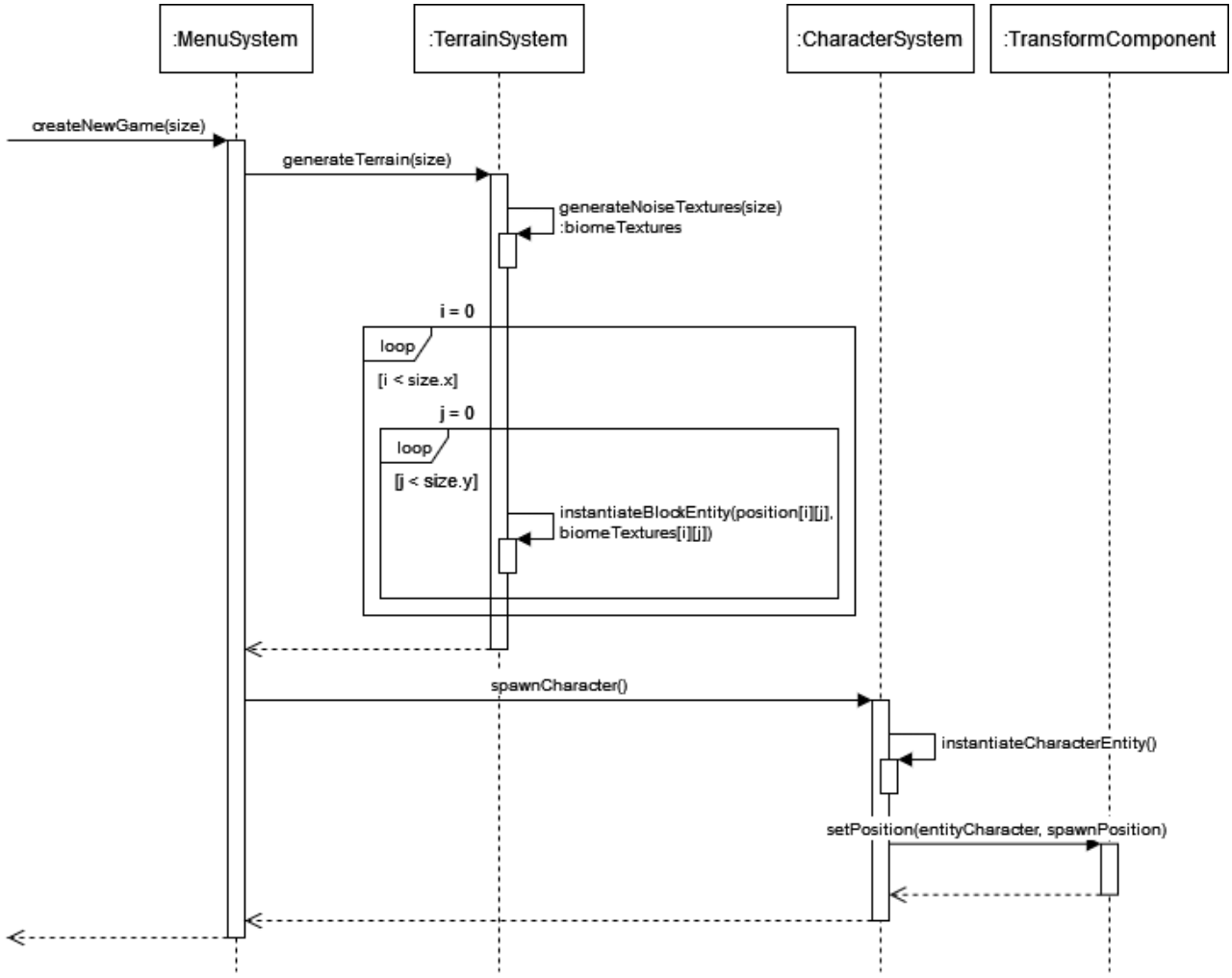


Figura 39: Diagrama de seqüència del CU01
Font: *Creació pròpia*

6.2.2 Col·locar un bloc

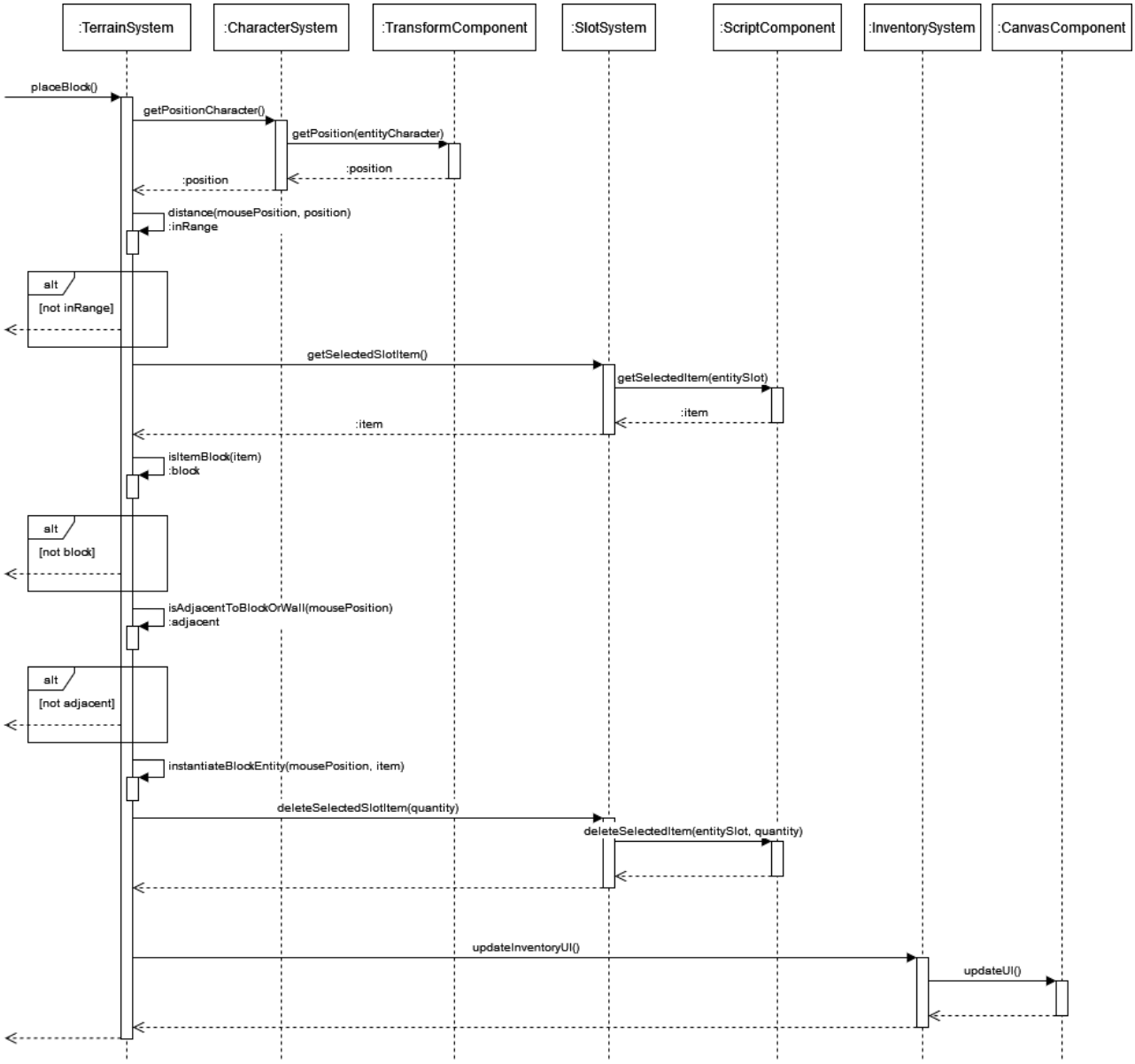


Figura 40: Diagrama de seqüència de l'acció de col·locar un bloc del CU03
Font: *Creació pròpia*

6.2.3 Fabricar un ítem

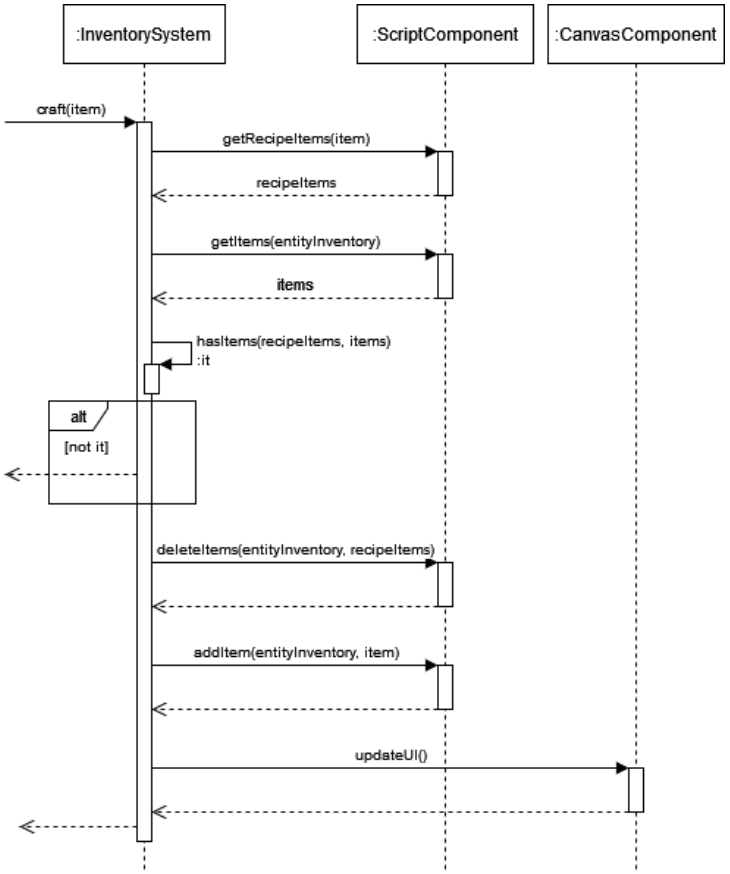


Figura 41: Diagrama de seqüència de l'acció de fabricar un ítem del CU03
Font: *Creació pròpia*

6.2.4 Reaparèixer el personatge

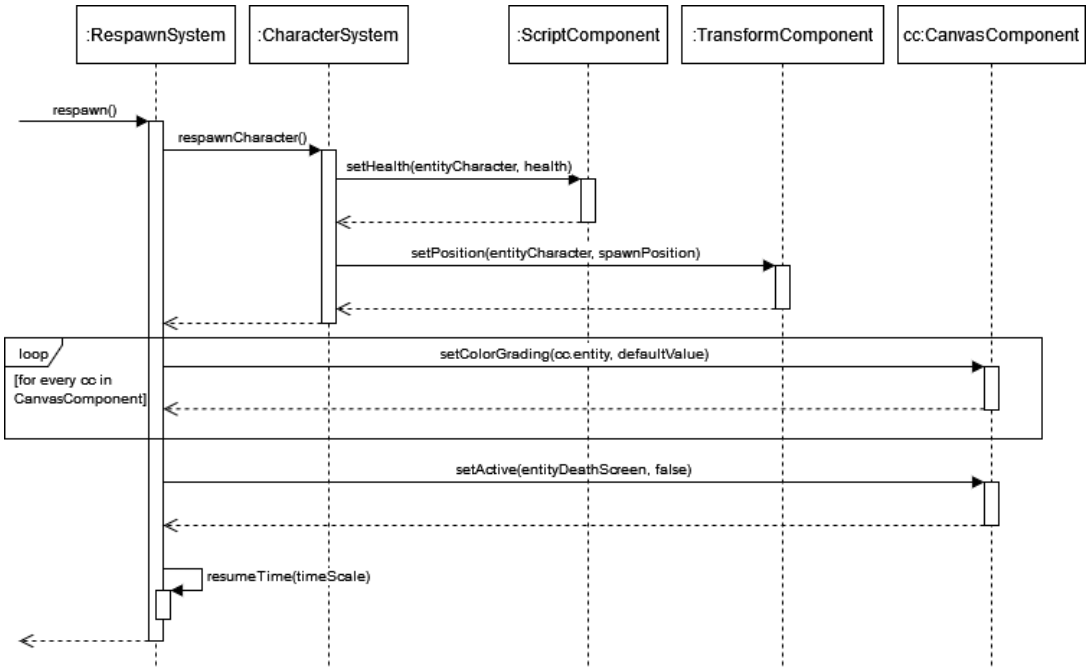


Figura 42: Diagrama de seqüència del CU06
Font: *Creació pròpia*

7 Implementació

En aquest apartat s'explica el procés d'implementació del videojoc i les eines utilitzades en aquest, mostrant el funcionament del framework, l'organització que s'ha seguit amb aquest, exemples de nomenclatures, i l'explicació d'algunes parts de codi que he considerat més importants o d'una complexitat elevada, entre d'altres.

7.1 Unity

Quan es crea un nou projecte de Unity, el primer que es veu és una jerarquia buida dins del que s'anomena *Scene*, o escena. Les escenes contenen els diferents objectes del joc, i poden ser emprades per crear un menú principal, nivells individuals o qualsevol altra cosa. En cada escena es col·loquen els diferents entorns, obstacles i decoracions, bàsicament dissenyant i construint el joc. En el cas d'ECS, una gran part dels diferents objectes que componen l'escena són les entitats, les quals tenen els diferents components gestionats pels sistemes. Tot seguit es mostra l'escena d'una partida del videojoc Lost Hero.

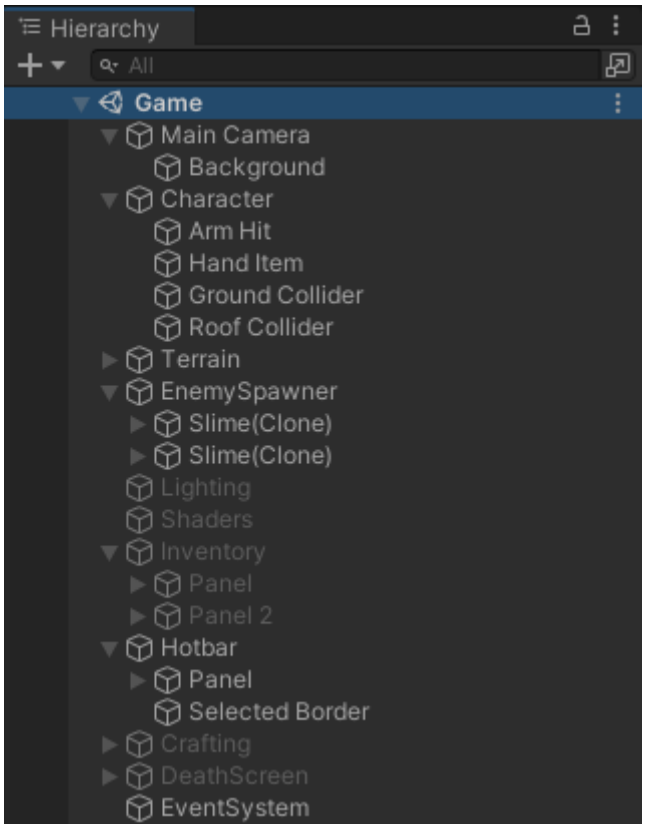


Figura 43: Jerarquia de l'escena Game
***Font:** Creació pròpia amb Unity*

En la següent figura podem veure un exemple d'entitat amb els seus components, en aquest cas els components de l'entitat Character de la figura anterior, que correspondrien al personatge que el jugador controla.

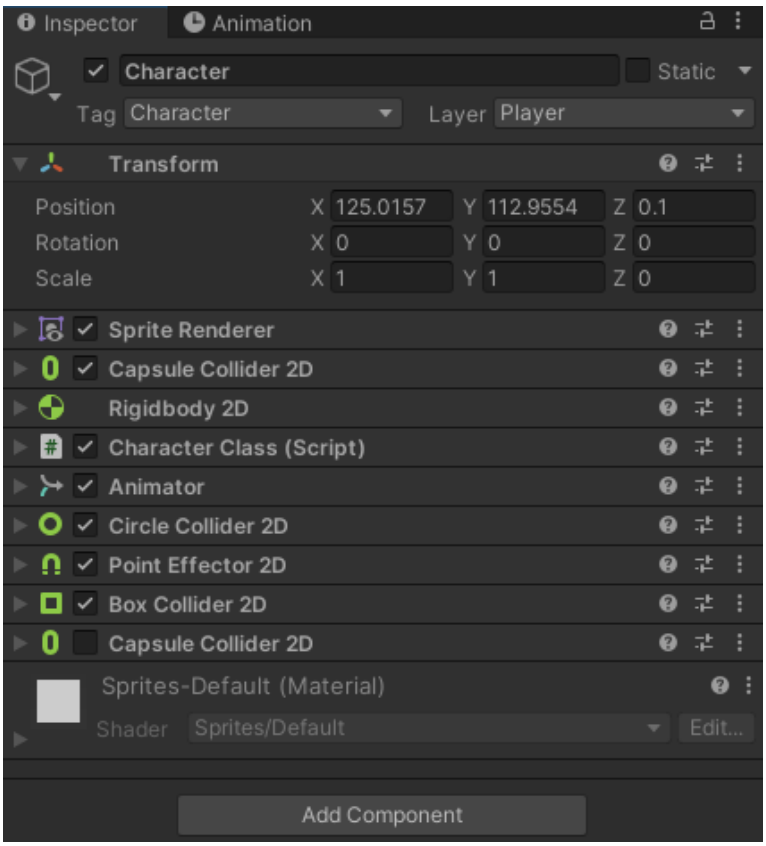


Figura 44: Components de l'entitat Character
Font: *Creació pròpia amb Unity*

7.2 Organització

En aquest punt s'exposa l'organització del projecte que s'ha seguit dins de Unity i una explicació en detall de cada secció que composa el videojoc si és necessària. A la figura següent es pot veure com estàn organitzats els diferents assets o elements d'aquest projecte en diferents carpetes.

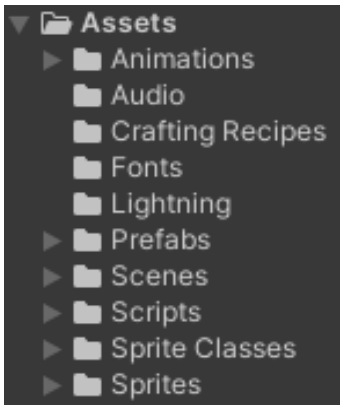


Figura 45: Organització dels assets
Font: *Creació pròpia amb Unity*

7.2.1 Animacions

A la carpeta *Animations* es troben totes les animacions del videojoc. Aquestes engloben les accions del personatge i dels diferents enemics, com són les accions de saltar, caminar o utilitzar una eina, i els diferents estats del personatge i dels enemics, com rebre dany. També està l'animació dels blocs quan s'estan destruint, per indicar al jugador que està realitzant l'acció en el bloc específic. A continuació es mostra un exemple d'animació, en aquest cas l'animació del personatge corrent.

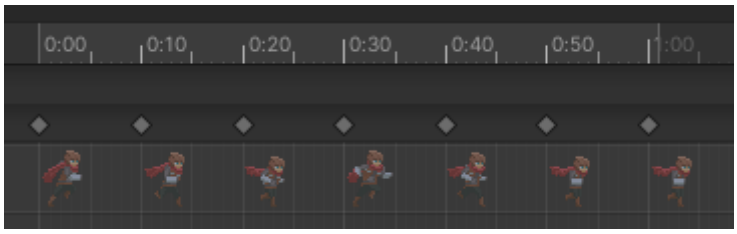


Figura 46: Animació Run del personatge
Font: *Creació pròpia amb Unity*

Les diferents transicions entre animacions es gestionen mitjançant el component *Animator* que proporciona Unity. La figura següent és un exemple de les transicions de les diferents animacions que té el personatge.

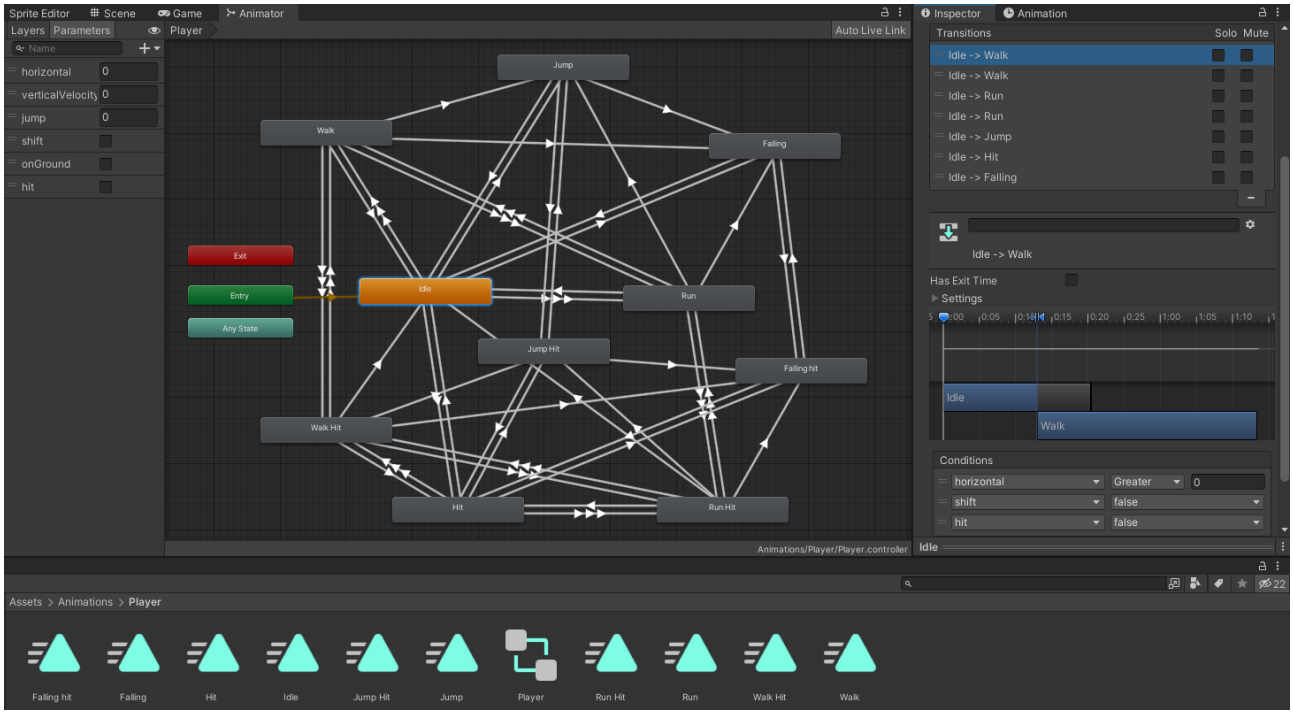


Figura 47: Transicions d'animació del personatge
Font: *Creació pròpia amb Unity*

Com es pot veure a l'esquerra de la imatge anterior, hi ha diversos paràmetres. Aquestes variables són les que permeten una comunicació amb el component Animator i els diferents sistemes. Per exemple, el paràmetre *horizontal* és diferent que 0 quan l'usuari fa l'input corresponent de moure el personatge, en aquest cas, pressionant la tecla 'a' o 'd'. També hi ha paràmetres que depenen d'altres factors, com és el cas del paràmetre *onGround*, un booleà que és cert quan el personatge està col·lisionant amb un bloc sota els seus peus.

A la dreta de la imatge podem veure les diferents propietats de l'animació *Idle*, la qual és l'animació del personatge quan aquest està quiet. Com a exemple, en l'inspector està seleccionada la transició *Idle* → *Walk*, amb les condicions per canviar de l'animació Idle a l'animació Walk quan *horizontal* és més gran que 0, *shift* és fals i *hit* és fals.

A continuació es troba un extracte d'una part del codi del *CharacterSystem*, on es pot veure com s'agafen els valors dels paràmetres i com s'assignen aquests a l'Animator corresponent. En aquest cas, *entity* es refereix a l'entitat *Character*.

```
//left/right click
if (clicks <= 0)
{
    if (hit || place)
    {
        hitAnimation = hit;
        placeAnimation = place;
        clicks = 1f / clicksPerSecond;
    }
    else
    {
        hitAnimation = false;
        placeAnimation = false;
    }
}

//running
if (shift)
    horizontal *= 1.5f;

horizontal = Input.GetAxis("Horizontal");
hit = Input.GetMouseButton(0);
place = Input.GetMouseButton(1);
jump = Input.GetAxisRaw("Jump");

verticalVelocity = entity.transform.rigidBody.velocity.y;

//run or walk
if (Input.GetKey(KeyCode.LeftShift) || Input.GetKey(KeyCode.RightShift))
    shift = true;
else
    shift = false;

//animator parameters
entity.GetComponent<Animator>().SetFloat("horizontal", horizontal);
entity.GetComponent<Animator>().SetFloat("verticalVelocity", verticalVelocity);
entity.GetComponent<Animator>().SetFloat("jump", jump);
entity.GetComponent<Animator>().SetBool("shift", shift);
entity.GetComponent<Animator>().SetBool("onGround", onGround || notFalling);
entity.GetComponent<Animator>().SetBool("hit", hitAnimation || placeAnimation);
```

Figura 48: Codi extret del CharacterSystem
Font: *Creació pròpia amb VisualStudio*

7.2.2 Àudio

En la carpeta *Audio* es troben tots els clips de música i efectes de so disponibles en el videojoc, els quals són reproduïts pel component AudioListener de la càmera quan s'indica en el codi d'altres components i scripts. La música ambient del videojoc és de twinmusicom [30].

7.2.3 Receptes de fabricació

En la carpeta *Crafting Recipes* es troben totes les receptes de fabricació disponibles en el videojoc. Per poder crear i emmagatzemar com a fitxers *.asset* les instàncies de les diferents receptes, Unity permet declarar la classe *CraftingRecipe* amb l'opció *CreateAssetMenu* per així poder llistar-la individualment i modificar les variables públiques de la classe. Tot seguit es mostra un extracte del codi de la classe i un exemple d'instància serialitzada de la recepta per la fabricació d'una destral de coure amb els materials que necessita, en aquest cas 4 trossos de fusta i 2 lingots de coure, i l'ítem que es fabrica com a resultat.

```
[System.Serializable]
3 referencias
public struct ItemQuantityMaterials
{
    public Sprite spriteItem;
    [Range(1, 999)]
    public int quantity;
}

[System.Serializable]
2 referencias
public struct ItemQuantityResults
{
    public ItemClass item;
    [Range(1, 999)]
    public int quantity;
}

[CreateAssetMenu(fileName = "newCraftingRecipe",
    menuName = "Crafting Recipe")]
Script de Unity | 1 referencia
public class CraftingRecipe : ScriptableObject
{
    public List<ItemQuantityMaterials> materials;
    public List<ItemQuantityResults> results;
```

Figura 49: Codi extret de CraftingRecipe
***Font:** Creació pròpia amb VisualStudio*

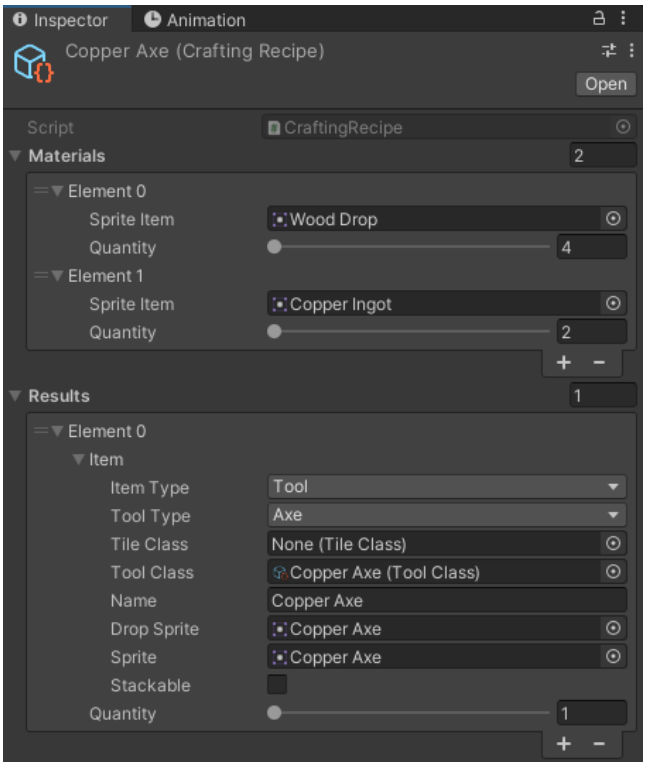


Figura 50: Exemple de CraftingRecipe
***Font:** Creació pròpia amb Unity*

7.2.4 Il·luminació

En la carpeta *Lighting* es troben l'asset de canalització de renderització universal (URP), un script preconstruït creat per Unity, juntament amb l'asset per renderitzar aquestes dades en 2D. Amb l'URP que ofereix Unity es pot aconseguir una il·luminació realista adequada per diversos estils d'art, en aquest cas una il·luminació per un videojoc 2D amb pixelart.

En una partida, a tot arreu on es veu el fons, és a dir, no hi ha cap paret ni bloc, es genera llum amb un radi de 7 blocs que disminueix segons la distància. El personatge també té una llum d'uns 7 blocs de radi amb una intensitat que va disminuint des del jugador fins a la distància màxima del radi de llum. Hi ha alguns blocs que també generen llum. A la figura següent es mostra un exemple de la il·luminació del videojoc.

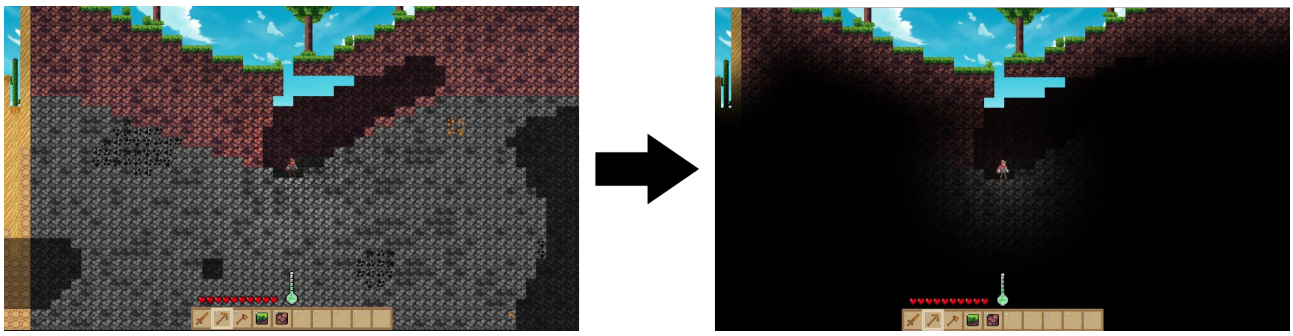


Figura 51: Exemple d'il·luminació
Font: *Creació pròpia amb Unity*

7.2.5 Prefabs

En la carpeta *Prefabs* es troben tots els GameObjects i entitats prefabricades. El sistema Prefab de Unity permet crear, configurar i emmagatzemar un GameObject o entitat completa amb tots els seus components, valors de propietat i els fills respectius, actuant així com una plantilla a partir de la qual es poden crear noves instàncies a l'escena, ja sigui en temps d'execució mitjançant codi o col·locades amb anterioritat. Per exemple, l'entitat Character, que representa el personatge, és un prefab que s'instancia quan es crea una partida, i a l'inici d'una partida nova sempre tindrà els mateixos components i els mateixos valors en aquests.

Els prefabs d'aquest projecte són el personatge, els slots de l'inventari, els drops dels ítems, els enemics i la llum, la qual s'instancia al crear o continuar una partida.

7.2.6 Escenes

En la carpeta *Scenes* es troben les dos escenes que componen el videojoc. Aquestes són l'escena del menú principal, on el jugador pot crear o continuar una partida, entrar a la configuració o sortir de l'aplicació, i l'escena de la partida. A la secció de Unity es mostra una figura d'exemple de l'escena d'una partida.

7.2.7 Scripts

En la carpeta *Scripts* es troben tots els scripts del videojoc, els quals es divideixen en tres subapartats. En el primer es troben els components Script de les entitats per guardar i modificar variables o altres components, com per exemple la vida del personatge, en el segon estan els diferents sistemes del videojoc, com el sistema del personatge per realitzar diverses accions, i en l'últim subapartat hi ha les diferents classes que actuen com a components sense entitats relacionades, com la recepta d'un ítem. Els scripts en Unity serveixen per fer connexions amb el funcionament intern d'aquest, i com s'ha explicat anteriorment, tots deriven de la classe MonoBehaviour, tot i que amb els sistemes s'implementa una altra classe per aquests, anomenada ComponentSystem. El llenguatge que suporten és C# i UnityScript, i com s'ha comentat a apartats

anteriors, en aquest projecte s'ha utilitzat el llenguatge C#. Tot seguit s'expliquen en detall els scripts més importants de tot el projecte.

TerrainSystem

El sistema *TerrainSystem* s'encarrega de gestionar els components de l'entitat Terreny. Quan es crea una nova partida, gestiona tota la generació procedural del món, incloent-hi els diferents biomes, la distribució dels chunks i la instanciació de tots els blocs que conté la partida. També gestiona els chunks actius d'una partida per tenir un rendiment òptim i les accions de construir i destruir del jugador, ja que aquestes afecten el terreny.

```
1 referencia
private void GenerateTextures()
{
    for (int i = 0; i < biomes.Length; i++)
    {
        biomes[i].caveTextureHigh = new Texture2D(worldSize.x, worldSize.y);
        GenerateNoiseTexture(biomes[i].caveFreqHigh, biomes[i].caveSizeHigh, biomes[i].caveTextureHigh);
        biomes[i].caveTextureLow = new Texture2D(worldSize.x, worldSize.y);
        GenerateNoiseTexture(biomes[i].caveFreqLow, biomes[i].caveSizeLow, biomes[i].caveTextureLow);
        for (int j = 0; j < biomes[i].ores.Length; j++)
        {
            biomes[i].ores[j].noiseTexture = new Texture2D(worldSize.x, worldSize.y);
            GenerateNoiseTexture(biomes[i].ores[j].frequency, biomes[i].ores[j].size, biomes[i].ores[j].noiseTexture);
        }
    }
}

3 referencias
private void GenerateNoiseTexture(float freq, float size, Texture2D noiseTexture)
{
    float pn;
    for (int x = 0; x < noiseTexture.width; x++)
    {
        for (int y = 0; y < noiseTexture.height; y++)
        {
            pn = Mathf.PerlinNoise((x + seed) * freq, (y + seed) * freq);
            if (pn > size)
                noiseTexture.SetPixel(x, y, Color.white);
            else
                noiseTexture.SetPixel(x, y, Color.black);
        }
    }
    noiseTexture.Apply();
}
```

Figura 52: Codi extret de TerrainSystem
***Font:** Creació pròpia amb VisualStudio*

La figura anterior és un extracte de la part del codi de TerrainSystem que crea les textures de les coves i els minerals de cada bioma. Aquestes textures es fan mitjançant l'algoritme *Perlin Noise*, un algoritme que genera un patró pseudoaleatori de valors del tipus float. UnityEngine té implementada per defecte l'estructura *Mathf*, amb la qual pots declarar la funció *Mathf.PerlinNoise* per invocar aquest algoritme, el qual és extremadament eficient. Els valors generats per l'algoritme s'expressen en un pla 2D i el soroll resultant no conté un valor aleatori en cada punt, sinó que consisteix en "ones" que tenen valors que augmenten i disminueixen gradualment al llarg del patró. A la figura següent es pot observar un exemple de l'algoritme PerlinNoise.

Figura 53: Exemple de PerlinNoise
***Font:** Mathf.PerlinNoise*

En el cas de les coves i els minerals he fet servir aquest algoritme per saber si en un punt en concret ha d'haver-hi un mineral o una cova. En aquest cas, com es pot observar en l'extracte de codi anterior, es té en compte la variable *size* per saber si el float resultant del PerlinNoise, que varia de 0 a 1, acaba sent blanc o negre. Cada bioma té les seves propietats de coves i minerals, i per cada bioma es generen dues textures per les coves i una textura per cada mineral. A continuació es mostra un exemple de textures del bioma Plains. Les dues imatges de dalt són les textures de les coves d'alta i baixa profunditat, i les dues imatges de baix són les textures dels minerals carbó i diamant respectivament. En les coves s'utilitza el color negre per definir-les, i en els minerals el color blanc.

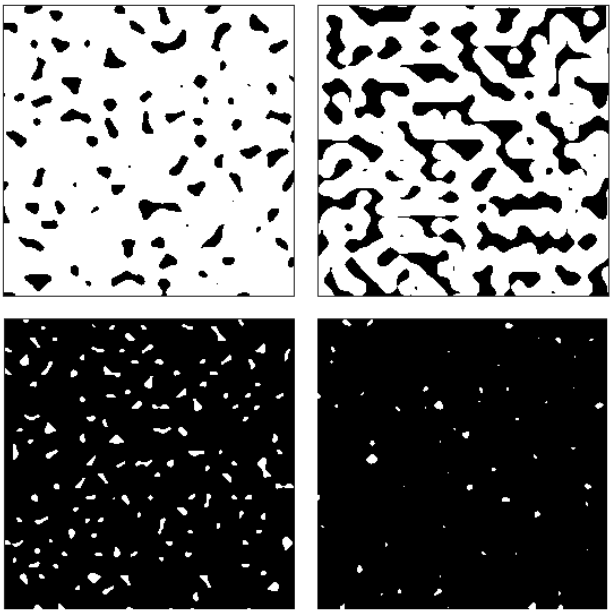


Figura 54: Textures de coves i minerals
***Font:** Creació pròpia amb Unity*

Cada bioma té uns atributs per generar les seves textures, i aquestes varien depenent de la *seed* de la partida per així fer-la més única. Una seed és bàsicament un número generat aleatòriament en crear una nova partida, i s'utilitza per crear procedimentalment el món del videojoc. Aquesta seed només té sentit en el context dels algoritmes utilitzats per generar el món, en aquest cas en les funcions Perlin Noise. No tots els valors d'un bioma estan vinculats a la seed, per exemple, la temperatura d'un bioma és un atribut únic establert per tots els biomes d'aquest tipus. A continuació es mostra un exemple de valors del bioma Plains d'una partida, creació pròpia a partir del component Script.

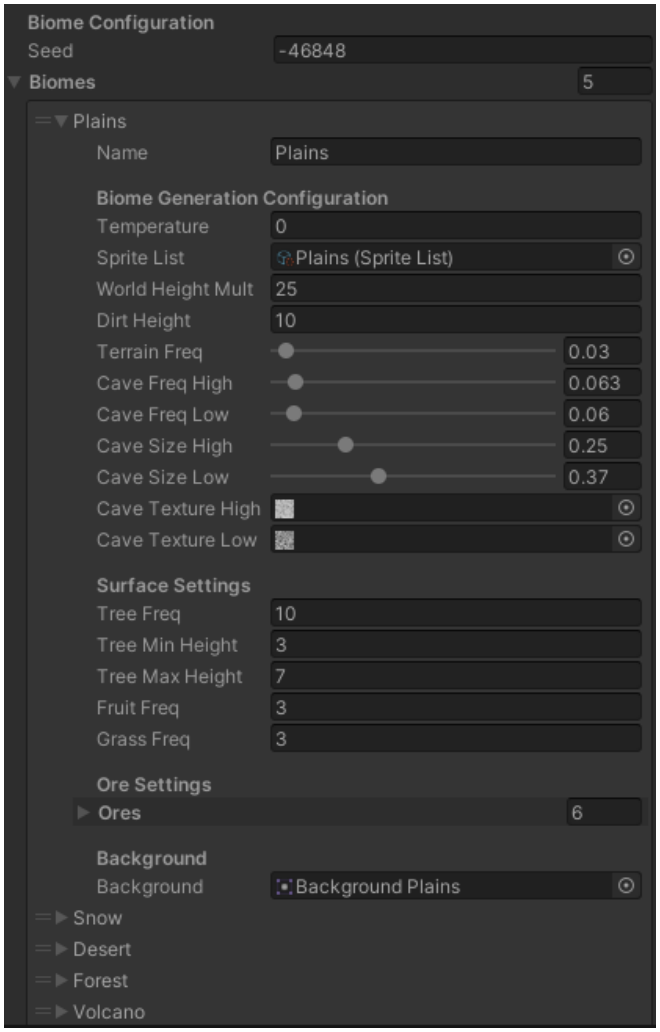


Figura 55: Atributs d'un bioma d'una partida
Font: *Creació pròpia amb Unity*

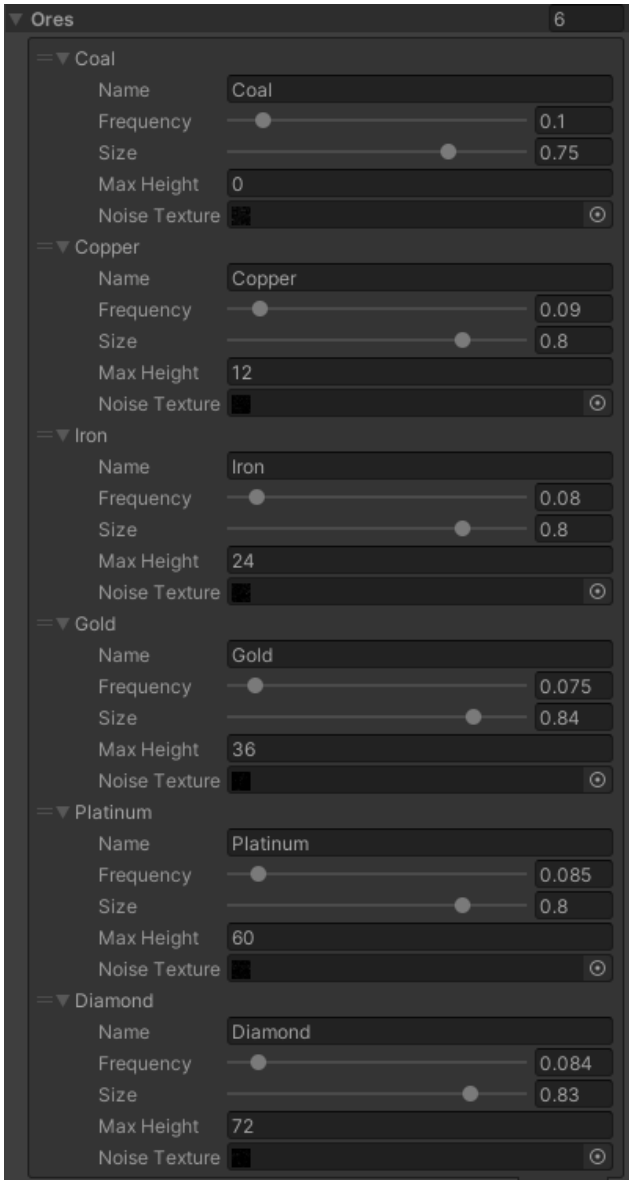


Figura 56: Atributs dels minerals d'un bioma
Font: *Creació pròpia amb Unity*

Un cop generades aquestes textures, es genera tot el terreny, bloc per bloc. Però abans de generar tot el món a partir d'aquestes textures, s'ha de definir la superfície del terreny. Per aconseguir això, el que he fet és utilitzar un bucle imbricat amb l'amplada i l'altura total del terreny a generar, i per cada iteració del bucle amb la mateixa amplada he aplicat l'algoritme Perlin Noise amb la seed de la partida per saber l'altura màxima corresponent, és a dir, he generat un Perlin Noise d'una dimensió per tenir el contorn de la superfície, la qual també és dependent del bioma en el qual es trobi el bloc corresponent. Per exemple, el bioma Plains té una superfície molt més plana que el bioma Snow, el qual té grans variacions simulant muntanyes. En la figura següent es mostra la línia de codi per saber l'altura màxima en la qual s'han d'instanciar els blocs i una part del codi on s'assignen i s'instancien aquests blocs. Aquest fragment de codi és només per conèixer el concepte general de la generació sense entrar en detall com està programat cada condicional i les funcions externes que es criden, ja que són masses línies de codi.

```
height = Mathf.PerlinNoise((x + seed) * currentBiome.terrainFreq, seed * currentBiome.terrainFreq) * currentBiome.worldHeightMult + worldHeight;
```

```
for (int y = 0; y < worldSize.y; y++)
{
    //player spawn point
    if (x == worldSize.x / 2)...

    //break
    if(y >= height)...

    //stone (and variants) or minerals/caves
    if (y < height - currentBiome.dirtHeight)...

    //dirt (and variants)
    else if(y < height - 1)...

    //grass top layer (and variants)
    else...

    //placing low height
    if (y < height / 2)...

    //placing high height
    else if (currentBiome.caveTextureHigh.GetPixel(x, y).r > 0.5f)...

    //generating trees and grass (and variants)
    if (y >= height - 1)...
}
```

Figura 57: Codi extret de TerrainSystem
Font: *Creació pròpia amb VisualStudio*

Sobre la gestió dels chunks actius per millorar el rendiment del videojoc, en el procés de la generació del terreny s'assigna cada bloc en un conjunt de blocs, anomenats chunks, que tenen una grandària de 16 blocs d'amplada per l'altura total del món. Quan es carrega una partida, s'activen els chunks més pròxims al personatge, i tots els altres es queden desactivats fins que el personatge s'apropa el suficient.

Amb aquest sistema he decidit implementar el requisit funcional RF14 "Món en bucle" mitjançant una translació del chunk sencer, és a dir, si el jugador està arribant al final del món i el pròxim chunk que s'activaria per proximitat no existeix perquè està ja al límit del terreny, s'activa el primer chunk del món i es transporta a la posició corresponent al final del món. Abans d'implementar aquest requisit d'aquesta manera, vaig plantejar diverses opcions. Una d'elles era teleportar el jugador a la punta oposada del món, però no quedava natural i no volia que es el jugador notés el canvi. Una altra opció era implementar dues càmeres i fer un canvi subtil quan el personatge arribés a una punta del món, però no vaig aconseguir un resultat fluid. Finalment vaig acabar implementant aquesta translació dels chunks amb el perill de baixar el rendiment del videojoc als extrems del món, però després de fer unes quantes proves amb el *Unity Profiler*, vaig comprovar que no hi havia una caiguda de rendiment destacable amb aquesta translació.

La figura següent mostra un exemple de tot un chunk d'una partida generada. Els blocs del chunk 15 seleccionat de l'escena surten seleccionats amb verd.

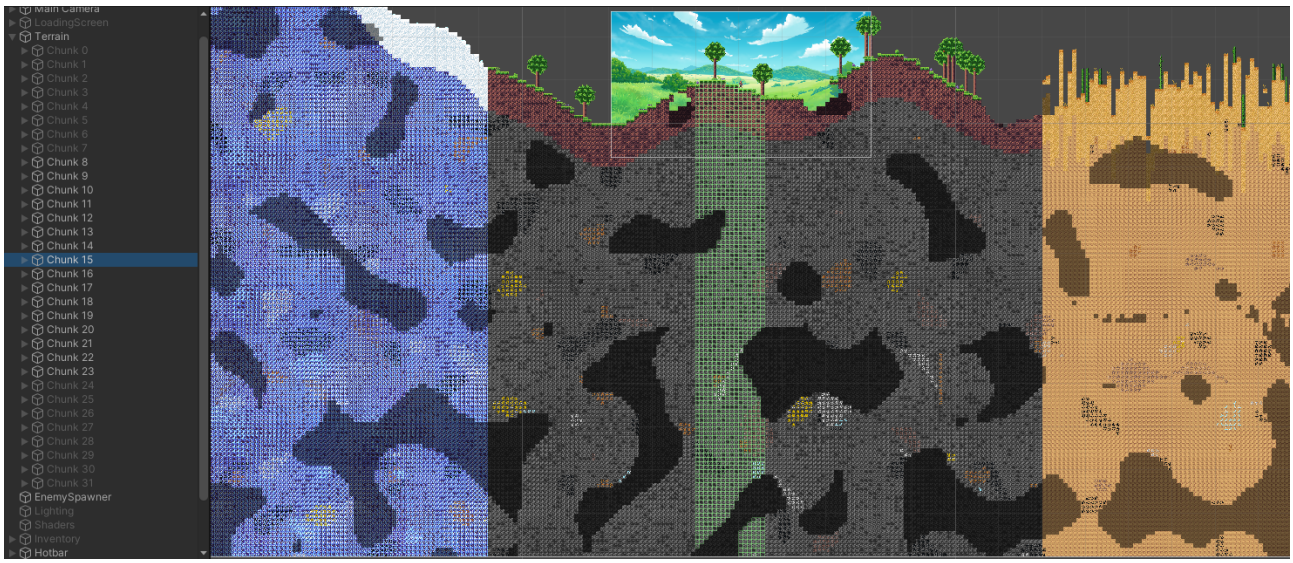


Figura 58: Exemple de chunk
Font: *Creació pròpia amb Unity*

7.2.8 Classes dels sprites

En la carpeta *Sprite Classes* es troben totes les classes dels diferents sprites del videojoc, que engloben les *Tile Classes* i les *Tool Classes*. De la mateixa manera que les receptes de fabricació, cada *tile* o tessel·la del videojoc està instanciada com una classe declarada amb diferents propietats, i el mateix amb les eines. A continuació es mostra un extracte del codi de la classe *Tile Class* i un exemple de la classe de la tessel·la *Stone* amb la seva ubicació.

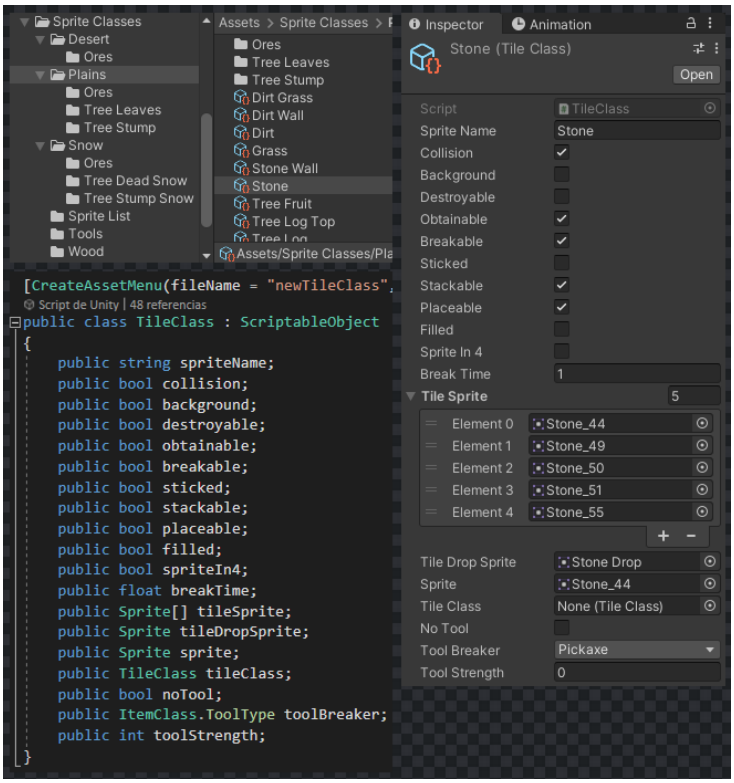


Figura 59: Exemple de TileClass amb el seu codi
Font: *Creació pròpia amb Unity i VisualStudio*

Com es pot veure en la figura anterior, cada tessel·la té moltes variables diferents. Tot seguit es descriuen aquestes variables explicant l'ús que tenen dins del videojoc.

- spriteName: string que indica el nom de la tessel·la en concret.
- collision: booleà que indica si la tessel·la té col·lisions.
- background: booleà que indica si el sprite de la tessel·la es renderitza en el fons del pla 2D. La vegetació i les parets són exemples de sprites *background*. En la imatge següent es pot veure el personatge davant dels sprites d'un arbre, i l'enemic Slime davant del sprite d'un arbust.

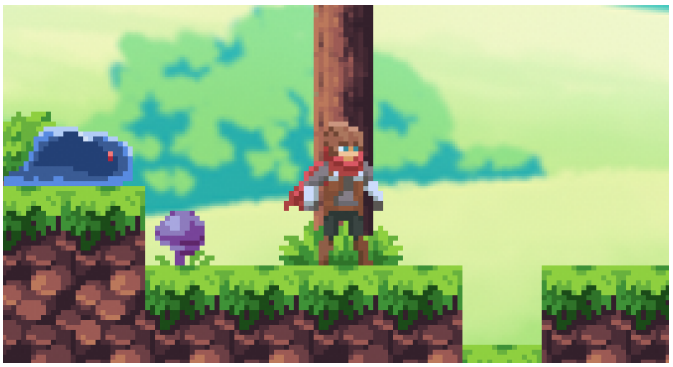


Figura 60: Exemple de tessel·la 'background'
Font: *Creació pròpia amb Unity*

- destroyable: booleà que indica si la tessel·la es pot reemplaçar per una altra tessel·la quan el jugador col·loca un bloc en la seva posició. L'herba i els bolets són exemples de tessel·les *destroyable*. Tot seguit es mostra un exemple d'aquest tipus de sprite.

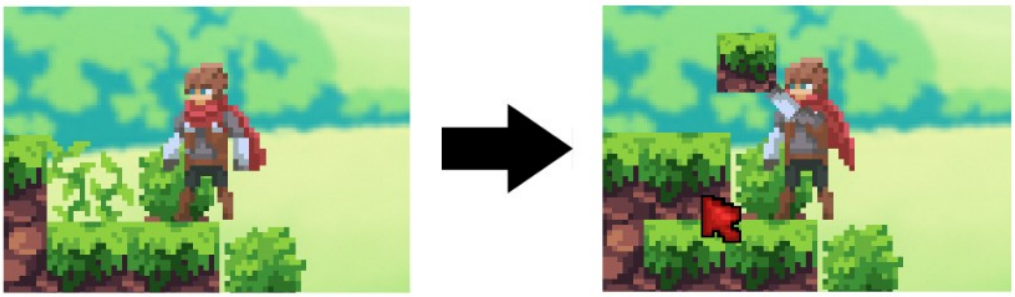


Figura 61: Exemple de tessel·la 'destroyable'
Font: *Creació pròpia amb Unity*

- obtainable: booleà que indica si la tessel·la es pot obtenir un cop destruïda. Si és el cas, quan la tessel·la és destruïda s'instancia el sprite corresponent en la seva posició, també anomenat *drop*, i el personatge el pot recollir-lo si passa per sobre, afegint-lo a l'inventari. Aquest drop té col·lisions amb el terreny però no entre altres drops, i també té una atracció cap al personatge si aquest està a una distància propera. A la següent figura es mostra un exemple.

Figura 62: Exemple de tessel·la 'obtainable' amb el seu drop
Font: *Creació pròpia amb Unity*

- breakable: booleà que indica si la tessel·la es pot destruir. La figura anterior també és un exemple de tessel·la *breakable*. No totes les tessel·la destructibles es poden obtenir, per això existeix aquesta diferenciació.
- sticked: booleà que indica si una tessel·la és dependent de les tessel·les del seu voltant. Quan una tessel·la *sticked* es destrueix, també es destrueixen les tessel·les *sticked* que tingui al damunt o als costats, i així successivament. Els cactus i els arbres estan formats per aquest tipus de tessel·les. La següent figura és un exemple de destrucció de dos arbres.

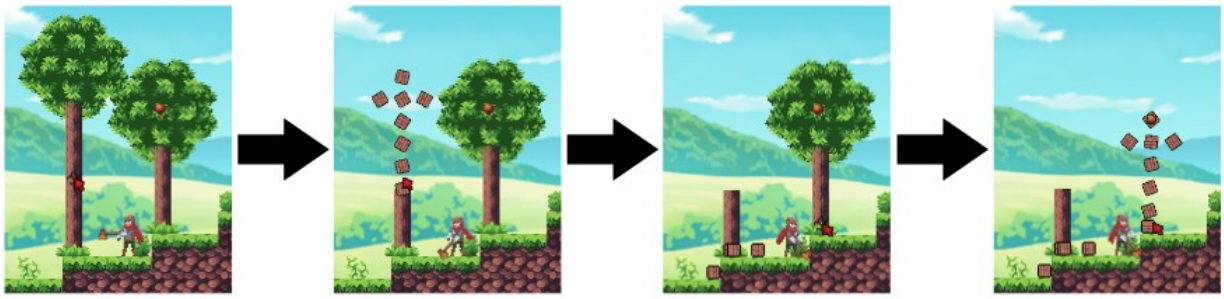


Figura 63: Exemple de tessel·les 'sticked'
Font: *Creació pròpia amb Unity*

- stackable: booleà que indica si la tessel·la es pot acumular en pila en un espai de l'inventari, és a dir, si es pot tenir el mateix ítem diversos cops en un mateix espai. Un exemple de tessel·la *stackable* és qualsevol bloc, i un exemple de sprite no acumulable és qualsevol eina.
- placeable: booleà que indica si la tessel·la es pot col·locar en el món. Per col·locar una tessel·la ha d'haver-hi una paret en el fons del pla 2D en la posició corresponent o una tessel·la amb col·lisió adjacent a la posició corresponent. Els blocs són un exemple d'aquest tipus.
- filled: booleà que indica si la tessel·la té dins una altra tessel·la, o dit d'una altra manera, si la tessel·la deixa anar una altra tessel·la quan es destrueix. Si és el cas, aquesta nova tessel·la ha d'estar indicada en la variable *tileClass* del sprite. Les tessel·les d'un arbre són un exemple d'aquest tipus, les quals tenen la tessel·la de fusta dins, sigui la soca, el tronc o les fulles.
- spriteIn4: booleà que indica si la tessel·la pertany a una seqüència de 4 sprites diferents. He creat aquest booleà pel disseny d'algunes tessel·les, com els de la neu o de la sorra, les quals estan formats per 4 sprites i s'han de col·locar en posicions específiques perquè quedin d'un estil uniforme.

- breakTime: float que indica el temps que es triga a destruir una tessel·la. Aquest temps es calcula tenint en compte el *breakTime* de la tessel·la i la potència de l'eina.
- tileSprite: conjunt de sprites que formen una tessel·la en concret. Quan s'instancia o es col·loca una tessel·la, es renderitza de manera aleatòria un sprite dels disponibles en *tileSprite,* sempre que no sigui una tessel·la *spriteIn4*, i s'assigna a la variable *Sprite*.
- tileDropSprite: sprite del drop de la tessel·la en concret, és a dir, el sprite que es deixa anar quan es destrueix la tessel·la.
- sprite: el sprite de la tessel·la.
- tileClass: tessel·la que està dins de la tessel·la en concret, si aquesta és *spriteIn4*.
- noTool: booleà que indica si la tessel·la es pot destruir sense una eina. Un exemple serien les herbes i els bolets.
- toolBreaker: eina amb la qual es destrueix la tessel·la. Com a exemple, els arbres es destrueixen amb destrals i els minerals amb pics.
- toolStrength: integer que indica la potència mínima de l'eina per poder destruir la tessel·la en concret. Per exemple, el diamant en mineral només es pot destruir amb un pic de platí o superior.

El fet de tenir tantes variables que condicionen una tessel·la en concret facilita la creació de noves tessel·les en un futur sense haver de crear una nova classe, ja que totes aquestes variables ja estan programades i implementades en el TerrainSystem.

7.2.9 Sprites

En la carpeta *Sprites* es troben tots els sprites del videojoc. Aquests són els fons per cada bioma, els diferents drops de cada sprite obtenible del joc, els enemics i el personatge amb les seves animacions, els elements de les interfícies, com la salut, la temperatura i l'inventari, els sprites de tots els blocs del terreny i les diferents eines. En l'apartat del document de disseny del videojoc es poden veure alguns d'aquests sprites.

8 Control de qualitat i proves

8.1 Unity Profiler

El Unity Profiler és una eina que proporciona Unity i que s'utilitza per obtenir informació sobre el rendiment de l'aplicació en execució, i es pot executar en l'editor de Unity per obtenir una visió general de l'assignació de recursos mentres es desenvolupa el projecte. He anat fent servir aquesta eina al llarg del projecte per veure el rendiment del videojoc per així millorar l'eficiència d'algoritmes i funcions mal optimitzades. Ha sigut una gran ajuda per complir el requisit no funcional RNF01. Tot seguit es mostra una imatge del Profiler d'una partida en execució.

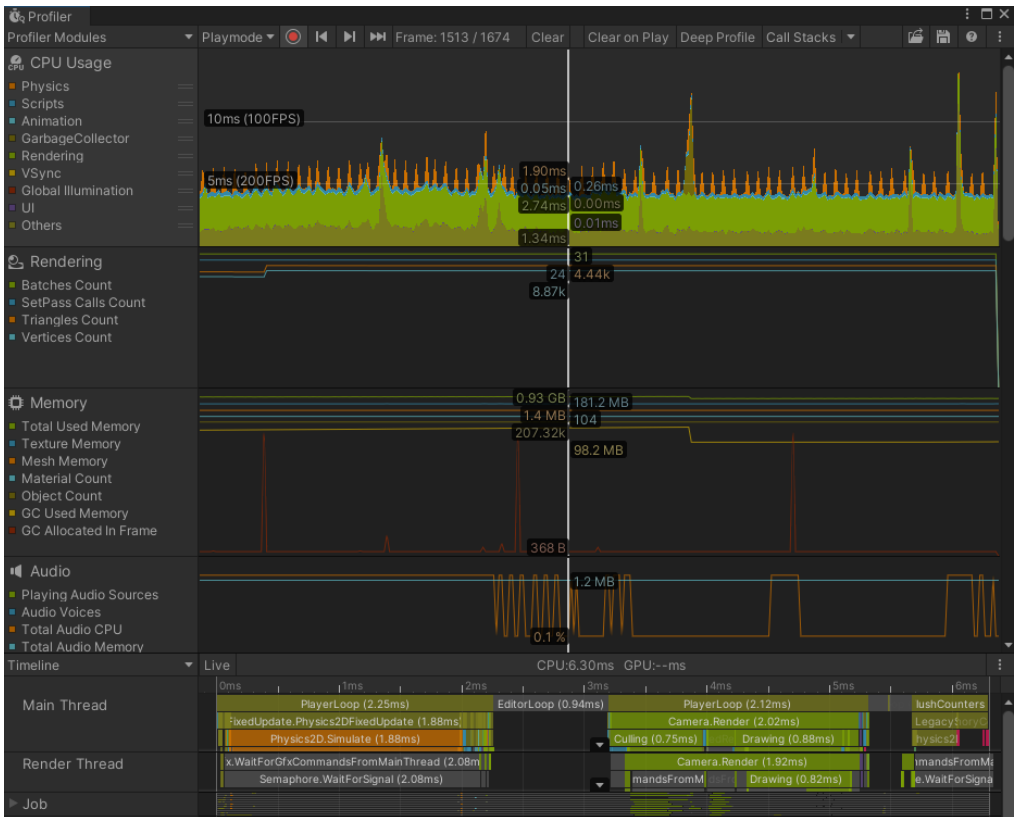


Figura 64: Rendiment d'un frame d'una partida amb el Unity Profiler
Font: *Unity Profiler*

Amb aquesta eina pots identificar com afecta el rendiment de l'aplicació les diferents parts del projecte, ja sigui el codi, els recursos utilitzats, la configuració de l'escena, la renderització de la càmera i la configuració de compilació, entre d'altres. Mostra resultats mitjançant gràfics, de manera que es poden visualitzar els pics del rendiment de l'aplicació en execució.

8.2 Excepcions i proves d'estrès

Per comprovar el correcte funcionament de les funcionalitats, s'han anat provant en l'editor de Unity un cop implementades, verificant que funcionen com s'espera en la definició dels requisits i tenint en compte els casos d'ús. L'editor de Unity permet visualitzar una simulació d'execució del

projecte actual amb tot el codi i els diferents assets de l'escena en concret.

En tenir el límit de ser una única persona provant que totes aquestes funcionalitats segueixen el comportament esperat, també s'han afegit excepcions en les parts necessàries del codi per evitar possibles errors en la versió final. Aquestes excepcions eviten que l'aplicació deixi de funcionar per algun error no previst i el jugador pugui continuar amb la seva partida.

També s'han realitzat proves d'estrès per veure si el rendiment és adequat, com per exemple generar el màxim d'enemics permesos a la vegada que hi ha el màxim d'ítems instanciats sense recollir en el terreny, amb altres funcionalitats actives que utilitzen la CPU, i comprovar que els fotogrames per segon continuen sent estables.

8.3 Condicions de satisfacció dels RNF

Com s'ha vist en l'apartat 5.1.2 sobre els requisits no funcionals, per cada un es van planejar unes condicions de satisfacció per saber si l'objectiu d'aquest està complert.

El requisit RFN01 sobre el rendiment del videojoc s'ha anat controlant durant el desenvolupament de les funcionalitats amb el Unity Profiler, com s'ha comentat anteriorment, revisant en tot moment que el rendiment del videojoc és òptim. Aquest requisit s'ha comprovat mitjançant diversos usuaris, els quals han provat el videojoc en els seus dispositius i m'han transmès si han tingut algun problema de rendiment durant l'execució d'aquest. Dels 12 usuaris que han provat el videojoc, 10 no han tingut cap problema de rendiment, i els altres dos han tingut alguna baixada de fotogrames en començar una partida i en realitzar alguna acció en concret dins d'una partida. D'aquests dos usuaris, un ha pogut continuar jugant sense cap altre contratemps, i l'altre usuari ha tingut un congelament del videojoc durant uns 5 segons. Aquests problemes de rendiment esmentats han sigut solucionats per futures versions. Amb aquestes dades he pogut concloure que el requisit no funcional del rendiment s'ha complert amb un 83% de satisfacció.

El requisit RFN02 sobre la interfície gràfica atractiva s'ha realitzat de la mateixa manera que l'anterior requisit amb els mateixos usuaris. Tot i que el pixelart no és el punt fort d'algun d'aquests usuaris, no he tingut cap resposta negativa per contemplar-la en contra del requisit.

El requisit RFN03 sobre l'emmagatzematge del videojoc s'ha complert, ja que aquest pesa poc més de 100 MB en total.

El requisit RFN04 sobre que el videojoc sigui multiplataforma s'ha complert. Unity et proporciona l'opció de configurar la *build* per diferents plataformes, incloses Windows, Mac i Linux, és a dir, l'executable del videojoc final amb els seus arxius.

El requisit RFN05 no s'ha acabat implementant, i el videojoc està disponible només en anglès. Això és degut al fet que he preferit invertir el temps extra per la traducció de tots els elements del joc en anglès en implementar i millorar altres requisits. Aquest requisit no funcional es queda pendent per ampliacions futures.

9 Desplegament del videojoc

Unity ofereix diverses opcions per desplegar l'aplicació en diferents plataformes i per ajustar la configuració d'aquesta. A continuació es mostra aquesta configuració per desplegar el videojoc. Com es pot veure, es poden seleccionar les escenes del projecte que seran incloses, en aquest cas, les dues escenes explicades en apartats anteriors, i també seleccionar en quina plataforma es podrà utilitzar i quina sèrie d'arquitectura de conjunt d'instruccions tindrà.

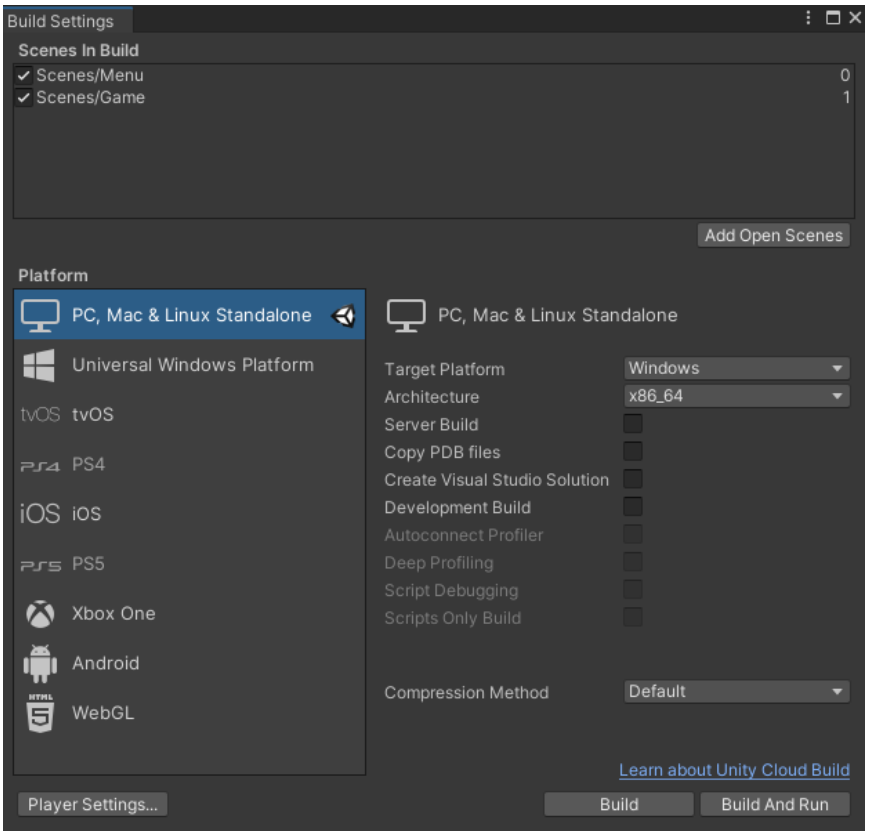


Figura 65: Configuració de desplegament
Font: *Unity*

Un cop el videojoc està desplegat amb el seu executable i els arxius necessaris perquè funcioni, l'usuari ja pot jugar mitjançant l'execució de l'executable en la plataforma adequada. El videojoc, com es comenta més endavant en l'apartat d'ampliacions futures, es publicarà en un futur en una plataforma de distribució de videojocs, i per aquesta raó en aquest document públic no s'adjunta un enllaç per descargar-lo per la possible distribució il·legal d'aquest.

10 Canvis respecte la planificació inicial

Al llarg del projecte hi ha hagut desviacions respecte a la planificació inicial amb les tasques a desenvolupar durant el projecte i la repercussió que han tingut amb els costos totals. A causa de la complexitat d'algunes funcionalitats, la necessitat de perfeccionar-les i el temps limitat per dedicar en programar-les s'ha posposat el requisit d'il·luminació global i rebutjat el de la generació d'estructures a canvi de dedicar més temps en implementar els requisits més importants i perfeccionar el seu funcionament per un resultat òptim en el videojoc.

En un videojoc el més rellevant és la jugabilitat. Pot tenir millors gràfics i una història espectacular, però no es pot considerar un bon videojoc sense un disseny ben fet de les mecàniques ni sense uns controls fluids que s'adaptin al videojoc, fent que el jugador es senti còmode jugant i així millorant la seva experiència. Per aconseguir això, un dels principals reptes és que el videojoc tingui un rendiment i que els requisits principals estiguin molt polits. És per això que he acabat dedicant molt més temps en la programació de les accions del personatge i en el testeig de totes les funcionalitats amb prioritat alta, per així tenir un videojoc que ofereixi la millor experiència possible. Això ha ocasionat canvis respecte a la planificació inicial. A continuació s'exposen en detall aquestes desviacions.

10.1 Desviacions

- S'ha posposat el subobjectiu d'implementar un sistema d'il·luminació global. La tasca per aquest subobjectiu es va començar a implementar en la data indicada en la planificació inicial, a la vegada que s'estaven testejant les implementacions més importants, les quals eren sobre la generació del terreny seguit dels moviments i accions del personatge. En tenir diversos problemes amb aquestes funcionalitats tan crucials, vaig acabar decidint posposar el sistema d'il·luminació, el qual no tenia una prioritat molt elevada, pels dies extra de la planificació descrits en el diagrama de Gantt que estaven reservats per aquestes desviacions.

- S'ha rebutjat el subrequisit de la generació d'estructures inclòs en el requisit de la generació del terreny. Vaig implementar un parell d'estructures per dos biomes diferents i afectava bastant al rendiment de la generació del terreny. Després d'invertir un parell de dies en millorar, vaig decidir rebutjar aquest subrequisit per millorar la generació dels biomes amb característiques úniques en comptes de generar estructures aleatòriament pel mapa. La idea no es descarta pel futur del videojoc.

- Han sorgit més bugs dels que s'esperaven, endarrerint tota la planificació inicial. Com s'ha comentat al principi d'aquest apartat, he donat molta importància en polir el videojoc per oferir la millor experiència possible, i això ha resultat en una inversió extra de temps en buscar solucions d'eficiència, de jugabilitat i d'errors d'algoritmes implementats.

- El requisit de la temperatura del personatge es va afegir després de fer la planificació inicial, ja que em semblava una bona idea donar als biomes una característica més única per la qual el jugador generés un interès per investigar-los i creés la necessitat de fabricar els ítems adients. Com s'ha

explicat en la implementació, un exemple és que sense l'ítem que calenta el personatge, aquest perd salut cada segon si es troba en el bioma de neu a causa de la temperatura d'aquest bioma. Aquest nou requisit ha fet una desviació en la planificació inicial.

- S'ha dedicat més temps del previst en els apartats del model conceptual, de l'arquitectura del videojoc i del diagrama d'aquesta. Això és degut a la necessitat d'explicar i justificar el funcionament del videojoc a alt nivell per poder entendre millor els següents apartats.

10.2 Planificació final

Respecte la planificació inicial, la metodologia de treball Agile, específicament Kanban, s'ha seguit utilitzant fins al final del projecte mitjançant el seguiment de les tasques amb Trello, com s'ha comentat en l'apartat de la planificació inicial. He decidit seguir fent servir aquesta metodologia per la flexibilitat d'adaptar la planificació proposada en qualsevol moment del desenvolupament del projecte i per l'organització que m'ha donat en prioritzar les tasques importants del projecte. En la última pàgina d'aquest apartat es mostra el diagrama de Gantt actualitzat. Els dies amb diverses tasques significa que s'han anat desenvolupant de manera intermitent.

10.3 Pressupost actualitzat

A continuació es mostra la taula resultant de les hores invertides juntament amb l'aproximació d'hores restants, amb el càlcul del cost estimat respecte aquestes hores. Només es mostren les taules dels rols amb les hores reals, dels imprevistos i del cost final. Les altres taules mostrades en la planificació inicial no han tingut cap canvi però s'han tingut en compte per la modificació del preu total.

Imprevistos

Risc	Hores	Rol implicat	Cost (€)
Errors de codi	30	Programador	546,9
Data ajustada	2	Cap de projecte	44,86
Poca experiència	10	Programador	182,3
Pèrdua de dades	0	Analista	0
TOTAL	-	-	**774,06**

Taula 70: Costos reals en imprevistos i contingències
Font: *Creació pròpia*

Recursos humans

Codi tasca	Hores	Cap de projecte	Analista	Arquitecte de software	Programador	Provador	Dissenyador gràfic	Cost (€)
T1	**85**	**85**	**0**	**0**	**0**	**0**	**0**	**1.906,55**
T1.1	25	25	0	0	0	0	0	560,75
T1.2	15	15	0	0	0	0	0	336,45
T1.3	15	15	0	0	0	0	0	336,45
T1.4	20	20	0	0	0	0	0	448,6
T1.5	10	10	0	0	0	0	0	224,3
T2	**35**	**0**	**5**	**0**	**30**	**0**	**0**	**636**
T2.1	35	0	5	0	30	0	0	636
T3	**95**	**0**	**25**	**40**	**0**	**0**	**30**	**1980,1**
T3.1	25	0	25	0	0	0	0	445,5
T3.2	40	0	0	40	0	0	0	1148,8
T3.3	30	0	0	0	0	0	30	415,8
T4	**190**	**0**	**0**	**0**	**190**	**0**	**0**	**3463,7**
T4.1	40	0	0	0	40	0	0	729,2
T4.2	50	0	0	0	50	0	0	911,5
T4.3	20	0	0	0	20	0	0	364,6
T4.4	10	0	0	0	10	0	0	183,2
T4.6	30	0	0	0	30	0	0	546,9
T4.7	10	0	0	0	10	0	0	183,2
T4.8	15	0	0	0	15	0	0	273,45
T4.9	15	0	0	0	15	0	0	273,45
T5	**50**	**0**	**0**	**0**	**0**	**50**	**0**	**635,5**
T5.1	40	0	0	0	0	40	0	508,4
T5.2	10	0	0	0	0	10	0	127,1
T6	**90**	**20**	**70**	**0**	**0**	**0**	**0**	**1.696**
T6.1	70	0	70	0	0	0	0	1.247,4
T6.2	20	20	0	0	0	0	0	448,6
TOTAL	**545**	**105**	**100**	**40**	**220**	**50**	**30**	**10.190,75**

Taula 71: Rols amb les hores reals i cost per cada tasca

Font: *Creació pròpia*

Cost total

Factor	Cost (€)
Recursos humans	10.190,75
Hardware	147,84
Software	0
Despeses generals	389,85
Imprevistos	774,06
TOTAL	**11.502,5 + IVA(21%) = 13.918,03**

Taula 72: Cost total del projecte
Font: *Creació pròpia*

Les hores estimades del projecte eren 550 i el cost estimat total de 16.236,54€, per tant, hi ha hagut una desviació total de 5 hores i 2.318,51€ menys. Aquestes dades indiquen que la planificació inicial del projecte es va fer correctament.

10.4 Diagrama de Gantt Final

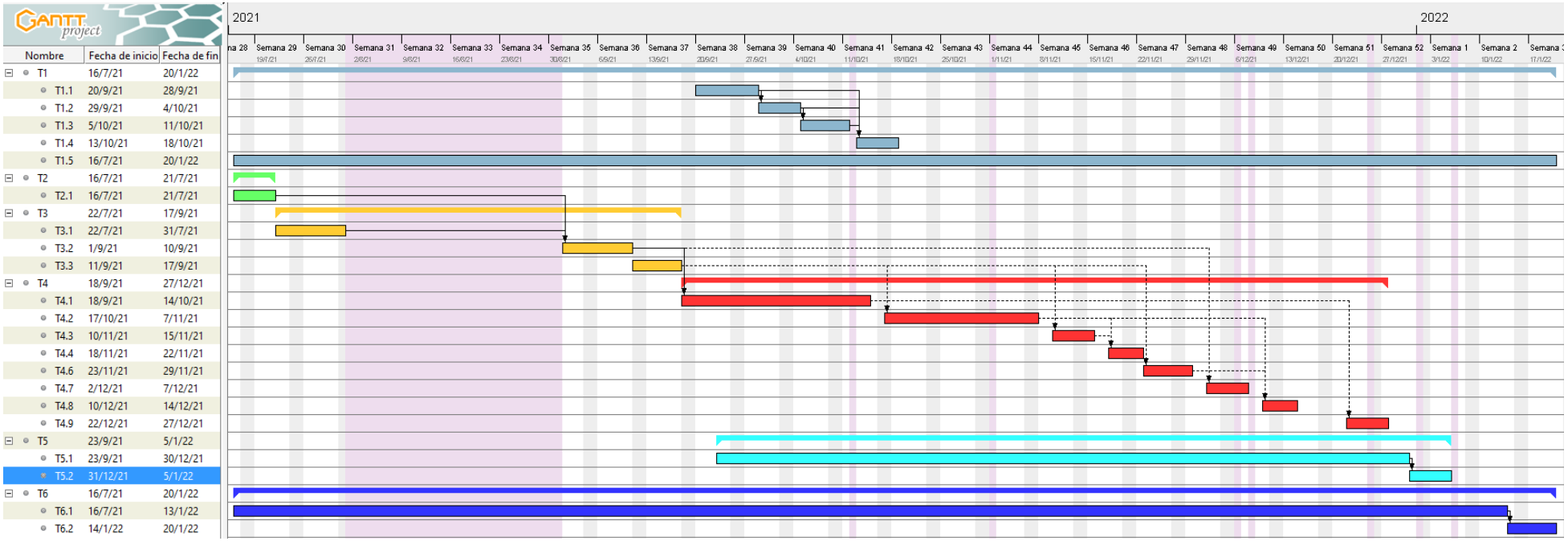


Figura 66: Diagrama de Gantt final
Font: *Creació pròpia amb el programa GanttProject*

11 Competències tècniques de software

A continuació es justifica l'aplicació de les competències tècniques de l'especialitat software plantejades a la inscripció del projecte.

CES1.1: Desenvolupar, mantenir i avaluar sistemes i serveis software complexos i/o crítics. [Bastant]

S'ha hagut d'estudiar l'arquitectura del framework de Unity i implementar un patró arquitectònic viable i eficient pel videojoc i compatible amb el framework. S'ha desenvolupat el videojoc des de zero i per cada funcionalitat implementada s'ha comprovat que el sistema funcionés correctament.

CES1.3: Identificar, avaluar i gestionar els riscos potencials associats a la construcció de software que es poguessin presentar. [Bastant]

S'han identificat, avaluat i gestionat tots els riscos potencials del projecte, fent una anàlisi en l'apartat dels possibles obstacles i riscos 3.2, i proposant solucions i plans alternatius a aquests riscos en l'apartat 4.5.

CES1.7: Controlar la qualitat i dissenyar proves en la producció de software. [Bastant]

En tot moment del desenvolupament del videojoc s'han anat fent proves de les funcionalitats implementades, com s'ha indicat en la tasca T5.1 sobre el testeig de les funcionalitats, i s'ha anat controlant que la qualitat d'aquestes és la desitjada pel videojoc final. També s'han realitzat proves d'estrès per comprovar que el rendiment del videojoc és òptim.

CES2.1: Definir i gestionar els requisits d'un sistema software. [En profunditat]

Una de les parts més importants del projecte ha sigut la definició i gestió dels requisits del videojoc, descrits en l'apartat 5 d'especificació de requisits i tinguts en compte en la planificació inicial i durant tot el desenvolupament del projecte.

CES2.2: Dissenyar solucions apropiades en un o més dominis d'aplicació, utilitzant mètodes d'enginyeria del software que integrin aspectes ètics, socials, legals i econòmics. [Bastant]

S'ha dissenyat el videojoc en un domini d'aplicació del software, en aquest cas, multimèdia. En l'apartat de limitacions legals 1.4 es tracten els aspectes legals del projecte, i en l'apartat del pressupost 4.6 es tracten els aspectes econòmics a tenir en compte pel desenvolupament del projecte. Els aspectes socials i ètics es comenten en l'apartat de l'informe de sostenibilitat 4.7.

CES3.1: Desenvolupar serveis i aplicacions multimèdia. [En profunditat]

El desenvolupament d'aquest projecte defineix aquesta competència tècnica, ja que un videojoc és una aplicació multimèdia.

12 Conclusions

Un cop finalitzat aquest projecte puc afirmar que he complert l'objectiu i els subobjectius plantejats inicialment i que estic molt content amb el resultat final, ja que he pogut plasmar la idea inicial que tenia sobre el videojoc i la majoria de funcionalitats plantejades s'han implementat com s'havia previst. També he après molt sobre el motor Unity, que era un dels motius principals pels quals vaig escollir fer aquest projecte, i tinc més coneixements sobre el disseny de videojocs que em poden ajudar a enfocar la meva vida laboral en aquesta indústria.

A més a més estic content d'haver emprat diversos coneixements adquirits de diferents assignatures durant el grau. Per exemple, les assignatures de programació, EDA i PROP en aprendre els coneixements bàsics per programar, entendre diferents llenguatges i dur a terme projectes amb l'aplicació d'algoritmes i en realitzar una bona fase d'implementació, IES i AS a l'hora de dissenyar els diferents diagrames de seqüència i l'arquitectura de software, ER respecte a definir les parts interessades, el model conceptual i els requisits, VJ per introduir-me a Unity i aprendre el bàsic per fer un videojoc, i IDI en crear interfícies d'usuari funcionals, entre d'altres.

12.1 Ampliacions futures

Tinc pensat seguir amb el desenvolupament del videojoc, expandint funcionalitats i afegint-ne de noves. Algunes d'aquestes serien implementar un cicle dia i nit amb noves funcionalitats relacionades amb aquesta mecànica, dissenyar més biomes i nous blocs, més enemics, estructures i masmorres per explorar, afegir líquids com aigua i lava, implementar entitats neutrals, com animals salvatges i mascotes que t'ajudin, i fins i tot crear una història que el jugador pugui seguir dins del videojoc, entre d'altres. També vull continuar optimitzant i millorant les funcionalitats actuals, i en un futur arribar a treure el videojoc al mercat.

13 Bibliografia

[1] José David Muñoz (15/08/2020): *Tres mil millones de personas en todo el mundo juegan ahora a videojuegos, según informa un estudio.* Disponible en: https://www.hobbyconsolas.com/noticias/tres-mil-millones-personas-todo-mundo-juegan-ahora-videojuegos-informa-estudio-698121 [Consulta: 22/09/2021]

[2] Unity. Disponible en: https://unity.com/es [Consulta: 22/09/2021]

[3] Romain Dillet (05/09/2018): *Unity CEO says half of all games are built on Unity.* Disponible en: https://techcrunch.com/2018/09/05/unity-ceo-says-half-of-all-games-are-built-on-unity/ [Consulta: 22/09/2021]

[4] Nación Gamer (17/03/2021): *La pandemia de Covid-19 dejó un 30% más de gamers y 40% más de ganancias en la industria.* Disponible en: https://www.marca.com/claro-mx/esports/2021/03/18/60529aa0268e3eb85a8b4576.html [Consulta: 23/09/2021]

[5] PEGI. Disponible en: https://pegi.info/es/que-significan-las-etiquetas [Consulta: 17/12/2021]

[6] Código de Propiedad Intelectual. Disponible en: https://www.boe.es/biblioteca_juridica/codigos/codigo.php?id=87&modo=2&nota=0&tab=2 [Consulta: 17/12/2021]

[7] Jacob (07/01/2019): *How Many Video Games Exist?* Disponible en: https://gamingshift.com/how-many-video-games-exist/ [Consulta: 23/09/2021]

[8] Starbound. Disponible en: https://playstarbound.com/ [Consulta: 26/09/2021]

[9] Astroneer. Disponible en: https://astroneer.space/ [Consulta: 26/09/2021]

[10] Minecraft. Disponible en: https://www.minecraft.net/es-es [Consulta: 26/09/2021]

[11] GitHub. Disponible en: https://github.com/ [Consulta: 25/09/2021]

[12] Santalucía Impulsa (03/02/2021): *Metodología Agile: ¿Qué es y para qué sirve?* Disponible en: https://www.santaluciaimpulsa.es/metodologia-agile-que-es-para-que-sirve/ [Consulta: 26/09/2021]

[13] Laia Gilibets (11/11/2020): *Qué es la metodología Kanban y cómo utilizarla.* Disponible en: https://www.iebschool.com/blog/metodologia-kanban-agile-scrum/ [Consulta: 26/09/2021]

[14] Trello. Disponible en: https://trello.com/ [Consulta: 26/09/2021]

[15] Juan Carlos (16/07/2021): *¿Qué es el Pixel Art?* Disponible en: https://www.tokioschool.com/noticias/que-es-el-pixel-art/ [Consulta: 02/10/2021]

[16] Pixlr. Disponible en: https://pixlr.com/es/ [Consulta: 02/10/2021]

[17] Bryan Wirtz (24/06/2021): S*haders in Game Design: Origin, Basic Design Types, and How to Create Your Own.* Disponible en: https://www.gamedesigning.org/learn/shaders/

[Consulta: 03/10/2021]

[18] PayScale: *Average Project Manager, Software Development Salary in Spain.* Disponible en: https://www.payscale.com/research/ES/Job=Project_Manager%2C_Software_Development/Salary [Consulta: 08/10/2021]

[19] PayScale: *Average Programmer Analyst Salary in Spain.* Disponible en: https://www.payscale.com/research/ES/Job=Programmer_Analyst/Salary [Consulta: 08/10/2021]

[20] PayScale: *Average Software Architect Salary in Spain.* Disponible en: https://www.payscale.com/research/ES/Job=Software_Architect/Salary [Consulta: 08/10/2021]

[21] PayScale: *Average Software Engineer/Developer/Programmer Salary in Spain.* Disponible en: https://www.payscale.com/research/ES/Job=Software_Engineer_%2F_Developer_%2F_Programmer/Salary [Consulta: 08/10/2021]

[22] PayScale: *Average Software Tester Salary in Spain.* Disponible en: https://www.payscale.com/research/ES/Job=Software_Tester/Salary [Consulta: 08/10/2021]

[23] PayScale: *Average Graphic Designer Salary in Spain.* Disponible en: https://www.payscale.com/research/ES/Job=Graphic_Designer/Salary [Consulta: 08/10/2021]

[24] Unity: *Animator.* Disponible en: https://docs.unity3d.com/ScriptReference/Animator.html [Consulta: 10/10/2021]

[25] Unity Technologies (11/05/2021): *Script lifecycle overview.* Disponible en: https://docs.unity3d.com/Manual/ExecutionOrder.html [Consulta: 10/11/2021]

[26] Unity Technologies (18/03/2020): *ECS concepts.* Disponible en: https://docs.unity3d.com/Packages/com.unity.entities@0.11/manual/ecs_core.html [Consulta: 12/11/2021]

[27] Unity Technologies (19/10/2020): *Systems.* Disponible en: https://docs.unity3d.com/Packages/com.unity.entities@0.11/manual/ecs_systems.html [Consulta: 13/11/2021]

[28] Unity Technologies (01/08/2017): *GameObjects*. Disponible en: https://docs.unity3d.com/Manual/GameObjects.html [Consulta: 12/11/2021]

[29] Unity Technologies (19/10/2020): *Components.* Disponible en: https://docs.unity3d.com/Packages/com.unity.entities@0.11/manual/ecs_components.html [Consulta: 14/11/2021]

[30] Twin Musicom (2014): *Innovative Audio Production.* Disponible en: http://www.twinmusicom.org/ [Consulta: 11/12/2021]