Universitat de Girona
Escola Politècnica Superior

Grau en Enginyeria Informàtica

PROJECTE FINAL DE GRAU

Desenvolupament d'un videojoc d'acció-aventura

Autor:
David Tempelaar Sanchez

Tutors:
Sergio Gonzalo Besuievsky Glikberg

MEMÒRIA

Convocatòria:
Juny 2023

Departament:
Informàtica, Matemàtica Aplicada i Estadística (IMAE)

Agraïments

Al meu tutor, Gonzalo Besuievsky, per marcar el camí a seguir i donar suport durant tota la creació d'aquest projecte.

A la família, pel seu suport i ànims que m'han acompanyat durant el desenvolupament.

Als amics, per contribuir amb idees i ser part de les proves finals del videojoc.

Índex

Índex de figures

1 Introducció

La popularitat dels videojocs s'ha trobat en constant creixement les últimes dècades, com també ha incrementat el seu preu de venda al públic, amb major impacte sobre els jocs publicats per grans companyies d'aquest àmbit. És incert la viabilitat d'aquest mercat amb l'actual inflació dels preus i existeixen especulacions que neguen la permanència d'aquesta.

A continuació podem veure la creixença d'aquest mercat a nivell mundial:

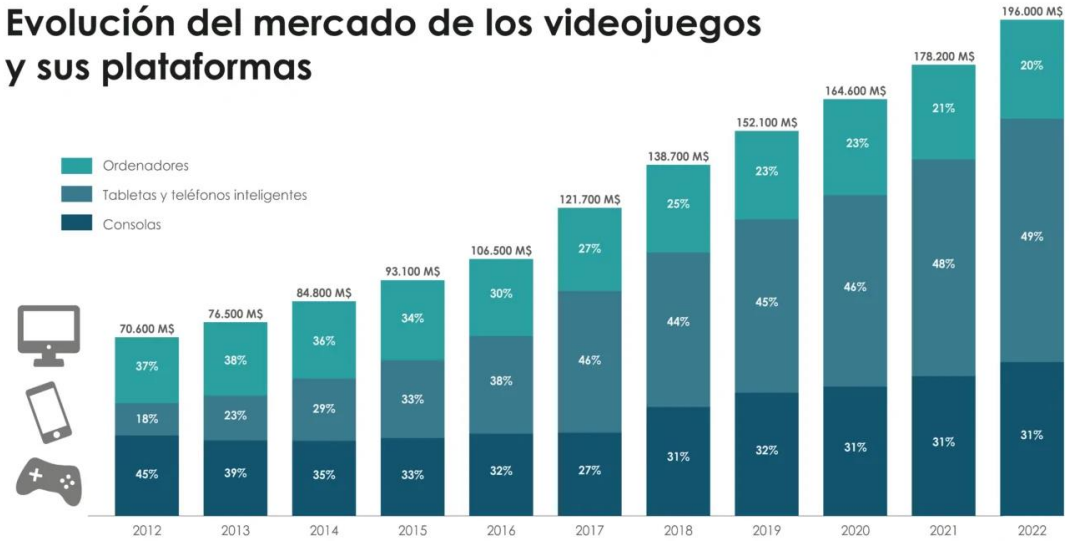

Figura 1: Gràfica ingressos en el sector de videojocs

Els videojocs indies, desenvolupats per una o poques personen, formen part d'aquest sector on han guanyat renom, i sense les facilitats financeres que una gran companyia pot aportar, han existit títols amb un gran èxit. Actualment, el mercat de videojocs indie es troba saturat de jocs, a causa de l'existència de programes de creació i motors gràfics de baix cost i fàcils d'aprendre, que ha fet créixer el nombre de creadors. La distribució online ha incrementat la possibilitat que un joc indie es doni a conèixer en la comunitat de jugadors i ser rentable.

La finalitat d'aquest projecte és crear un videojoc de caràcter acció-aventura que es basarà principalment en l'exploració i completar mapes, formats per puzles. L'escenari en 3D amb estructures amb baixa densitat de polígons juntament amb la perspectiva en tercera persona aportaran una experiència més immersiva.

Són varis els jocs que inclouen mecàniques semblants, alguns amb molt renom en la industria com podrien ser la saga The Legend of Zelda i Inside, entre altres. Encara que el primer és un projecte molt més extens que engloba altres mecàniques, es comentaran algunes idees extretes d'aquests títols en l'apartat 2.6. Estudi de mercat.

1.1 Motivacions

La motivació darrera d'aquest projecte apareix després d'haver treballat amb Blender, un programa de modelatge, il·luminació i renderització, en un projecte durant la carrera i en certs projectes personals. Principalment centrats en modelatge i crear esquelets per animar aquests models.

Donar vida a aquests models és el que impulsa a escollir fer un videojoc per al Treball final de Grau, juntament amb l'experiència que pot aportar treballar amb nou programari com pot ser Unity. Veure les fases que requereix el desenvolupament d'un joc enriqueix el coneixement sobre un sector de la Informàtica que no s'ha tocat en la carrera.

1.2 Propòsits i objectius

El projecte consisteix en un videojoc d'acció-aventura en 3ra persona que constarà de diferents nivells, cada un amb escenari diferent el qual tindrà una mecànica única per a passar al següent. Cada escenari serà diferent i el compondran objectes amb estètica simple i baixa densitat de polígons.

La complicació de cada nivell resideix en completar una mecànica que podria ser un puzle, uns enemics que hem de evitar d'una manera específica, entre altres.

L'objectiu d'aquest treball és assolir els diferents requeriments necessaris per a completar el desenvolupament d'un videojoc que s'han diferenciat entre ells com:

- Triar el programari necessari per al desenvolupament.
- Dissenyar la història que transcorre durant el joc.
- Dissenyar i modelar els personatges.
- Dissenyar i modelar els escenaris.
- Dissenyar i implementar mecàniques i funcionalitats de cada nivell.
- Dissenyar i implementar menús.
- Plantejar millores gràfiques i de rendiment.
- Utilitzar una tècnica de treball adequat per el tipus de projecte.

1.3 Distribució de tasques

L'equip de treball es compondrà només d'una persona per la que la distribució de tasques recauran només en ell. Encara així podem distingir entre diferents grups segons la especialitat que engloba, i ja que cada un tindrà un pes diferent durant el desenvolupament els diferenciarem pel volum de feina que se n'ha invertit.

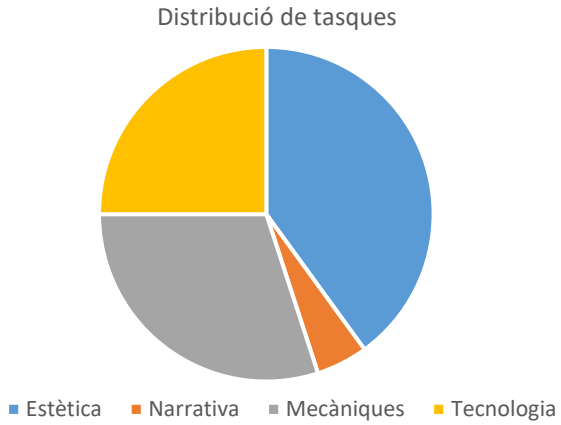

Figura 2: Gràfica de distribució de tasques

L'estètica (40%) inclou tots els escenaris, tant els elements com els personatges, i el nivell artístic dels menús o altres núvols que apareguin durant el joc. La major part serà creat personalment pel que portarà un nivell de feina major que les altres tasques.

La narrativa (5%) serà la que menys s'haurà treballat ja que per la naturalesa del projecte s'ha estimat que no és la faceta que més es vol destacar. Cada nivell i la seva temàtica formaran part de la història global del joc i donaran indicis del perquè s'hi troben certs elements i com aporta més claredat a aquesta.

Les mecàniques (30%) engloba el moviment dels personatges, els puzles a completar i altres funcionalitats que donen al videojoc que formaran part del seu atractiu. Cada escenari tindrà la seva mecànica particular encara que hi ha funcionalitats bàsiques s'introdueixen esglaonadament per a la més fàcil comprensió d'aquesta.

La tecnologia (25%) consistirà en passar aquestes idees dissenyades a ser funcionals fent ús del programari adequat. Al no haver tractar amb programes per aquestes finalitats s'impartirà temps en aprendre a utilitzar-les adequadament i donar consistència al funcionament del videojoc.

2 Estudi de viabilitat

La viabilitat que té un videojoc en el mercat pot variar molt segons el públic al que va dirigit i el cost de producció que suposa el seu desenvolupament. Per poder assumir si aquest projecte té possibilitat de ser rentable estudiarem aquests paràmetres i estimarem si és factible.

2.1 Recursos tècnics

El projecte, un videojoc Indie sense un objectiu massa ambiciós, no requereix d'una tecnologia molt potent, de manera que tant el hardware com el programari utilitzats no han requerit despeses i es trobaven en possessió inicialment.

2.1.1 Hardware

L'equip utilitzat per al desenvolupament del videojoc ha sigut un ordinador de gama mitja amb unes prestacions senzilles comparat amb les existent en el mercat.

Les prestacions esmentades són:

- Processador: AMD Ryzen 5 2600
- Targeta gràfica: Gigabyte Aorus GTX 1060 6GB
- Memòria RAM: HyperX Fury DDR4 16GB

2.1.2 Software

Tot el software utilitzat per la realització del projecte son gratuïts. Alguns tenen alguna condició a l'hora de vendre el producte creat a partir del seu programa però els requeriments d'aquest projecte no compleix aquests requeriments.

Diagrams.net

Diagrams.net és una aplicació web que permet crear diagrames d'una forma intuïtiva i ràpid. Principalment s'ha fet servir per a dibuixar tots els diagrames necessaris durant la documentació.

Figura 3: Logotip Draw.net

Unity

Unity és el motor de joc que es farà servir per donar vida al videojoc. És totalment gratuït amb la condició de que l'ingrés generat pel producte creat amb aquesta plataforma no superi els 100,000$. Si es supera aquest ingrés cal passar-se a Unity Plus, una subscripció que augmenta aquest ingrés límit a 200,000$ i té un cost mensual de 37€.

Està disponible per a les diferents plataformes Windows, Mac i Linux i principalment es codifica en C#. Dissenyat per a la creació de jocs en 2D i 3D amb la possibilitat de compilar-los per a 19 plataformes diferents com podrien ser per a mòbils, ordinadors, consoles i realitat virtual.

Figura 4: Logotip Unity

Existeixen altres motors de joc que podrien haver complert la mateixa funció que Unity, com és el cas de Unreal Engine o Godot. La raó principal de triar Unity és degut a la gran comunitat que existeix darrera aquesta plataforma que aporta molta informació i tutorials per a entendre el seu funcionament. El poc coneixement sobre els motors de joc amb el que es disposava al començament del projecte ha facilitat aquesta decisió.

Blender

Blender és un programa de modelatge, il·luminació, renderització, animació i creació de gràfics 3D. Publicat sota la llicència GPL brinda una total llibertat a utilitzar de manera totalment gratuïta. El software s'ha utilitzat tant per a la modelació com per l'animació dels personatges i elements

Figura 5: Logotip Blender

Trello

Trello és un software d'administració de projectes amb interfície web, iOS i Android. Utilitza el sistema kanban que permet organitzar la feina atomitzant les tasques en targetes que es poden moure deliberadament entre taules. Com que el nostre sistema de organització de tasques és similar al SCRUM, aquest sistema de targetes ha funcionat perfectament.

Figura 6: Logotip Trello

Microsoft Word

Microsoft Word és un software de tractament de textos. Microsoft distribueix el Word juntament amb altres softwares com a paquet anomenat Microsoft Office. La llicencia per la seva utilització no és gratuïta i té un cost anual de 69€ al any. Al ser estudiant de la Universitat de Girona es té accés a la versió Student que surt gratuït de manera que s'ha pogut utilitzar sense necessitat de pagar.

Existeixen altres softwares que compleixen sobradament aquesta funció com podria ser el LibreOffice que és idèntic, o el Overleaf que és un editor de LaTeX i la seva aplicació web és gratuïta.

Figura 7: Logotip Microsoft Word

Visual Studio

Visual Studio és un entorn de desenvolupament integrat amb aplicació disponible per a Windows i macOS. És compatible amb un gran nombre de llenguatges de programació següent un d'ells el C#, el que principalment utilitzarem per a aquest projecte. És l'editor de codi per defecte de Unity i es farà servir amb aquesta finalitat ja que proporciona facilitats a l'hora de cridar llibreries integrades en Unity.

Figura 8: Logotip Visual Studio

2.2 Recursos humans

El desenvolupament del projecte s'ha portat a terme per una sola persona de manera que ha tingut que complir els diferents rols tècnics que es troben normalment en un equip professional de desenvolupament de videojocs.

Aquests equips solen estar formats per:

- Dissenyador
- Programador
- Artista gràfic
- Artista musical

Els creadors de videojocs Indie solen englobar més d'un d'aquests perfils postura que també s'ha agafat en aquest projecte.

2.3 Recursos econòmics

Les despeses calculades en aquest apartat són estimacions de lo que costaria el projecte si no es tingués en possessió els recursos necessaris des del principi i es contractés un equip professional per al desenvolupament del projecte. Els sous es compten en funció del temps pel que es faran càlculs, un amb el temps real dedicat al projecte i un altre amb duració de dos anys, temps aproximat de lo que podria tardar crear-se el joc complet.

Primer deduirem el que valdria l'equip professional:

Professió	Sou anual mitjà	Sou 3 mesos	Sou 2 anys
Dissenyador	23.000 €	5.750 €	46.000 €
Programador	28.000 €	7.000 €	56.000 €
Artista gràfic	25.000 €	6.250 €	50.000 €
Artista musical	28.000 €	7.000 €	56.000 €
SUMA	104.000 €	26.000 €	208.000 €

Figura 9: Taula de sous

Aquest cost no inclou els recursos tècnics que necessitaran per portar a terme les seves feines. L'ordinador amb les especificacions que s'han esmentat en l'apartat **2.1.1 Hardware**, en el moment de compra, el seu cost era de 1200 €. Si per a cada treballador assignem un ordinador, el cost total per a 3 mesos i per als 2 anys serà:

	3 mesos	2 anys
Cost total professionals	26.000 €	208.000 €
Cost recursos tècnics	4.800 €	4.800 €
Cost total	30.000 €	256.000 €

Figura 10: Taula costos total

2.4 Viabilitat legal

Tot el software que es fa servir pel desenvolupament del videojoc és de llicència lliure pel que podem utilitzar tota nostra creació feta a partir d'ella per a la venta al públic. Només es tindrà en compte que els ingressos obtinguts de les ventes no passin dels 100.000 $ per a que no canviïn les condicions de Unity i necessitem una subscripció més cara.

2.5 Mètode de distribució

Per a la distribució del nostre projecte existeixen diferents plataformes que ens poden ser útils. En aquest cas s'ha decantat per itch.io, una pàgina web on penjar un videojoc com a desenvolupador és gratuït i el preu de venta es pot triar a gust, sense cap condició.

Una altre opció més coneguda és la plataforma Steam encara que aquesta sí demana una quantitat per a poder penjar jocs, i en aquest cas es vol gastar el mínim per a la distribució.

2.6 Estudi de mercat

L'estudi dels videojocs en el mercat, del mateix gènere que el projecte que s'està desenvolupant, pot ajudar a tenir una idea més clara a l'hora de planificar quins son els punts forts i quins els punt fluixos d'aquest i poder tractar-los adequadament.

A continuació es comparen els videojocs amb temàtiques i jugabilitat semblants.

2.6.1 Zelda

Figura 11: Imatge del joc The Legend of Zelda

The legend of Zelda és una sèrie de videojocs d'acció-aventura desenvolupat per Nintendo. Tracta de Link, un jove guerrer que s'enfronta a perills i resols puzles per ajudar a la princesa Zelda a derrotar a Ganondorf.

La perspectiva és en tercera persona i els punts més forts són les lluites i els puzles.

2.6.2 Poppy playtime

Figura 12: Imatge del joc Poppy playtime

Poppy playtime és un videojoc de terror i puzles desenvolupat per MOB Entretainment en el 2021. Es juga en primera persona i tracta de sobreviure en una fàbrica de joguines i afrontar diferents enemics que apareixen en els nivells repartits en capítols.

2.6.3 Inside

Figura 13: Imatge del joc Inside

Inside és un videojoc desenvolupat per Playdead en 2016. Tracta d'un nen en un món postapocalíptic, i es basa en resoldre puzles i evitant la mort. La perspectiva és en 2D.

Es pot observar que una de les principals mecàniques són els puzles i funciona amb perspectiva en tercera persona. La temàtica d'horror també combina bé amb el tipus de joc basat en resoldre trencaclosques.

L'originalitat de la història serà el punt diferencial en el nostre projecte.

3 Metodologia

A l'hora de planificar i organitzar el desenvolupament d'un programa és necessari utilitzar un mètode que funcioni per a l'equip de desenvolupament i les tasques a fer siguin clares. Existeixen molts mètodes que han anat apareixent segons les diferents necessitats de cada projecte pel que estaven destinats.

Exemples de metodologies conegudes són:

- Agile
- Scrum
- Waterfall
- Kanban
- Scrumban

La metodologia posada en pràctica és el Scrumban adaptat a les necessitat del projecte ja que al no estar format per un equip de desenvolupadors, algunes característiques del model no seran d'utilitat. La metodologia Scrumban sorgeix de la fusió entre Scrum i Kanban.

Kanban està enfocat en facilitar la visualització de les tasques fent servir elements com podrien ser taules. D'aquesta manera el progrés i els problemes són ràpidament detectables de manera visual.

Scrum, per altre banda, es basa en l'ús de sprints. Els sprints són dates, separades entre elles per un període que com més curt és millor, on l'equip de desenvolupament entrega una funcionalitat del programa totalment funcional. A posteriori es discuteixen les taques per a la següent funcionalitat que es vol implementar i es crea així el backlog per al següent sprint.

Les característiques del Scrum que farem servir seran:

- Atomitzar les tasques i estimar el temps de realització.
- Convocar sprints cada dos setmanes amb el tutor.
- Crear el backlog amb les tasques per al següent sprint.

Les característiques que adoptaren de kanban són:

- Utilització de taules per a la fàcil visualització de les tasques que formen el backlog del sprint.

L'ordre d'execució del sistema de treball que hem escollit és:

- De les tasques pendents a fer del projecte es trien els més importants per al sprint següent, tenint en compte el temps d'elaboració de cada una per a que sigui coherent amb temps que s'hi pot dedicar.

- Durant el període entre sprints s'implementen les tasques.

- Al sprint, es revisen les tasques i el seu funcionament correcte. Si existeixen errors o alguna tasca no ha tingut temps de fer-se, passen a formar part del següent backlog com a nova tasca amb el seu temps de realització. La resta de tasques s'escolliran del conjunt de tasques per fer del projecte de la mateixa manera que s'ha fet en el primer pas.

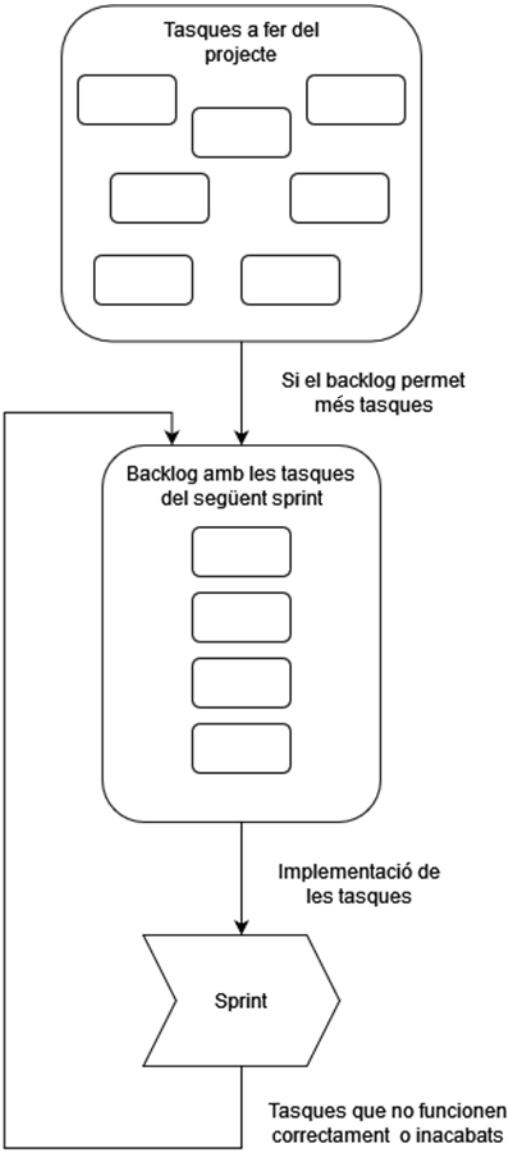

Figura 14: Diagrama de flux de treball

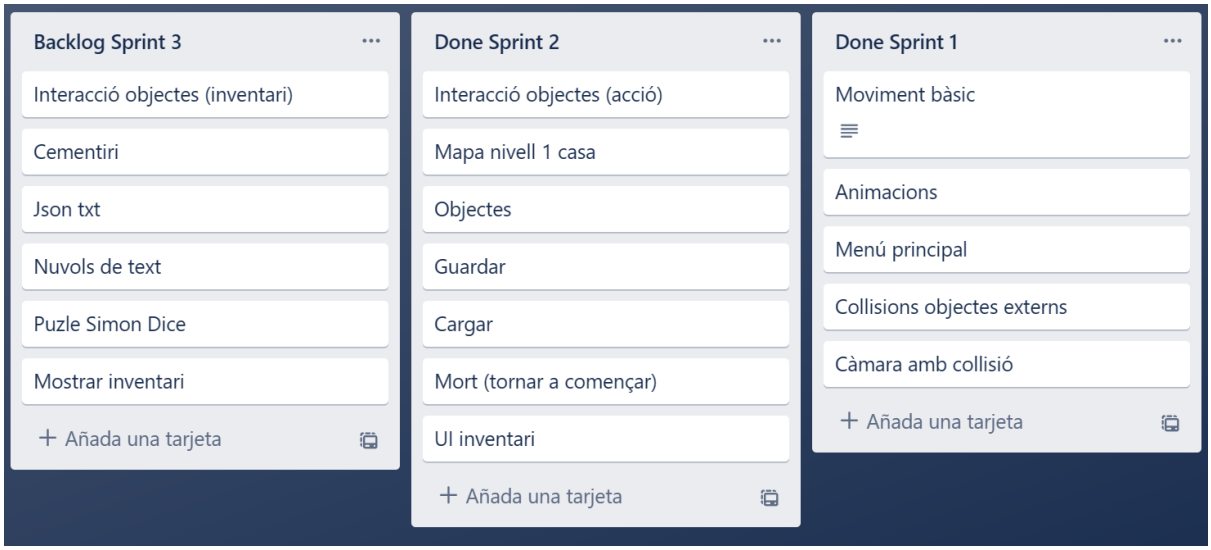

Figura 15: Exemple de tasques en taules estil Kanbas

4 Planificació

La planificació juga un paper important en el desenvolupament d'un projecte. La combinació d'una metodologia adequada i una bona planificació agilitza el desenvolupament i facilita els canvis que poden sorgir a mesura que es va avançant a l'objectiu.

El projecte es fracciona en quatre grans grups que posteriorment es dividiran en tasques més específiques per a la utilització en el mètode Scrum. A les tasques els hi és assignat un temps estimat d'elaboració que facilitarà la planificació de la seqüència en la que s'executaran.

Els grups i les seves tasques pertinents que s'acabaren creant són:

- Estètica
    - **Modelar i animar el personatge principal**: Modelar i animar el personatge principal amb Blender. Aquest serà el personatge que controlarà el jugador.

      Duració estimada: 1/2 mes.
    - **Dissenyar l'escenari del primer nivell**: Crear el primer mapa i els objectes que en formen part amb Blender. Els models, un cop importats a Unity, formen un sol objecte pel que els elements que necessiten modificacions posteriors es creen com a objectes separats.

      Duració estimada: 1/2 mes.
    - **Dissenyar l'escenari del segon nivell**: Crear el mapa i els objectes amb Blender.

      Duració estimada: 1/2 mes.
    - **Dissenyar l'escenari del tercer nivell:** Crear el mapa i els objectes amb Blender.

      Duració estimada: 1/2 mes.
    - **Modelar i animar els enemics del tercer nivell:** Modelar i animar l'enemic del tercer nivell.

      Duració estimada: 1/2 mes.
    - **Dissenyar les interfícies:** Dissenyar els menús, les vinyets de text i la interfície de l'inventari.

      Duració estimada: 1/2 mes.

- Narrativa

  - **Crear la història del joc:** La història tindrà pes sobre la temàtica de cada nivell donant importància a que sigui una història ben consolidada. Possibles canvis posteriors poden deixar incoherents detalls ja implementats.

    Duració estimada: 1/4 mes.

  - **Dissenyar els nivells de manera que tinguin relació amb la història principal:** La història principal es va explicant a través de les ambientació dels nivells i els events que hi tenen lloc, encara que aquests últims no necessàriament tenen relació amb la història.

    Duració estimada: 1/4 mes.

- Mecàniques

  - **Dissenyar el moviment del personatge principal:** Determinar les variables i funcions necessàries per a que el personatge principal tingui moviment controlat per el jugador, gravetat i col·lisions. Aquest serà el personatge que controlarà el jugador.

    Duració estimada: 2 mesos.

  - **Dissenyar el moviment dels enemics:** Determinar la IA que dona vida als enemics com també les interaccions que aquest tindrà amb el personatge principal i el seu entorn.

    Duració estimada: 1/2 mes.

  - **Dissenyar els objectes interaccionables:** Dissenyar de quina manera els objectes interactuen amb el jugador i altres elements. Diferenciem tres tipus d'elements interaccionables: el que s'activa per proximitat, el que s'activa amb una acció i un objecte que el jugador pot recollir.

    Duració estimada: 1 mes.

  - **Dissenyar les mecàniques del segon nivell:** Determinar la mecànica del segon nivell que és un mini joc que es basa en el joc popular "Simon dice". Consisteix en crear els botons amb colors que creen una seqüència que el jugador ha de repetir.

    Duració estimada: 1 mes.

- **Dissenyar les mecàniques del tercer nivell:** Determinar la mecànica del tercer nivell que tracta de sortir d'un laberint amb un temps màxim. Consisteix en enemics que persegueixen al jugador mentre aquest no els il·lumina amb la llanterna, i per altre banda el jugador ha d'agafar recàrregues per a que no s'acabi la bateria de la llanterna, indicat per un temporitzador.

  Duració estimada: 1 mes.

- Tecnologia

  - **Implementar el moviment i col·lisió de la càmera del personatge principal:** Implementar el comportament de la càmera 3D que segueix al jugador, tant el moviment com les col·lisions contra els elements de l'escenari per a no travessar-los.

    Duració estimada: 1/2 mes.

  - **Implementar les mecàniques al motor gràfic**: Implementar totes les mecàniques dissenyades en el nostre videojoc amb el motor gràfic Unity.

    Duració estimada: 3 mesos.

Altres tasques necessaris per acabar de tancar el cicle de desenvolupament d'un videojoc són testejar i corregir els errors que puguin sorgir al moment implementar codi nou, i l'exportació del videojoc per la sortida al públic. La metodologia Scrum ens situa la majoria testeigos propers al dia en el que es fa el sprint, les proves restants es portaran a terme sempre que s'implementin noves mecàniques.

- Testeig i correcció d'errors: Els testeigos i la correcció d'errors van apareixent a mesura que s'implementen funcionalitats noves.

  Duració estimada: 3 mesos.

- Exportació del joc: Exportar el videojoc i testejar si existeixen errors a l'hora de jugar-hi.

  Duració estimada: 1/4 mes.

4.1 Diagrama de Gantt

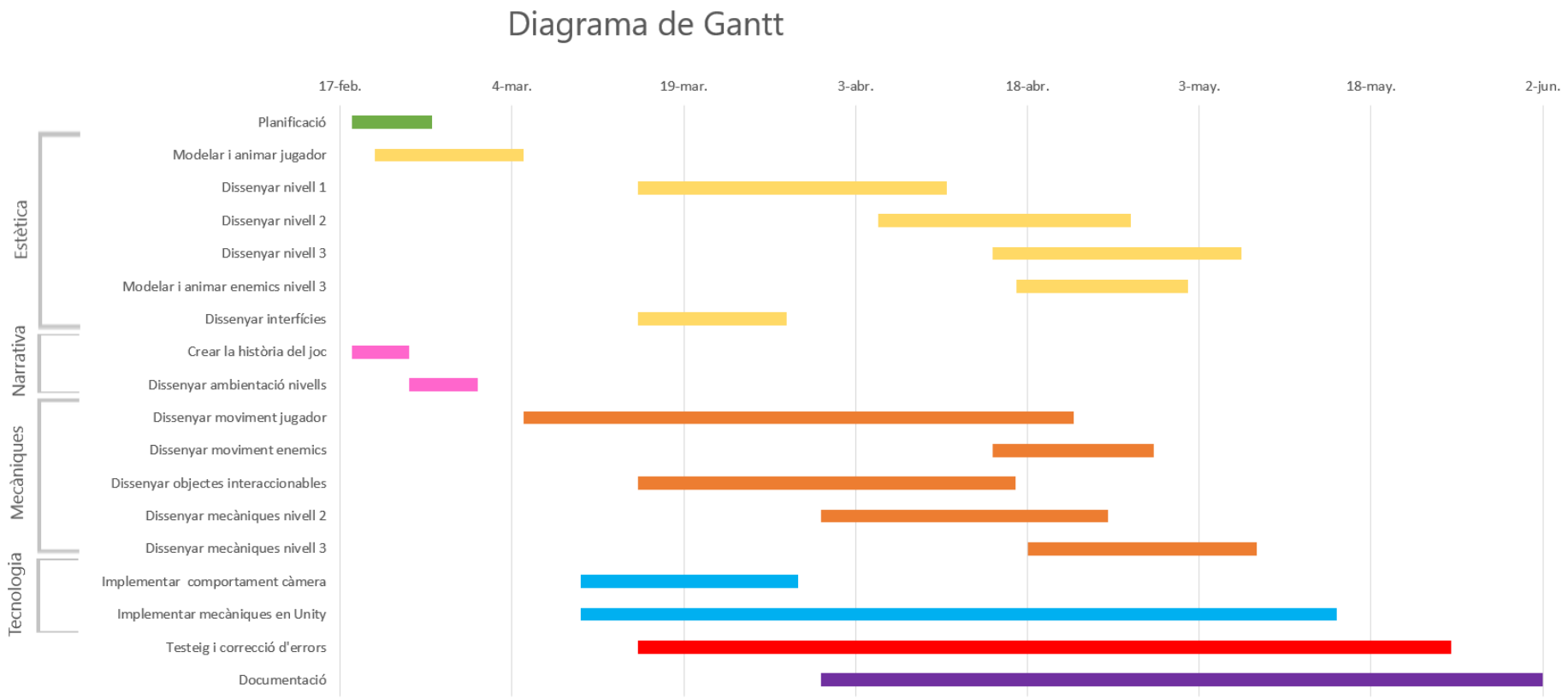

Figura 16:Diagrama de Gantt

5 Marc de treball i conceptes previs

En aquest capítol es descriuen els conceptes i nomenclatures específiques del projecte fonamentals pel seguiment de la memòria.

5.1 Conceptes de Unity

- **Mesh**: La paraula mesh significa cos amb filaments entrecreuats. Blender i Unity utilitzen aquesta nomenclatura per a referir-se a qualsevol cos amb volum composat per un conjunt de vèrtex. Tots els objectes que s'implementen en aquest projecte son del tipus mesh. Els dos programes suporten perfectament la importació i exportació d'objectes amb aquestes característiques.

- **Scenes**: Una "scene", que anomenarem escena en futures referències, és on es treballa amb contingut en Unity. Un projecte pot contenir una o més escenes i serveixen per separar continguts entre elles per evitar carregar-los amb molts elements o per temes de optimització. Escenes varies poden estar activades a la vegada i els elements de les dos estaran carregades en el espai de treball. Unity dona eines per facilitar la seva gestió.

- **GameObject**: El GameObject és un element de Unity que funciona com una càpsula a la qual se li poden afegir un gran ventall d'elements varis com podria ser un mesh, una càmera, element d'àudio entre altres. Ens brinda la possibilitat de crear un objecte com podria ser el jugador, que estarà composat per GameObject amb diferents elements adjunts com podria ser la càmera, un controlador de moviment, un detector de col·lisions.

- **Prefabs**: és una abreviatura de la paraula "prefabricated" de l'anglés. En Unity es refereix als objectes que poden contenir dates de tota mena i que el programa ens facilita poder exportar i importar a diferents escenes,

- **Assets**: Els assets són tots els elements que es tenen a mà facilitant la ràpida integració en el projecte si escau. Abarca qualsevol tipus d'elements. En aquest projecte fem servir sobretot objectes prefabs d'altre creadors, imatges, tipus de materials per a aplicar en els meshs en Unity i els nostres propis prefabs.

- **UI (User Interface):** La interfície d'usuari és l'encarregat de rebre les interaccions humanes i traduir-lo de manera que el programa sàpiga com actuar. En el nostre projecte fa referència principalment a les entrades de teclat i ratolí, les interfícies de menús i l'inventari amb el que es pot interaccionar.

- **Collider**: Objecte de Unity que delimita una àrea que al entrar en contacte amb un altre collider l'identifica com una col·lisió.

- **Trigger**: Significa disparador i en Unity normalment es refereix a la característica que poden habilitar els collider, el qual crida un event en els dos objectes implicats si xoquen. Aquesta utilitat apareix sovint en aquest projecte.

- **Canvas**: És una característica de Unity que facilita la creació de interfícies gràfiques.

- **SimonDice**: Aquesta nomenclatura és propi del projecte. El nivell dos del videojoc consisteix en completar el popular joc "Simon dice", que constava de quatre botons de colors verd, groc, blau i vermell per separat i aquestes s'il·luminaven amb un ordre que el jugador havia repetir amb el mateix ordre.

6 Requisits del sistema

Per reconèixer els requisits de la nostra aplicació és necessari estudiar els actor que interactuen amb el nostre videojoc. D'aquesta manera podrem identificar els requisits que el nostre programa necessita per a que l'actor tingui una bona experiència al utilitzar-lo. Existeixen altres requisits que no estan lligats a l'actor de manera que separarem els tipus de requisits per funcionals, que inclouen a l'actor i els no funcionals, que fan referència als requisits que necessita el joc per a que funcioni correctament.

6.1 Requisits funcionals

Els requeriments funcionals del nostre sistema:

- El jugador ha de poder triar si començar una nova partida, seguir amb la sessió anterior o jugar un nivell anterior ja completat.
- El jugador ha de poder posar en pausa el joc en tot moment.
- El jugador ha de poder sortir del joc en tot moment.
- El jugador haurà de completar la mecànica del nivell per completar-lo.
- El jugador serà introduït progressivament a les mecàniques a mesura que avança el videojoc.

6.2 Requisits no funcionals

Els requisits no funcionals d'aquest projecte:

- El videojoc es desenvoluparà completament amb Windows 10 per evitar conflictes de compatibilitat.
- El videojoc s'actualitzarà periòdicament fins acabar tots els nivells necessaris per a l'explicació de la història.
- El videojoc necessita un teclat i un ratolí per poder captar les instruccions del jugador.
- S'ha estipulat que els requisits del sistema on es vol executar el joc hauran de tenir les prestacions mínimes que demana Unity per a funcionar.

Requisits mínims	Requisits recomanats
15 Gb d'espai lliure en disc 4 Gb de memòria RAM Targeta gràfica amb DX9 o DX11	15 Gb d'espai lliure en disc 4 Gb de memòria RAM Targeta gràfica amb DX9 o DX11 Targeta gràfica Nvidia o ATI amb 1Gb VRAM dedicada o superior

Figura 17: Taula de requisits mínims i requisits recomanats

7 Disseny del videojoc

Són molts els factors que poden impulsar a un videojoc a tenir èxit, alguns fora de l'abast dels desenvolupadors com pot ser la sort, el panorama del mercat i l'acceptació del gènere del que tracta; com d'altres on els desenvolupadors sí tenen poder per a treure el màxim profit com és la jugabilitat, la originalitat, la qualitat i si és entretingut.

En aquest apart parlarem del disseny pel que s'ha adoptat per a complir amb el nivell suficient per tenir possibilitats en el mercat.

7.1 Objectiu del videojoc

El joc desenvolupat en aquest projecte consta de diferents nivells amb les seves respectives mecàniques i puzles que el jugador ha d'anar completant. Cada nivell és un escenari diferent i juntament amb els elements que s'hi troba es va explicant la història principal progressivament.

Algunes mecàniques que van sortint poden aparèixer també en nivells posterior com és el cas dels objectes portables que apareix en el segon escenari i es trobarà en tots els nivells posteriors. Aquest mètode ajuda al fet que l'aprenentatge de mecàniques més difícils es poden introduir progressivament i evitar saturar el jugador.

En l'estat actual del videojoc no existeix un final consolidat pel qual està obert a nous escenaris amb mecàniques noves.

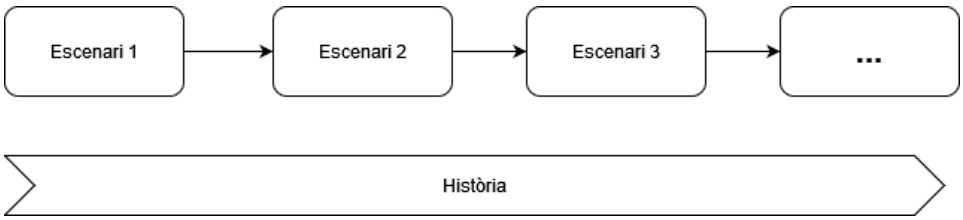

Figura 18: Desenvolupament progressiu de la història

7.2 Jugador

El jugador serà l'agent que controlarà el personatge principal i interactuarà amb les interfícies que formen part del videojoc. Les accions permeses a l'usuari són les següents:

Moviment: El personatge es pot moure endavant, endarrere, a la dreta i a l'esquerra. Aquesta acció és controlada pel jugador a través del teclat, amb les tecles W, S, A i D.

Càmera: La càmera en tercera persona té una rotació circular completa al personatge. El jugador podrà moure la càmera dins d'aquest rang a través del ratolí.

Interaccionar amb objectes: Segons el tipus d'objecte i la seva finalitat dins del joc, la interacció pot variar però tots segueixen una seqüencia semblant a l'hora de mostrar si s'hi pot interactuar i com s'hi interactua.

Proximitat: Tots els objectes amb els que el jugador pot interactuar es ressalta amb un color més vistós que el color per defecte que té l'objecte, si el jugador està en el rang adequat. El rang varia segons la finalitat de l'element però sol ser proper.

Figura 19: Objecte sense activar per proximitat

Figura 20: Objecte activat per proximitat

Acció: Segons el tipus d'objecte aquest punt pot ser diferent. Els objectes que funcionen com un botó, en el moment d'entrar en el rang de detecció aquest ja s'activa i fa l'acció que li pertoca. En canvi si és un objecte que l'acció la ha de controlar el jugador, com els objectes que es volen agafar o palanques, s'activarà solament si el jugador utilitza la tecla d'interacció assignada.

Objecte d'inventari: L'objecte d'inventari es diferència de els elements de l'escenari fixes ja que al interactuar amb ell desapareix visualment i quedarà registrat en l'inventari del jugador.

Gestionar objectes de l'inventari: Els objectes bàsics només es mostren en l'inventari quan el jugador els vols visualitzar, no hi pot interaccionar el jugador. En canvi es poden trobar objectes equipables, amb funcionalitats diferents que el jugador pot aprofitar al utilitzar-los. El numero d'objectes equipats a la vegada pot variar a mesura que avança el joc i el jugador controla quins són els que es vol equipar o desequipar.

Figura 22: Personatge sense cap objecte

Figura 21: Personatge amb una llanterna equipada

Menús: Els menús són les interfícies on el jugador pot decidir sobre qüestions del videojoc que no formen part de la jugabilitat com podria ser començar un a nova partida, sortir del joc i carregar un nivell específic. Al iniciar el joc apareix el menú inicial al que només es podrà tornar a accedir al començar un nou joc. El menú de pausa apareix al pausar el joc, controlat pel jugador amb la tecla assignada per pausar. S'explicaran els menús més a fons en l'apartat 7.8 Interfícies.

7.2.1 Casos d'ús

Per visualitzar les funcionalitats del videojoc que controla el jugador s'han creat els diagrames de cassos d'ús per a l'inici de partida mostrat en la Figura 19, com de pausa en la Figura 20.

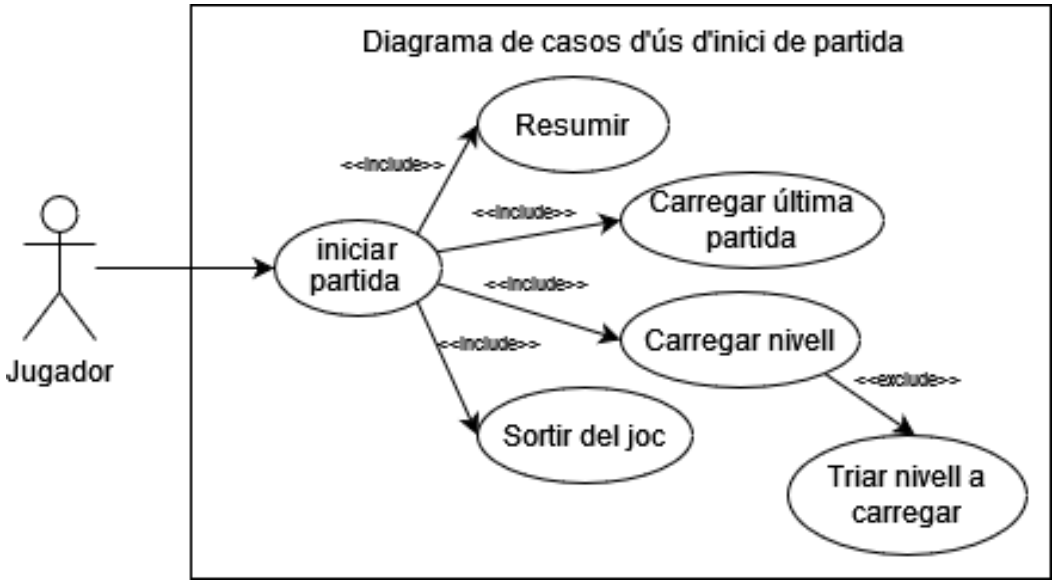

Figura 23: Diagrama de casos d'ús d'inici de partida

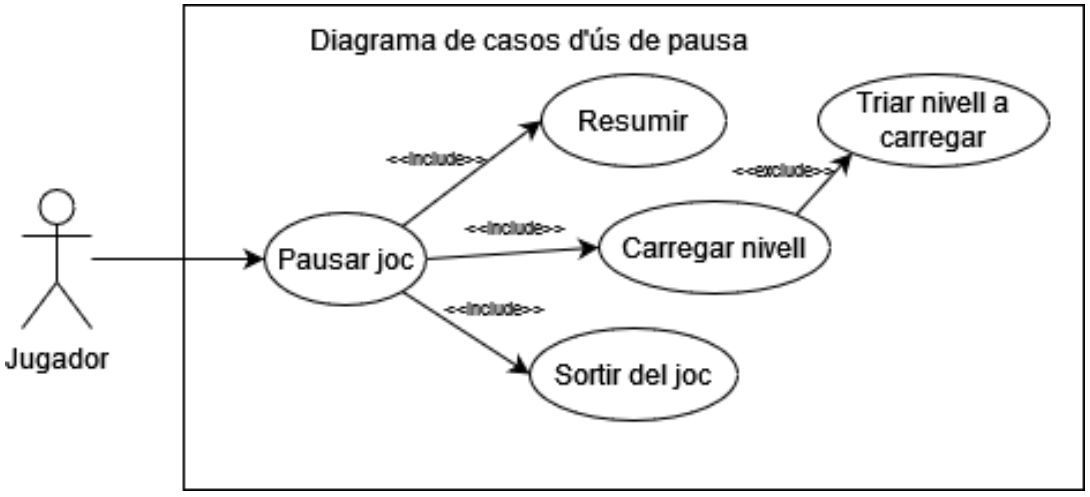

Figura 24: Diagrama de casos d'ús de pausa

7.2.2 Diagrama d'activitats

S'ha creat un diagrama d'activitats, Figura 21, per veure el flux que segueix el joc a partir de que s'inicia i va avançant a mesura que es juga.

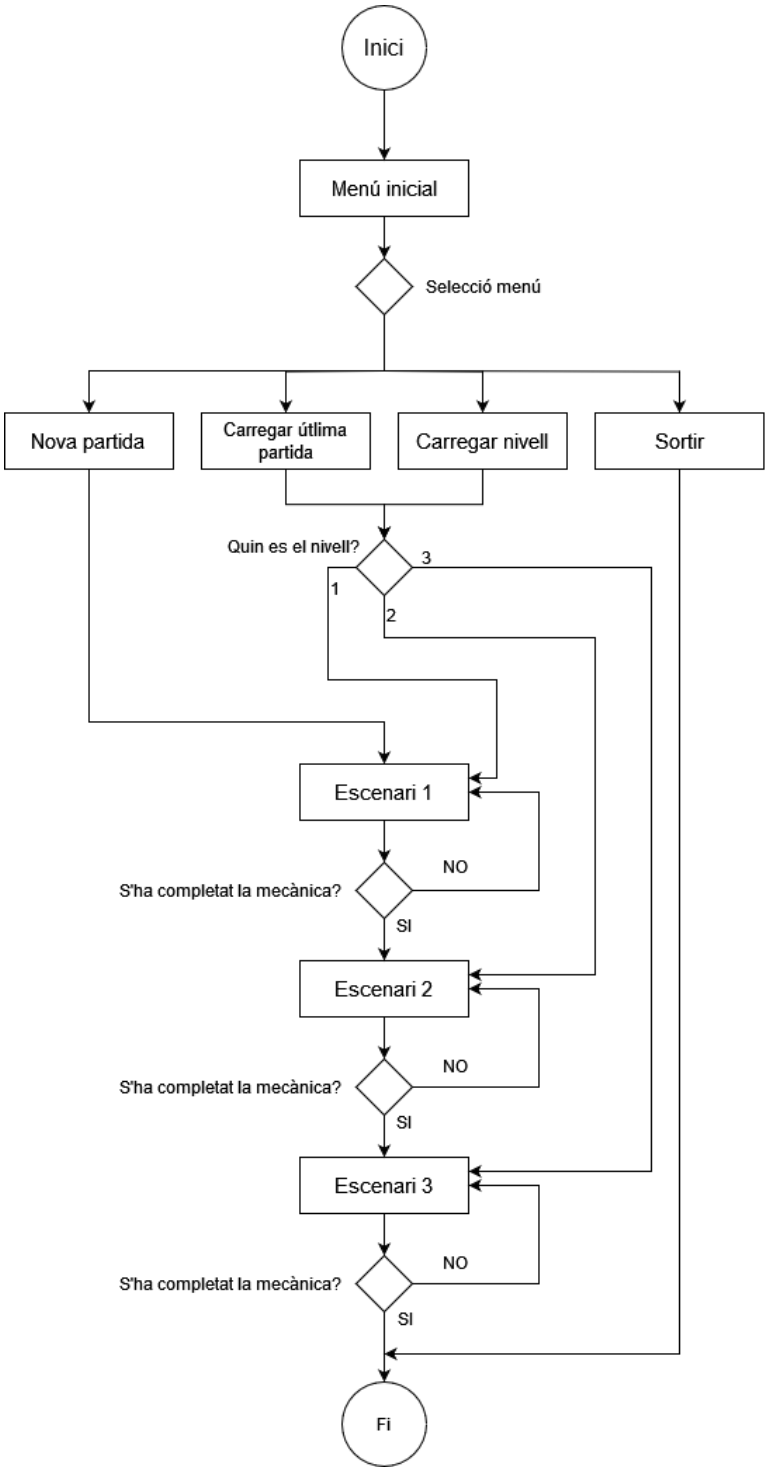

Figura 25: Diagrama d'activitats del joc total

7.3 Narrativa

La trama del videojoc s'explica al jugador per diferents vies com els textos que apareixen quan el personatge agafa un objecte, pensaments que el personatge a vegades mostra i per la temàtica de l'escenari. El primer nivell introdueix al personatge principal i explica la raó per la que s'embarca en aquesta aventura i l'objectiu que té. A mesura que es completen els diferents nivells van introduint-se nous personatges, enemics principalment, que formaran part de les mecàniques dels nivells.

Argument

La història tracta de una noia que es desperta a mitja nit a causa d'un soroll que sent mentre dorm. Veu que el seu gos no es troba dormint a la habitació on normalment sol estar pel que es lleva per veure si està bé. Al sortir de l'habitació troba indicis de que han entrat a casa i el gos no es troba enlloc. Surt al carrer per veure si troba més informació sobre els culpables i la ubicació de la seva estimada mascota. A mesura que avança es troba en situacions tenebroses i criatures malignes que no són del seu món com zombies i fantasmes. Tot això es deu a que els que han entrat a casa seva i han robat el gos són integrants d'una secta satànica que volen sacrificar al pobre animal per invocar un esperit maligne i com que no és la primera vegada que tracten amb esperits malignes cada cop que el personatge principal s'apropa on s'amaguen els perpetuadors veu més coses paranormal.

7.4 Nivells

7.4.1 Primer escenari

Mapa

L'escenari del primer nivell és la casa del personatge principal. La finalitat d'aquest mapa és la d'introduir el personatge al jugador i explicar la situació en la que es troba a través de objectes que es troba en la zona. Petjades de sabates en el terra, una gerra caiguda, la foto del gos seran els encarregats de introduir el jugador al món creat en la història.

Figura 26: Escenari 1

Figura 27: Escenari 1 vist des de dalt

Mecàniques

La mecànica principal d'aquest nivell és interactuar amb tots els objectes possibles per a poder sortir de la casa. S'assegura així que el jugador ha pogut veure tota la introducció a la història al haver llegit tota la informació que proporcionen aquests elements interactius.

Per passar al següent nivell el jugador s'haurà d'apropar a la porta i el joc el transferirà automàticament al segon mapa.

7.4.2 Segon escenari

Mapa

El segon escenari transcorre fora de la casa de la protagonista a prop d'un cementiri on es trobarà el mini joc de SimonDice. Les zones no accessibles estan delimitades per una paret invisible per evitar que el jugador pugui sortir del mapa i dirigir-lo a la posició on es troba la mecànica. El mini joc s'ha ambientat en un cementiri on cada element és una tomba.

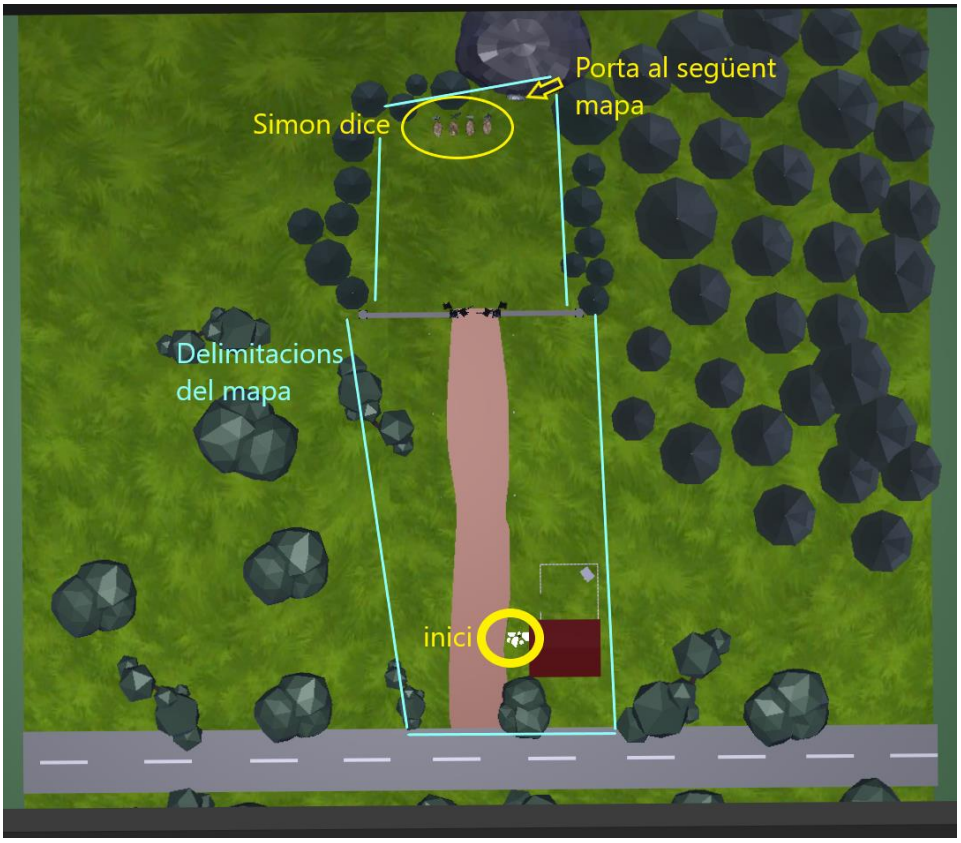

Figura 28: Escenari 2 vist des de dalt

Figura 29: Escenari 2 amb il·luminació

Mecàniques

El segon escenari es completa si s'aconsegueix completar el mini joc SimonDice. La dificultat pot incrementar-se o disminuir per codi encara que el joc no té implementat nivells de dificultat globals. Per l'ambientació s'han dissenyat els botons del SimonDice com tombes amb una flor del color que pertoca en cada un.

Figura 30: SimonDice amb la pedra que ploqueja el camí

Quan el jugador s'apropa a les tombes s'activa el començament del joc on es prohibeix el moviment al jugador, la càmera se situa de manera que totes les tombes siguin visible i es mostra la seqüencia de colors amb unes mans zombie que surten de la tomba. Un cop s'ha vist tota la seqüència la càmera torna a la posició normal i el jugador pot moure's novament. Per completar el joc necessitarà trepitjar les tombes per l'ordre mostrat anteriorment. Si falla, automàticament es torna al pas anterior de mostrar la seqüència. Si encerta, la pedra que bloqueja el pas al següent nivell es mourà per a deixar pas al jugador.

Figura 31: La mà zombie apareix per mostrar la seqüència

Figura 32: Jugador activant la tomba imitant la seqüència

7.4.3 Tercer escenari

Mapa

El tercer i últim escenari transcorre en una cova fosca i sense llum natural. Els passadissos que el travessen creen un laberint amb una sortida final, però on també trobem camins sense sortida o camins circulars que tornen a sortir allà mateix on han començat. S'hi troben també uns éssers paranormals que persegueixen al jugador.

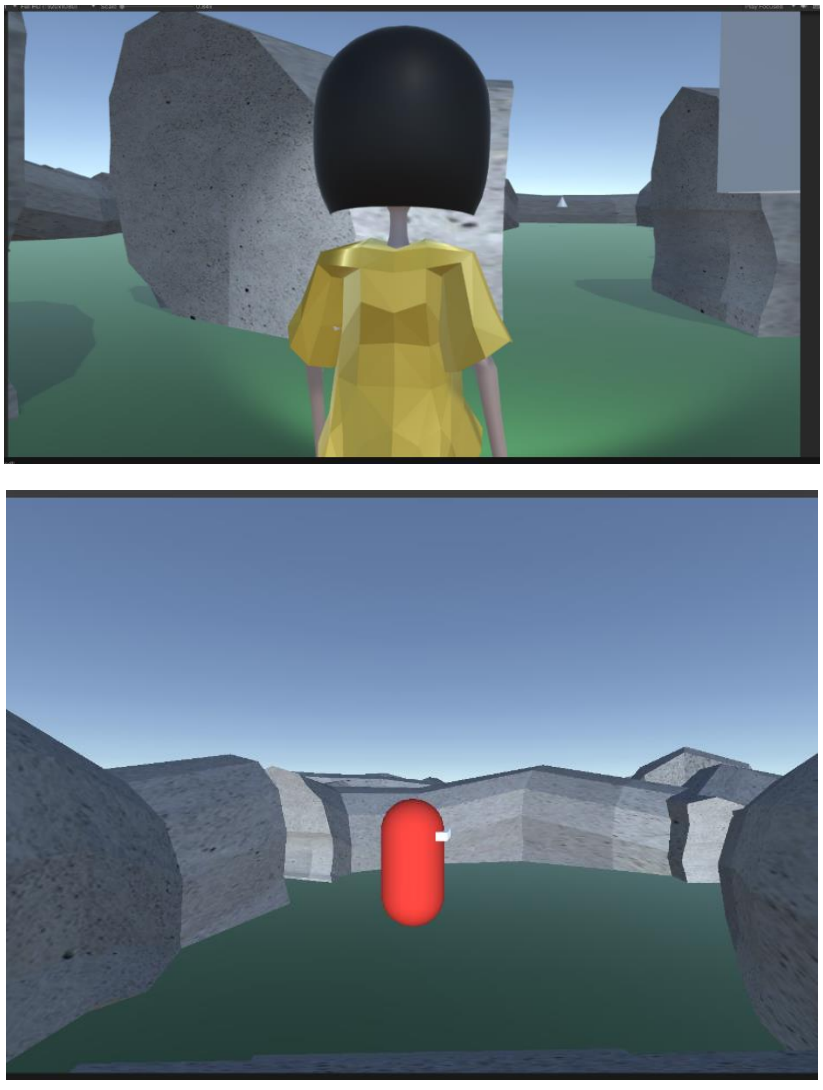

Figura 33: Vistes diferents de l'interior de la cova

Mecàniques

La finalitat d'aquest nivell és sortir evitant que al personatge se li apagui la llanterna que porta equipat i que els monstres no l'atrapin.

El temps de llum restant que li queda a la llanterna es mostra amb un temporitzador. Si aquet temporitzador arriba a zero la llanterna s'apaga, el personatge es transporta a l'inici de la cova i es torna a començar de nou. Per a evitar que el temporitzador acabi, es troben en el laberint, unes bateries que al tocar-les sumen més temps al temporitzador.

Els enemics persegueixen al personatge sempre i quan no estiguin il·luminats per la llum de la llanterna, que si és cas es quedaran quiets fins que la llum ja no els enfoqui. Si l'enemic entra en contacte amb el personatge, aquest torna a l'inici de la cova i haurà de tornar a començar.

Figura 34: Escenari 3 vist des de dalt

7.5 Enemics

Els enemics són actors que dificulten al jugador passar al següent nivell, amb una mecànica que pot ser única per un mapa en concret o comparteixen la mateixa amb enemics d'altres nivells.

En l'estat actual del projecte només tenim un tipus d'enemic que es troba en el tercer nivell. Formen un grup d'enemics que es comporten de manera idèntica i tenen la finalitat d'atrapar el protagonista. En el moment que el jugador es atrapat, aquests tornarà a començar des del principi del nivell.

7.6 Interfícies

7.6.1 Menú inicial

El menú principal només apareix quan s'inicia el joc. Aquest menú donarà la opció de:

- **Començar partida nova**: Borra les dades de la partida anterior i comença des de zero. El jugador apareix en el nivell 1.
- **Carregar última partida**: El jugador apareix en el nivell més alt assolit de la última partida.
- **Carregar nivell**: Dona la possibilitat de triar el nivell al que es vol anar si aquest ja ha estat completat anteriorment en la partida actual.
- **Sortir del joc**: Tanca el joc.

Figura 35: Menú inicial

7.6.2 Menú de pausa

El menú de pausa es pot accedir en qualsevol moment de la partida si el jugador activa la tecla assignada per pausar el joc. Mostra dos opcions:

- **Resumir**: Amaga el menú de pausa per seguir jugant.
- **Sortir**: Tanca el joc.

El menú està dissenyat per a que en un futur mostri també l'opció de triar nivell .

Figura 36: Menú de pausa

7.6.3 Pantalla final

La pantalla final surt una vegada el jugador ha completat l'últim nivell que a l'altura que es troba el projecte és el tercer nivell. Mostra un missatge d'ànims i dona l'opció de tancar el joc.

Figura 37: Pantalla final

8 Implementació i proves

8.1 Personatge

El model del personatge ha estat modelat amb Blender utilitzant com a base un model 3D de noia trobat online. Posteriorment, s'ha creat un esquelet per a poder animar els moviments de caminar, córrer i ajupir-se.

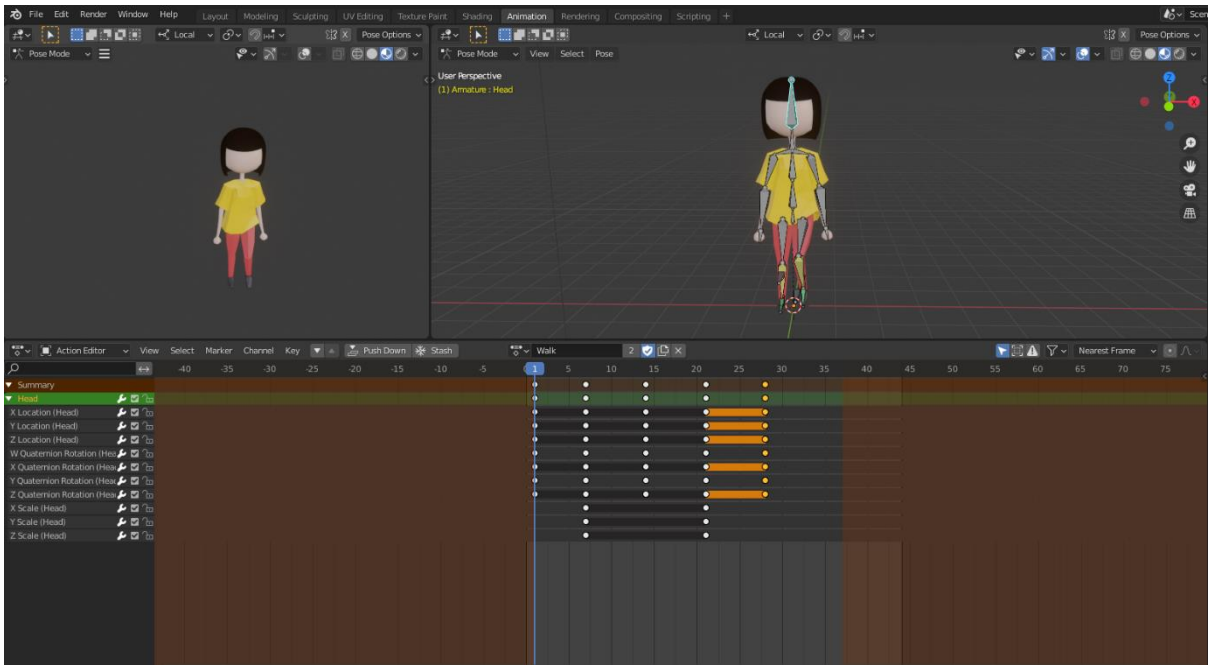

Figura 38: Modelatge de la protagonista en Blender

8.2 Nivells

Els nivells estan formats per diferents escenes de Unity de manera que al canviar de nivell és necessari carregar el nou escenari i descarregar l'anterior. La complicació de fer servir escenes separades és que els elements com el Jugador i els menús no es comparteixen per les que han d'existir en cada escena com a objectes diferents.

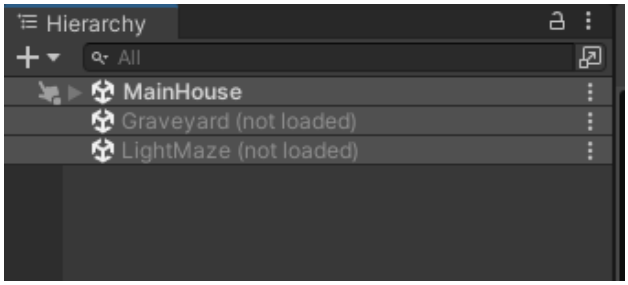

Figura 39: Primera, segona i tercera escena del videojoc

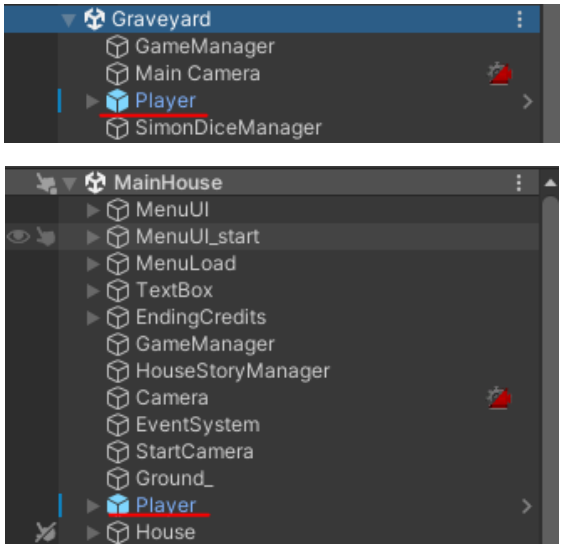

Figura 40:Exemple d'elements repetits en diferents escenes

Els escenaris s'han modelat en Blender i importat a Unity com objecte.fbx. Formarà un sol cos pel que tots els elements que tenen moviment o funcions pròpies, alienes al mapa es modelen per separat.

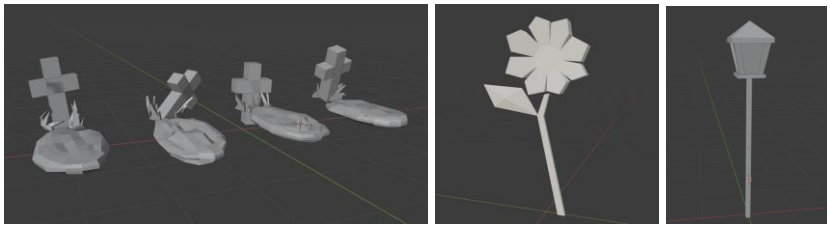

Figura 41: Exemple d'objectes modelats per separat de l'escenari

8.2.1 Game_Manager

La gestió de les escenes es fa a través del script *Game_Manager* que s'encarrega de les funcions relacionades amb la sessió de joc:

- **Gestionar la càrrega d'escena**: Carrega la nova escena i treu la que anteriorment estava carregada.

```
public void LoadScene(int i)
{
    UnityEngine.SceneManagement.SceneManager.LoadScene(i, UnityEngine.SceneManagement.LoadSceneMode.Single);
}
```

Figura 42: Funció que gestiona la càrrega i descàrrega dels escenaris

- **Inicialitzar la sessió de joc**:
  - Carrega el primer escenari de fons i mostra el menú d'inici.
  - Comprova si existeix una sessió anterior de joc. Si no existeix només deixarà començar partida nova. Si ha trobat una sessió antiga, carregarà la informació del jugador i donarà l'opció de continuar amb aquella sessió.

```
void Start()
{
    endingCredits.enabled = false;
    textBox.enabled = false;
    player = GameObject.Find("Player");
    plymv = player.GetComponent<PlayerMovement>();
    if (!player.GetComponent<PlayerManager>().ExistData()) { player.GetComponent<PlayerManager>().SaveToJson(); }
    player.GetComponent<PlayerManager>().LoadFromJson();
    Debug.Log("firstgame: " + player.GetComponent<PlayerManager>().playerData.FirstGame());
    if (player.GetComponent<PlayerManager>().playerData.FirstGame()) { Initial_Game(); }
    else Resume_Game();
}
```

Figura 43: Funció encarregada d'iniciar la sessió de joc

- **Gestionar el pausat**:
  - Mostra el menú de pausat.

- **Gestionar sortir del joc**:

```
public void Quit_Game() //Quit game menu button
{
    player.GetComponent<PlayerManager>().playerData.SetFirstGame(true);
    player.GetComponent<PlayerManager>().SaveToJson();
    Application.Quit();

}
```

Figura 44: Funció que tanca l'execució del joc

- **Gestionar la càmera**: Gestiona la càmera que s'està mostrant al jugador utilitzant el paquet *cinemachine* de Unity, al que s'entrarà en més detall en l'apartat 8.3 Càmera.

```
public void ChangeCamera(int cam)
{
    switch (cam)
    {
        case 0:
            gameCam.Priority = 1;
            startCam.Priority = 0;
            aux1Cam.Priority = 0;
            break;
        case 1:
            gameCam.Priority = 0;
            startCam.Priority = 1;
            aux1Cam.Priority = 0;
            break;
        case 2:
            gameCam.Priority = 0;
            startCam.Priority = 0;
            aux1Cam.Priority = 1;
            break;
    }
}
```

Figura 45: Funció que gestiona les càmeres actives

- **Gestionar passar de nivell**: En passar de mapa es carrega l'escenari següent que normalment serà un nivell més que el nivell màxim assolit, però en poder tornar a repetir un mapa ja completat és necessari tenir un mètode per a guardar tant el nivell màxim assolit com també el nivell que s'està jugant. Per tant, quan es passa al següent escenari s'ha de comprovar si és el màxim o s'està repetint un nivell antic.

```
public void finisMap()
{
    PlayerData playerData = player.GetComponent<PlayerManager>().playerData;
    Debug.Log("GameManager-> Map " + playerData.getCurrentMap() +" finsihed!!!");
    //Sum maps done on player mapscompleted list
    int cm = playerData.getCurrentMap();
    int comp = playerData.getMapsCompleted();
    if (cm + 1 < mapsAvaliable) { playerData.setCurrentMap(cm+1); }
    if (comp + 1 < mapsAvaliable && cm + 1 > comp) { playerData.setMapsCompleted(comp + 1); }
    player.GetComponent<PlayerManager>().SaveToJson();

    if (playerData.getCurrentMap()>cm) { LoadScene(playerData.getCurrentMap()); }
    else //Game Finished
    {
        Debug.Log("GAME FINISHED!!!");
        GameObject.Find("Player").GetComponent<PlayerMovement>().changeMovability(false);
        GameObject.Find("EndingCredits").GetComponent<Canvas>().enabled = true;
    }
}
```

Figura 46: Funció que gestiona el pas al següent nivell

8.2.2 Escenari 1

El model usat per a la casa del primer escenari s'ha importat des d'una pàgina web que distribueix models 3D. Per als objectes que formen part de la mecànica s'han modelat per separat. Els elements que componen aquest escenari són:

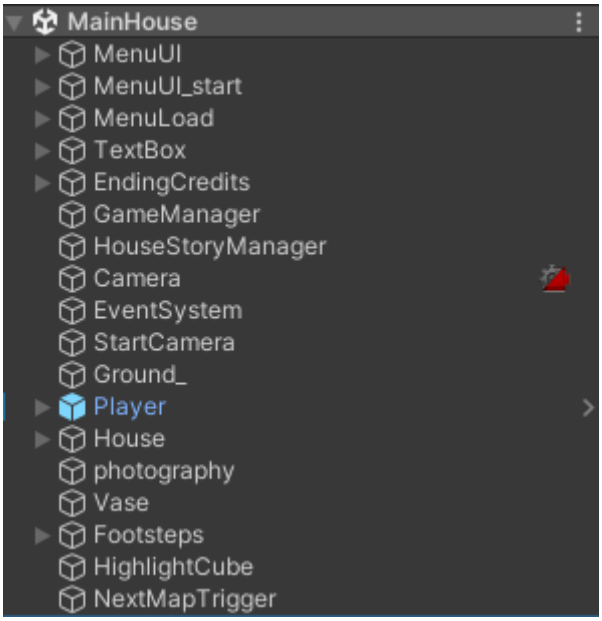

Figura 47: Elements de l'escenari 1

Alguns elements apareixen reiteradament en totes les escenes i es tractaran en els seus apartats corresponents, com són: les interfícies, la càmera, el *Game_Manager* i *NextMapTrigger*.

El *HouseStoryManager* és el script encarregat de comptar els objectes amb el que el jugador ha interactuat i habilitar la porta per a passar al següent nivell si ha interactuat amb totes les necessàries.

```
public class HouseStoryManager : MonoBehaviour
{
    bool[] totalItems = new bool[3] { false, false, false };

    0 referencias
    public void FoundItem(int i)
    {
        totalItems[i] = true;
    }
    0 referencias
    public bool GotAllItems()
    {
        return (totalItems[0] && totalItems[1] && totalItems[2]);
    }
}
```

Figura 48: Script que gestiona la mecànica de l'escenari 1

Els objectes interactius de la mecànica han estat completament modelats en Blender.

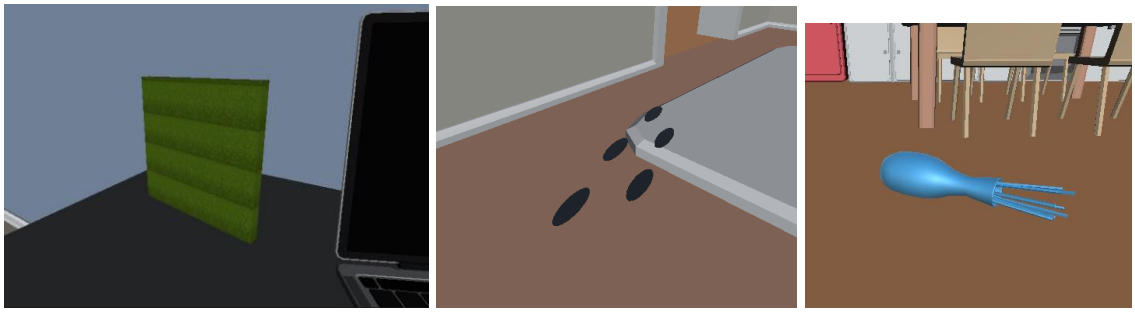

Figura 49: Objectes interactius que formen part de la mecànica de l'escenari 1

8.2.3 Escenari 2

Aquest és l'escenari on el jugador es troba un cementiri i ha de completar el mini joc de SimonDice. El mapa ha sigut modelat en Blender com un sol arxiu i els elements mòbils han estat modelats per separat.

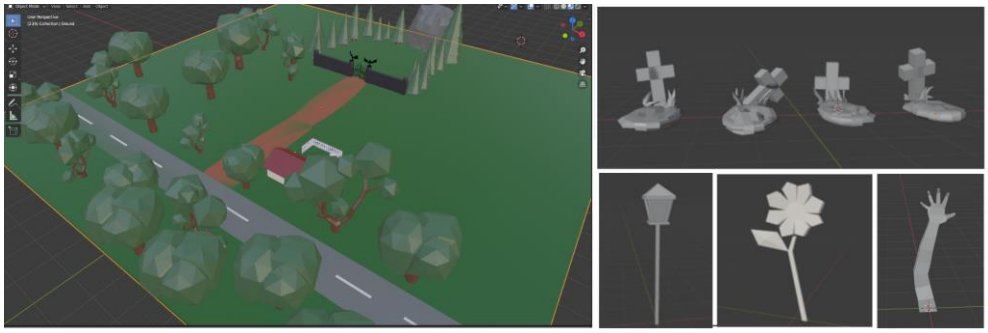

Figura 50: Modelatge en Blender del mapa i els objectes de l'escenari 2

Els elements que componen aquest escenari són:

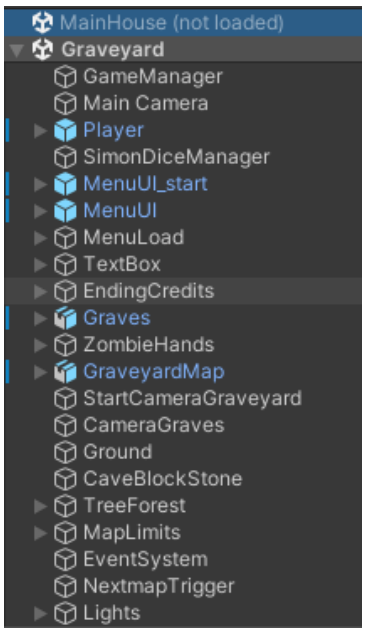

Figura 51: Elements de l'escenari 2

Igual que en els altres mapes, alguns elements apareixen reiteradament en totes les escenes i es tractaran en els seus apartats corresponents, com són: les interfícies, la càmera, el *Game_Manager* i *NextMapTrigger*.

L'encarregat de la gestió del mini joc és el script *SimonDiceManager* el qual està constituït per diferents funcions que desenvolupen diferents estats del joc.

- Inicialització: El joc comença quan el jugador entra en el trigger, una caixa de col·lisió just abans d'arribar a les tombes, i activa la primera funció:

```
private void StartSimon() //Fase
{
    //calculate fases and sequences
    fasesDone = 0;
    restaMm = maxIter - minIter;
    if (fases == 1) { sumIter = 1; firstIter = maxIter; Debug.Log("simonDice-> fases = 1"); }
    else if (restaMm > fases)
    {
        sumIter = Mathf.Floor(restaMm / fases);
        firstIter = maxIter - sumIter * fases;
        Debug.Log("simonDice-> restaMn<fases");
    }
    else { sumIter = 1; firstIter = 1; Debug.Log("simonDice-> else startsimon"); }

    //Create fasePattern
    fasePattern = new int[fases][];
    for (int i = 0; i<fases; i++) {
        float nIter = firstIter + sumIter * i;
        fasePattern[i] = new int[(int) nIter];
        for (int j = 0; j < fasePattern[i].Length; j++)
        {
            fasePattern[i][j] = Random.Range(0, 3); //range inclusive
        }
    }
    for (int i = 0; i < fases; i++)
    {
        for (int j = 0; j < fasePattern[i].Length; j++)
        {
            Debug.Log(fasePattern[i][j]+",");
        }
        Debug.Log("---");
    }
    Debug.Log("==========================");
    gameSimonDice();
}
```

Figura 52: Funció que inicialitza el joc SimonDice

Segons els paràmetres de dificultat, la funció crea una seqüència de colors per cada fase indicada. Com a exemple: si tenim 2 fases amb 4 iteracions cada una, primer es mostrarà la primera seqüència de 4 iteracions, seguidament es detectarà que el jugador imiti bé la seqüència i perseguirà mostrant la segona seqüència de 4 iteracions i esperarà de nou l'entrada del jugador.

Aquesta funció només crea la seqüència, mostrar-la s'encarrega la funció:

```
private void gameSimonDice()
{
    if (fasesDone < fases)
    {
        iterDone = 0;
        StartCoroutine(ShowIter(iterSpeed)); //show iteration of actual fase
    }
    else
    {
        Debug.Log("SimonDice-> updateIterarion OPENING CAVEDOOR!!");
        //open cavedoor
        StartCoroutine(openCaveDoor());
    }
}
```

Figura 53: Funció encarregada de mostra la seqüència de SimonDice al jugador

El *startCoroutine()* és una funció que proporciona Unity amb el que es poden crear temps d'espera. Corren de forma paral·lela a la principal línia d'execució. Per cada iteració que es mostra una mà de zombie esperarà cert temps condicionat per la dificultat, com més difícil més ràpid.

```
IEnumerator ShowIter(float t) //Iteration
{
    GameObject.Find("Player").GetComponent<PlayerMovement>().changeMovability(false);
    GameObject.Find("GameManager").GetComponent<Game_Manager>().ChangeCamera(2);
    yield return new WaitForSeconds(faseDelay);
    for (int i = 0; i < fasePattern[fasesDone].Length; i++)
    {
        if (fasePattern[fasesDone][i] == 0) { blueAnim.SetTrigger("actHand"); }
        else if (fasePattern[fasesDone][i] == 1) { yellowAnim.SetTrigger("actHand"); }
        else if (fasePattern[fasesDone][i] == 2) { redAnim.SetTrigger("actHand"); }
        else if (fasePattern[fasesDone][i] == 3) { greenAnim.SetTrigger("actHand"); }
        yield return new WaitForSeconds(t);
    }
    GameObject.Find("GameManager").GetComponent<Game_Manager>().ChangeCamera(0);
    GameObject.Find("Player").GetComponent<PlayerMovement>().changeMovability(true);
}
```

Figura 54: Funció encarregada de detectar si el jugador ha activat una tomba

Un cop el jugador ha vist la seqüència li toca trepitjar les tombes per ordre correcte. Aquesta acció l'executa l'objecte de la tomba que funciona com un botó. Cada tomba té el seu respectiu trigger i el script *GraveActioner,* que a l'entrar en contacte amb el jugador inicia un event que cridarà a la funció *UpdateIteration()* que es troba en el script *SimonDiceManager*.

El *UpdateIteration()* comprova si la tomba que el jugador ha accionat és el correcte en la seqüència, si és correcte i encara falten iteracions que el jugador ha d'entrar, el script s'espera a la següent activació de tomba. Si al comprovar la tomba activa aquesta és incorrecte, el jugador se situa endavant de les tombes i es repeteix la visualització de la seqüència de les mans.

```
private int groundID;
private bool triggered = false;
4 referencias
public void setID(int v)
{
    groundID = v;
}
Mensaje de Unity | 0 referencias
private void OnTriggerEnter(Collider colliderInfo)
{
    if (colliderInfo.tag == "Player" && !triggered)
    {
        triggered = true;
        GameObject.Find("SimonDiceManager").GetComponent<simonDice>().updateIteration(groundID);
    }
}
```

Figura 55: Script GraveActioner

```
public void updateIteration(int groundID)
{
    if(fasePattern[fasesDone][iterDone] == groundID) {
        iterDone += 1;
        Debug.Log("SimonDice-> updateIterarion CORRECT!!"); // sound of correct
        if (fasePattern[fasesDone].Length == iterDone)
        {
            fasesDone += 1;
            gameSimonDice();
        }
    }
    else {
        Debug.Log("SimonDice-> updateIterarion ERROR!!"); // sound of error
        GameObject.Find("Player").GetComponent<PlayerMovement>().changePlayerPos(playerCheckpoint, new Quaternion(0, 1, 0, 0));
        //fasesDone = 0; // comment if you don't wanna begin from first fase
        iterDone = 0;
        gameSimonDice();
    }
}
```

Figura 56: Funció de comprovació de seqüència correcte entrada pel jugador

8.2.4 Escenari 3

El tercer mapa està format per la cova, la qual crea un laberint que serà la mecànica del nivell. El modelatge s'ha fet completament en Blender per posteriorment importar-lo a Unity. En l'estat de desenvolupament del joc, els elements de recàrrega de la llanterna com els enemics no s'han modelat pel que només en veurem el prototip.

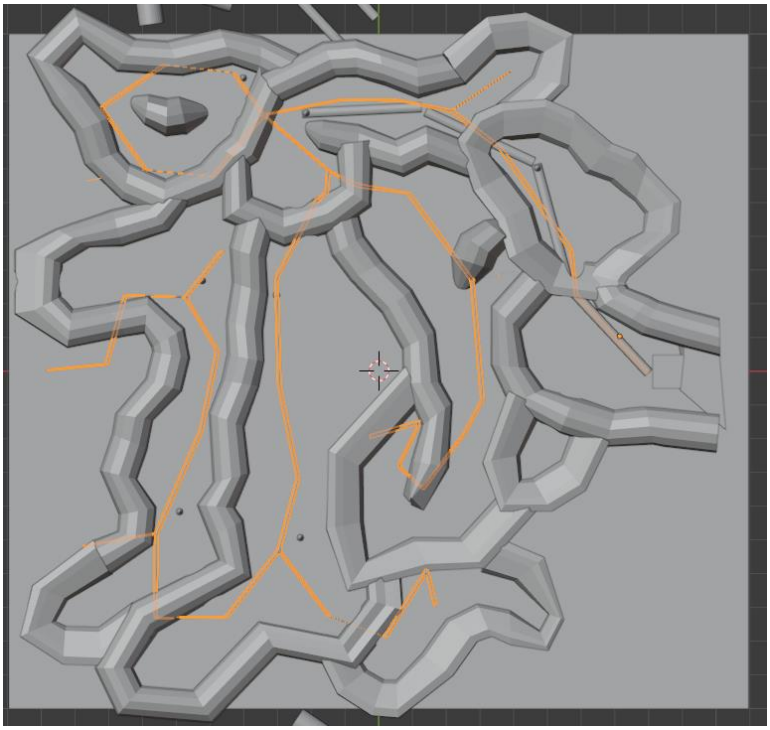

Figura 57: Modelatge de l'escenari 3

En aquest nivell s'introdueixen els objectes equipables següent el primer objecte la llanterna per a la protecció contra els enemics. La implementació s'explica més a fons en l'apartat 8.5 Objectes interactius.

Els elements que conformen el tercer escenari són:

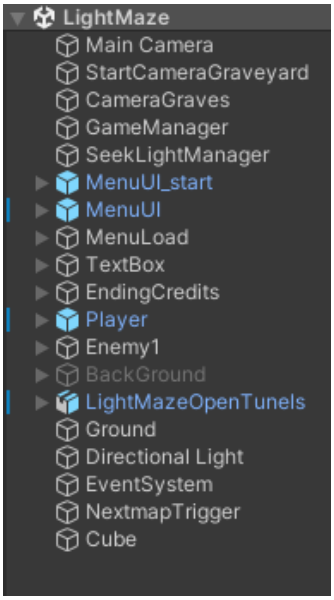

Figura 58: Elements de l'escenari 3

Com en els altres mapes, alguns elements apareixen reiteradament en totes les escenes i es tractaran en els seus apartats corresponents, com són: les interfícies, la càmera, el *Game Manager* i *NextMapTrigger*.

La mecànica del nivell es gestiona a través del script *SeekLightManager* que s'encarrega del temporitzador i l'augment de temps en aquest quan el jugador agafa una recàrrega de bateria.

Controla també tant la velocitat dels enemics perquè corrin més un cop el temporitzador hagi arribat a zero, com també el moment que comencen a perseguir al jugador. El senyal perquè un enemic comenci a perseguir al protagonista és donada per triggers situats en els passadissos del laberint. La mecànica dels enemics es tracta amb més detall en l'apartat 8.7 Enemics.

8.3 Nivell completat

Cada nivell té un objecte *NextMapTrigger* que es troba al final del mapa, normalment indicat per una porta o un túnel. Quan el jugador col·lideix amb el trigger aquest crea un event que crida la funció *finishMap()* del script *Game_manager* el qual comprova si el jugador és apte per passar al segon nivell o la mecànica encara no s'ha assolit.

L'event creat per l'objecte NextMapTrigger és:

```
private bool triggered = false;
Mensaje de Unity | 0 referencias
private void OnTriggerEnter(Collider colliderInfo)
{

    if (colliderInfo.tag == "Player" && !triggered)
    {
        Debug.Log("PlayerColisioned atr end trigger");
        triggered = true;
        GameObject.Find("GameManager").GetComponent<Game_Manager>().finisMap();

    }
}

Mensaje de Unity | 0 referencias
private void OnTriggerExit(Collider colliderInfo)
{
    triggered = false;
}
```

Figura 59: Script endMapTrigger que forma part de l'objecte NextMapTrigger

Cal assenyalar que els triggers que s'han implementat en aquest projecte necessiten un booleà que indici si és la primera vegada que s'activa el trigger, ja que si no es crida més d'una vegada l'event *OnTriggerEnter()*.

La funció que s'encarrega de fer el canvi d'escena és l'encarregada també d'actualitzar el nivell màxim assolit del jugador si escau.

```
public void finisMap()
{
    PlayerData playerData = player.GetComponent<PlayerManager>().playerData;
    Debug.Log("GameManager-> Map " + playerData.getCurrentMap() +" finsihed!!");
    //Sum maps done on player mapscompleted list
    int cm = playerData.getCurrentMap();
    int comp = playerData.getMapsCompleted();
    if (cm + 1 < mapsAvaliable) { playerData.setCurrentMap(cm+1); }
    if (comp + 1 < mapsAvaliable && cm + 1 > comp) { playerData.setMapsCompleted(comp + 1); }
    player.GetComponent<PlayerManager>().SaveToJson();

    if (playerData.getCurrentMap()>cm) { LoadScene(playerData.getCurrentMap()); }
    else //Game Finished
    {
        Debug.Log("GAME FINISHED!!!");
        GameObject.Find("Player").GetComponent<PlayerMovement>().changeMovability(false);
        GameObject.Find("EndingCredits").GetComponent<Canvas>().enabled = true;
    }
}
```

Figura 60: Funció finisMap() del script Game_Manager

8.4 Càmera

Per a la gestió de la càmera s'ha usat el paquet de eines Cinemachine de Unity. Aquest mòdul facilita la implementació de diferents càmeres i la seva gestió. La "Main Camera" passa a ser el CinemachineBrain que tindrà els elements generals per a gestionar els canvis de càmera.

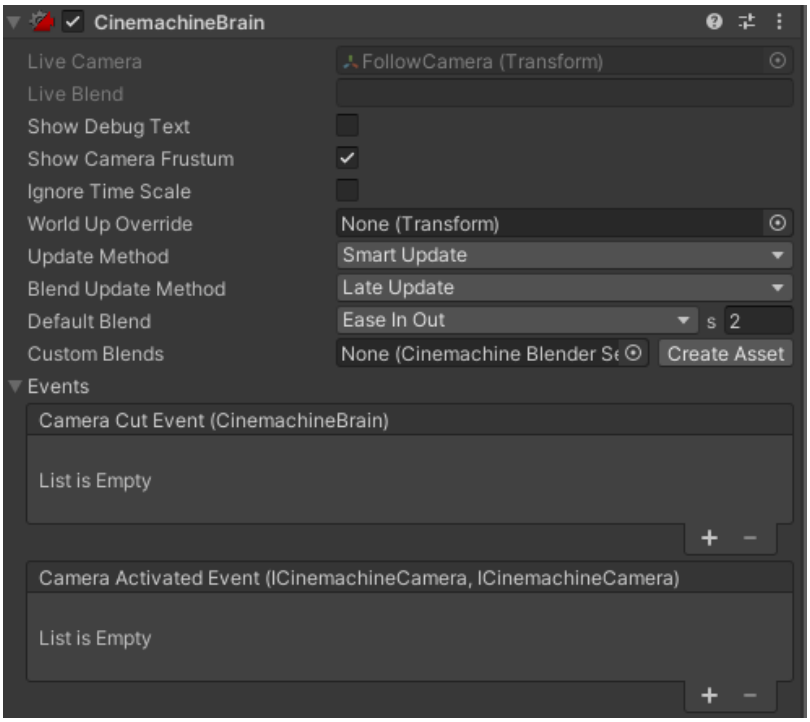

Figura 61: Paràmetres del CinemachineBrain

Amb Cinemachine tenim diferentes tipus de càmeres que podem integrar en el nostre joc. En aquest projecte s'han utilitzat dos tipus: el CinemachineFreeLook i el CinemachineVirtualCamera.

El tipus CinemachineFreeLook és la que dona la visió en tercera persona del personatge. La càmera l'hem integrada en l'objecte *Player* per a que sempre tingui la referència de la posició i rotació del personatge.

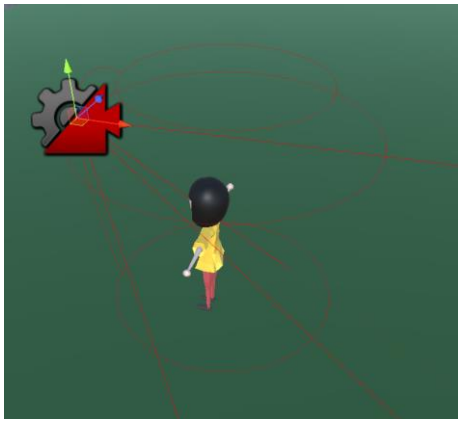

Figura 62: Càmera de tipus CinemachineFreeLook

La càmera CinemachineFreeLook és de tipus perspectiva en tercera persona i el moviment pot voltar circularment al voltant de l'objecte que té com a objectiu. El rang de moviment que té la càmera es pot veure a la Figura 58 amb línies vermelles al voltant del personatge. Aquest tipus de càmera té paràmetres ajustables per modificar el rang i la velocitat de moviment, l'objectiu al qual està orientat, entre altres.

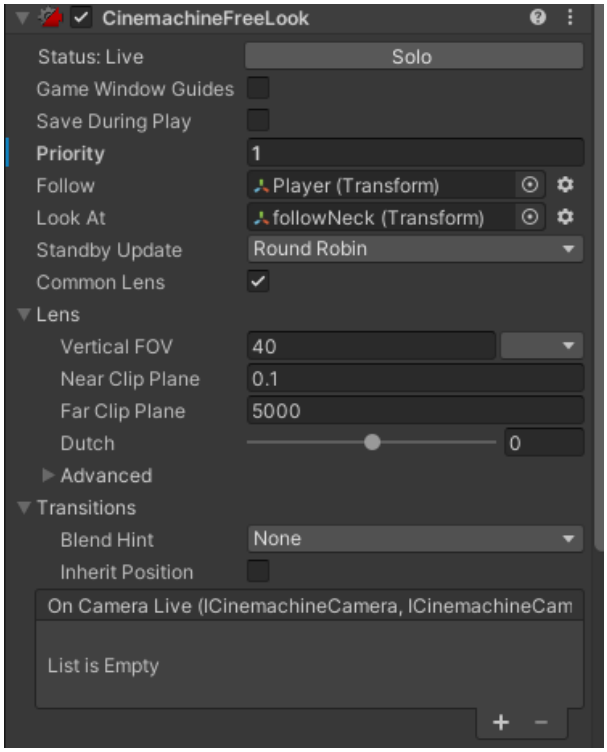

Figura 63: Paràmetres del CinemachineFreeLook

El mòdul Cinemachine conté l'element Cinemachine Collider, que al integrar-se en una càmera, aquesta no travessarà objectes que tinguin un element el collider de Unity.

Figura 64: Paràmetres del Cinemachine Collider

Per les càmeres estàtiques com la que trobem en el segon nivell la qual enfoca les tombes per mostrar la seqüència del SimonDice, fem servir el tipus de càmera CinemachineVirtualCamera. Aquesta funciona com una càmera normal de Unity però gestionada pel CinemachineBrain.

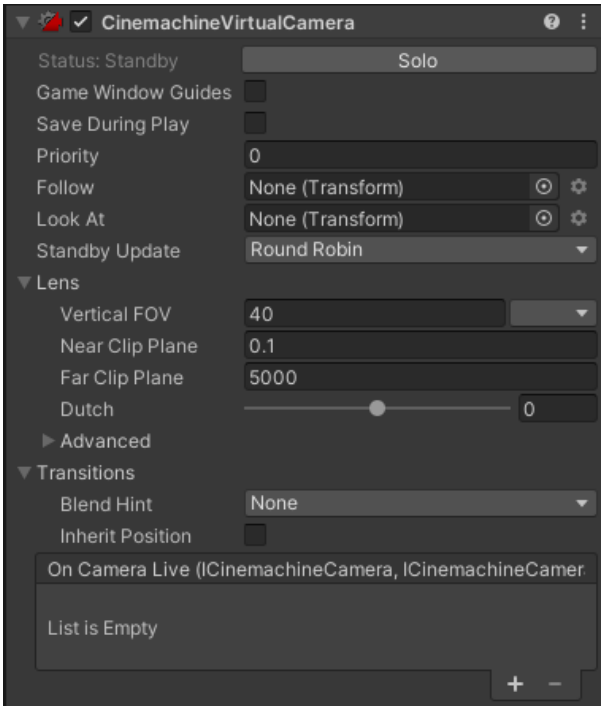

Figura 65: Paràmetres del CinemachineVirtualCamera

La càmera que es mostra per pantalla és la que té el valor "Priority" més alt. El Game *Manager* és l'encarregat de modificar aquests valors segons quina càmera es vol mostrar en cada situació. La funció utilitzada és la *ChangeCamera()* mostrada en la Figura 41.

8.5 Objectes interactius

Els objectes interactius són els elements de l'escenari amb què el jugador pot interactuar. Es poden diferenciar entre dos tipus d'objectes interactius:

- Objecte activat per proximitat
- Objecte activat per proximitat i acció del jugador

Un subtipus és l'objecte que el jugador que pot agafar, el qual desapareix en el moment d'activació.

Encara que canviïn algunes funcions entre aquests objectes, continuen tenint moltes semblances, raó per la qual farem servir l'herència per implementar-los.

La classe de la qual derivaran és *Interactable* que té implementada: la funció de canvi de material d'objecte a un color més viu, perquè destaqui quan el jugador estigui a prop; la funció de tornar el material al què tenia per defecte; i la funció d'acció. Aquesta última és una funció virtual de manera que per cada classe hereditària es podrà sobreescriure.

Classes implementades que deriven de *Interactable* són *ItemPickup* que seran els objectes que s'eliminen a l'interaccionar amb ells; i el *GraveActioner*, que seran les tombes que funcionaran com a botons per al mini joc SimonDice.

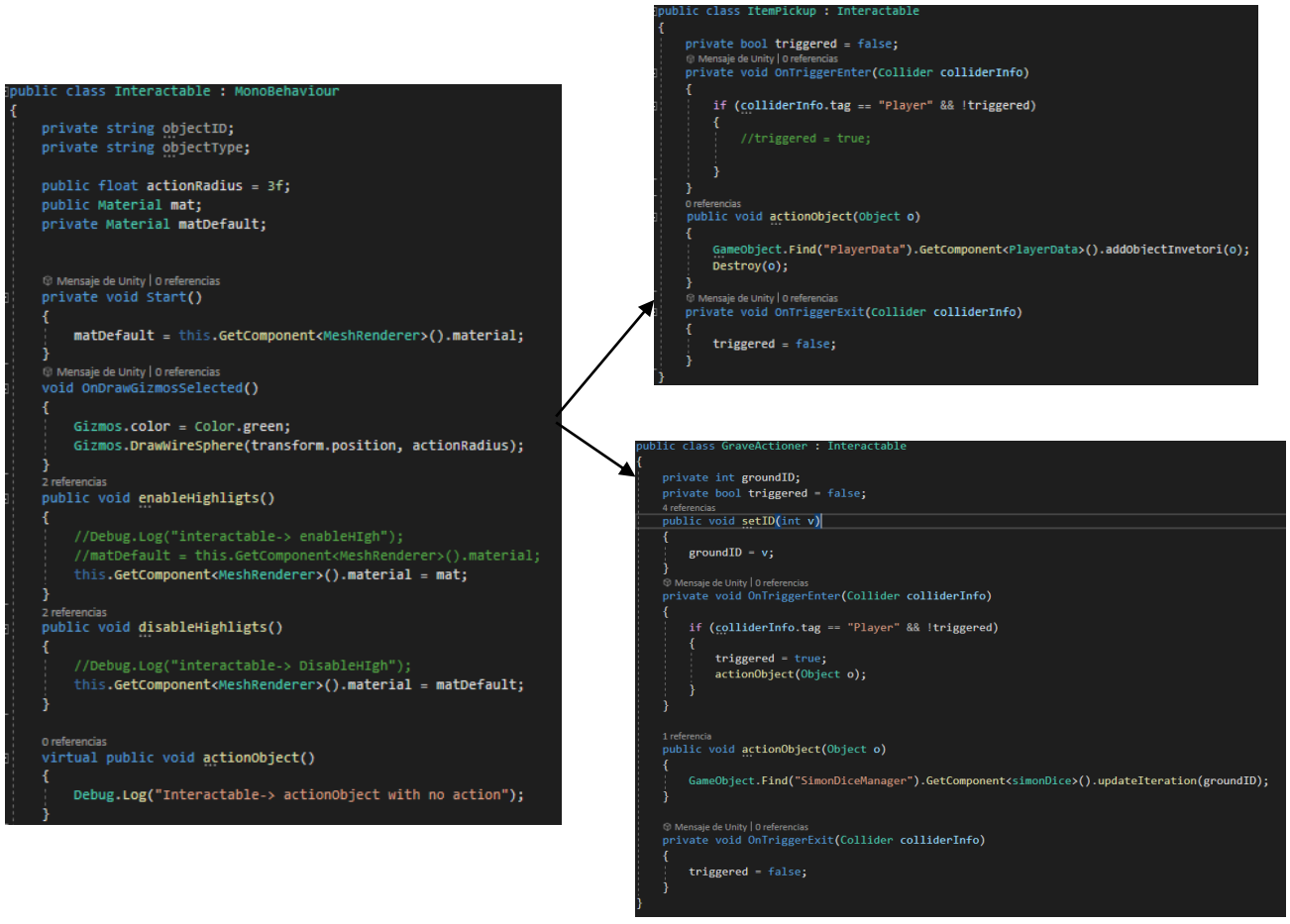

Figura 66: Diagrama d'herència de Interactable

Perquè funcioni l'objecte interactiu és necessari que contingui l'element collider, perquè al moment de col·lisionar amb el jugador, el collider cridi la funció *OnTriggerEnter()* en el mateix objecte.

8.6 Jugador

L'objecte Player encapsula totes les funcionalitats amb el que el jugador pot interactuar directament. Es composa de diferents objectes com la càmera virtual, el model del personatge i els objectes equipables. També inclou uns quants scripts i element de Unity que explicarem més en detall.

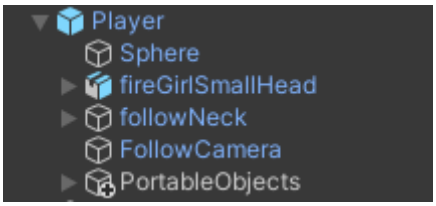

Figura 67: Objectes integrats en Player

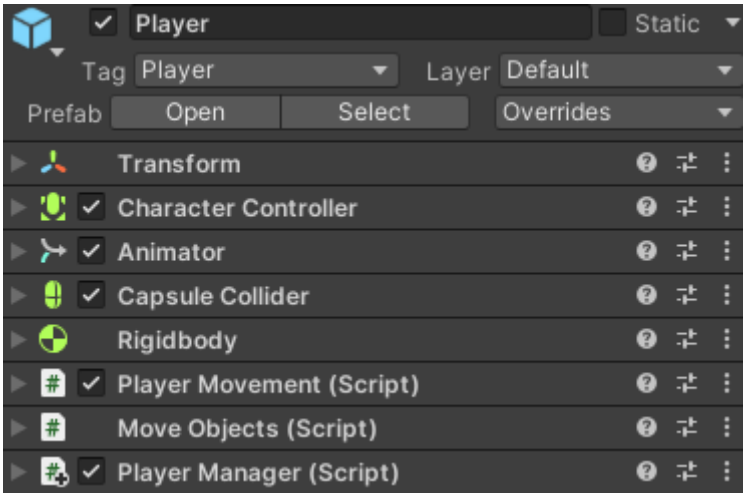

Figura 68: Elements que conté l'objecte Player

Moviment: El moviment és controlat pel jugador a través del Character Controller i el script *PlayerMovement*. El Charater Controller és un element de Unity que facilita el control del moviment posant a disposició mètodes de fàcil implementació.

A la funció Update, del script *PlayerMovement,* es criden dues funcions, una és el *Move()*, que gestiona el moviment entrat pel jugador; i l'altre és la funció *Fall()*, el qual gestiona la gravetat del personatge, la qual es detallarà amb més profunditat l'apartat de gravetat.

La funció *Move()* utilitza els mètodes, que Unity té integrats, següents:

- GetAxisRaw(): per detectar les entrades del jugador, en aquest cas les tecles W,S per l'eix horitzontal i A,D per l'eix vertical.
- Les funcions de la llibreria Mathf i Quaternion, per tractar el canvi de la posició del personatge que està en angles de Euler.
- CharacterController.Move(): mou el personatge cap a la direcció i amb la velocitat passat com a paràmetre.
- Animator.setBool(): activa l'animació passada per paràmetre. El personatge té l'animació de caminar i ajupir-se ("cannon").
- GetKey(): Torna "true" si la tecla indicada està activada, en aquest cas la "r".

```
private void Move()
{
    float horitzontal = Input.GetAxisRaw("Horizontal");
    float vertical = Input.GetAxisRaw("Vertical");
    Vector3 direction = new Vector3(horitzontal, 0f, vertical).normalized;

    if (direction.magnitude >= 0.1f)
    {
        float targetAngle = Mathf.Atan2(direction.x, direction.z) * Mathf.Rad2Deg + cam.eulerAngles.y;
        float angle = Mathf.SmoothDampAngle(transform.eulerAngles.y, targetAngle, ref turnSmoothVelocity, turnSmoothTime);
        transform.rotation = Quaternion.Euler(0f, angle, 0f);

        moveDir = Quaternion.Euler(0f, targetAngle, 0f) * Vector3.forward;

        animator.SetBool("Run", false);
        controller.Move(moveDir.normalized * speed * Time.deltaTime);
    }
    animator.SetFloat("Speed", (Vector3.Distance(lastPos, transform.position) * 10000f * Time.deltaTime));
    lastPos = transform.position;

    if (Input.GetKey("r")) animator.SetBool("cannon", true);
    else animator.SetBool("cannon", false);
}
```

Figura 69: Funció Move() del script PlayerMovement

El diagrama d'estats de l'animació del personatge està formada per 4 estats, el Idle, quan no es mou; el Walk, que s'activa quan el personatge es mou; el Run, activat si la velocitat de moviment és major al llindar estipulat, encara que en els tres nivells no es pot activar; i per útim el Cannonball, postura del personatge ajupit formant una bala de canó.

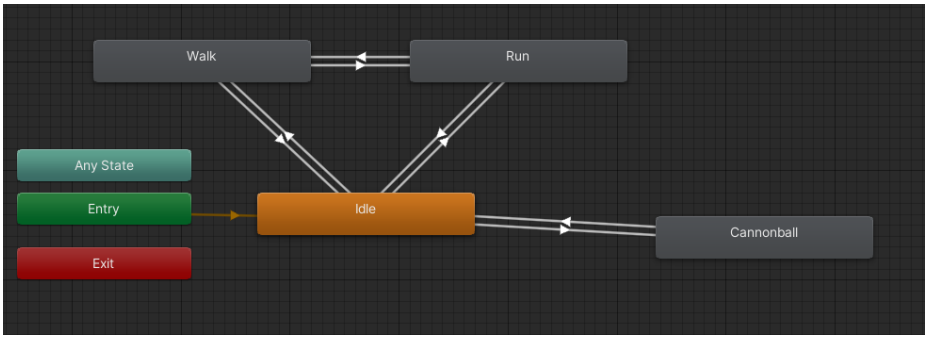

Figura 70: Diagrama d'estats de l'animació del protagonista

Gravetat: La gravetat que afecta el personatge s'ha implementat amb un moviment cap a terra, i amb un collider esfèric a la zona dels peus, vinculat a l'objecte Sphere, que frena el moviment quan toca el terra del mapa modificant el booleà *isGrounded*. Cada cop que es crida la funció *Fall()* el sistema comprova si toca terra, si no es compleix, aplicarà el moviment, direcció a la terra.

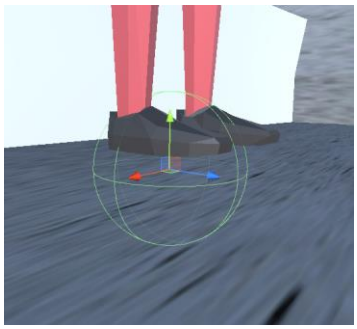

Figura 71: Collider que indica si el personatge toca a terra

```
private void Fall()
{
    if (!isGrounded)
    {
        fallVelocity.y = -2f;
        fallVelocity.y += gravity * Time.deltaTime;
        controller.Move(fallVelocity * Time.deltaTime);
    }
}
```

Figura 72: Funció Fall() del script PlayerMovement

Objectes equipables: Els objectes equipables estan vinculats al Player, d'ençà que s'inicia el joc, de manera que quan el jugador se n'equipa un, només s'activa l'objecte equipat. Per equipar l'objecte es buscarà primer entre els "children" de Player i després s'activarà o desactivarà modificant valor booleà de *SetActive()* a "true" o "false" respectivament.

Dades del jugador: En Unity, al canviar entre escenes es perd la informació del jugador si aquesta no es guarda en un sistema per separat. Per aquesta raó s'ha creat la classe PlayerData que conté totes les dades necessàries com els nivells completats i l'inventari.

```
[System.Serializable]
6 referencias
public class PlayerData
{
    public bool firstGame = true;
    public int currentMap = 0;
    public int mapsCompleted = 0;
    public int possiblePortables = 4; // future feature
    public string[] activatedPortables; // future feature
    public List<ItemValue> inventory = new List<ItemValue>();

    2 referencias
    public bool FirstGame() { return firstGame; }
    4 referencias
    public void SetFirstGame(bool b) { firstGame = b; }
    0 referencias
    public void GameStarted() { firstGame = false; }
    4 referencias
    public int getCurrentMap() { return currentMap; }
    2 referencias
    public void setCurrentMap(int i) { currentMap = i; }
    2 referencias
    public int getMapsCompleted() { return mapsCompleted; }
    1 referencia
    public void setMapsCompleted(int i) { mapsCompleted = i; }
}
```

Figura 73: La classe PlayerData

La informació de la classe es guarda fora de Unity per poder recuperar-la o bé quan s'ha fet el canvi de mapa o si s'ha inicial el joc i el jugador vol carregar la partida de la sessió anterior. Les dades es guarden en un fitxer en format Json i es guarda o extreu amb un mètode conversor que Unity té integrat en el paquet Generic.

```
public bool ExistData()
{
    string filePath = Application.persistentDataPath + "/PlayerData.json";
    return System.IO.File.Exists(filePath);
}
6 referencias
public void SaveToJson()
{
    string playerDataJSON = JsonUtility.ToJson(playerData);
    string filePath = Application.persistentDataPath + "/PlayerData.json";
    System.IO.File.WriteAllText(filePath, playerDataJSON);
    Debug.Log("PlayerManager->Json saved at: " + filePath);
}

2 referencias
public void LoadFromJson()
{
    string filePath = Application.persistentDataPath + "/PlayerData.json";
    string playerDataObject = System.IO.File.ReadAllText(filePath);
    playerData = JsonUtility.FromJson<PlayerData>(playerDataObject);
    Debug.Log("PlayerManager->JsonLoaded");
}
```

Figura 74: Mètodes de guardat i extracció de les dades jugador

8.7 Enemics

L'objecte enemic està compost per un model físic i pels components NavMeshAgent, Rigidbody i un collider.

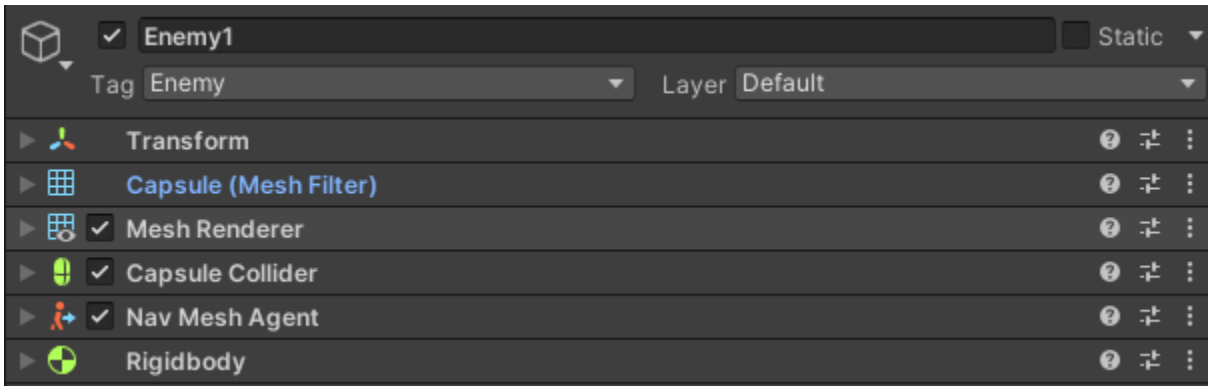

Figura 75: Components que formen l'enemic

El moviment dels enemics està implementat amb el Nav Mesh Agent, un element de Unity que crea una zona de navegació en el mapa segons les zones que el desenvolupador indica si són accessibles o no. Perquè funcioni l'enemic ha de contenir el component Rigidbody.

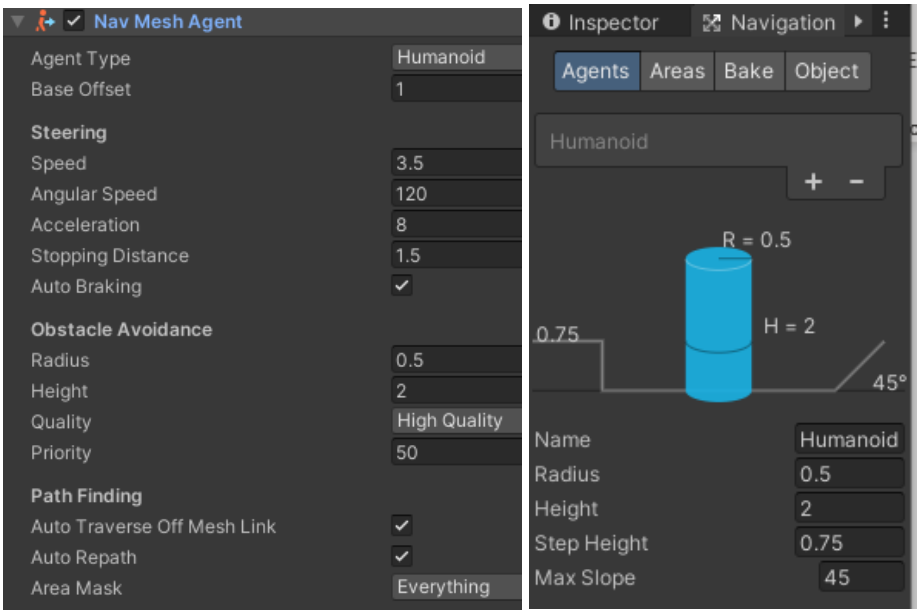

Figura 76: Paràmetres del Nav Mesh Agent

Una vegada ja tenim la zona per on pot caminar l'enemic només falta marcar l'objectiu perquè el segueixi. El NavMeshAgent té un mètode de moviment, *SetDestination()*, al qual se li passa la posició del jugador, de manera que l'enemic sempre es mourà on es troba el protagonista.

```
void Start()
{
    triggered = false;
    lookDestination = true;
    player = GameObject.Find("Player").GetComponent<Transform>();
    nav = this.GetComponent<NavMeshAgent>();
    //disableMovement();
}

// Update is called once per frame
Mensaje de Unity | 0 referencias
void Update()
{
    if (lookDestination) { nav.SetDestination(player.position); }
}
```

Figura 77: Implementació del moviment de l'enemic

La mecànica del nivell evita que els enemics es moguin si estan enlluernats per la llum de la llanterna del jugador. Per simular la llum s'ha creat una piràmide cònica, amb un collider integrat, situada enfront del jugador. Aquesta piràmide, quan xoca amb un enemic, enviarà l'event *OnTriggerEnter()* a l'enemic perquè es quedi quiet.

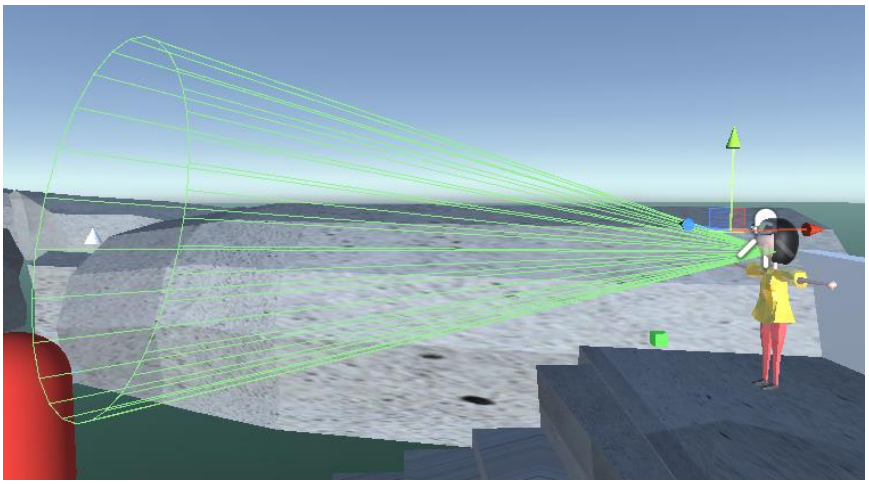

Figura 78: Piràmide cònica que simula la llum

```
public void enableMovement()
{
    nav.enabled = true;
    lookDestination = true;
    //nav.speed = 3.5f;
}

1 referencia
public void disableMovement()
{
    lookDestination = false;
    nav.enabled = false;
    //nav.speed = 0f;
}
```

```
private void OnTriggerEnter(Collider colliderInfo)
{
    if (colliderInfo.tag == "eyeSight" && !triggered)
    {
        triggered = true;
        disableMovement();
        Debug.Log("PROTOenemy-> Spotted!!");
    }
}
Mensaje de Unity | 0 referencias
private void OnTriggerExit(Collider other)
{
    triggered = false;
    enableMovement();
}
```

Figura 79: Implementació del comportament de l'enemic

8.8 Interfícies

Les tres interfícies: Menú inicial, Menú de pausa i Pantalla final, s'implementen de la mateixa manera. Estan formades per un Canvas, objecte pla, on es mostren totes les components UI que se li agregui.

Cada element que es fica en el Canvas serà un objecte independent i estaran compostes per:

- TextMeshPro – Text (UI): solució de Unity per inserir text en l'escenari, en aquest cas sobre el canvas del menú.
- Button: Objecte que funciona com un botó i activa un trigger si aquest és premut.

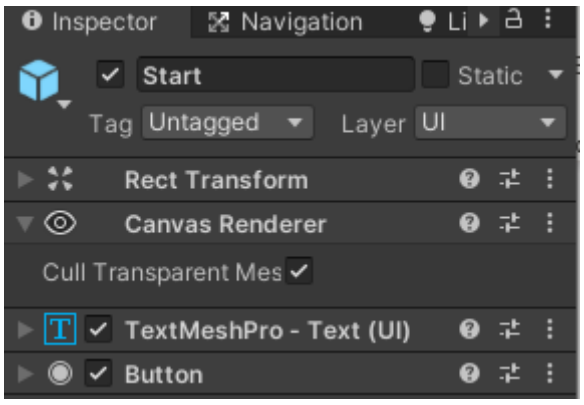

Figura 80: Components del botó Start en el menú inicial

Els Button tenen l'opció de tenir una referència a una funció i cridar-la directament al ser premut. En la Figura77 es pot veure que té vinculada la funció *Start_Game()* del *Game_Manager*.

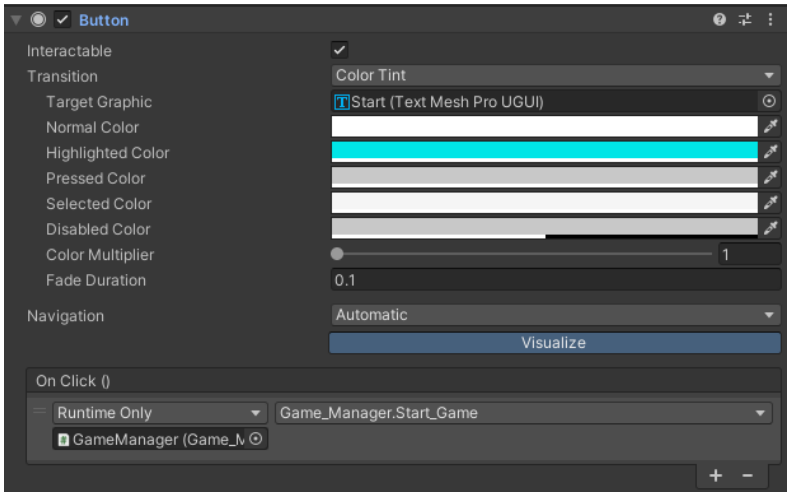

Figura 81: Component Button

Per mostrar o amagar un menú es busca el seu component Canvas es canvia el valor de la variable "enabled". En la Figura78 es pot veure com amaga tots els possibles menús de la manera indicada.

```
public void Resume_Game()
{
    Debug.Log("RESUM GAME");
    //Cursor.lockState = CursorLockMode.Locked;
    //Cursor.visible = false;
    GameObject.Find("Player").GetComponent<PlayerMovement>().changeMovability(true);
    ingame_menu.enabled = false;
    start_menu.enabled = false;
    load_menu.enabled = false;
    ChangeCamera(0);
}
```

Figura 82: Funció de resumir el joc en Game_Manager

9 Resultats

9.1 Legislació i normativa vigent

El projecte no presenta cap problema en aspectes legislatius

Al no tractar dades crítiques de caràcter personal del jugador, no es té en compte el Reglament General de Protecció de dades (RGPD).

La llei de serveis de la societat de la informació i comerç electrònic (LSSICE) tampoc s'aplica, ja que el nostre projecte no constitueix cap activitat econòmica.

9.2 Resultats finals

En aquest apartat es mostren les funcionalitats del joc amb captures de pantalla comentades.

Les dues interfícies: Menú principal i Menú de pausa, tractats en l'apartat 8.8 Interfícies.

Figura 83: Menú inicial

Figura 84: Menú de pausa

La interacció amb els objectes interactius, explicat en l'apartat 8.5 Objectes Interactius.

Figura 85: Seqüència d'interacció amb un objecte

Figura 86: Objectes faltant per completar la mecànica

La protagonista mou els objectes que tenen col·lisió implementada.

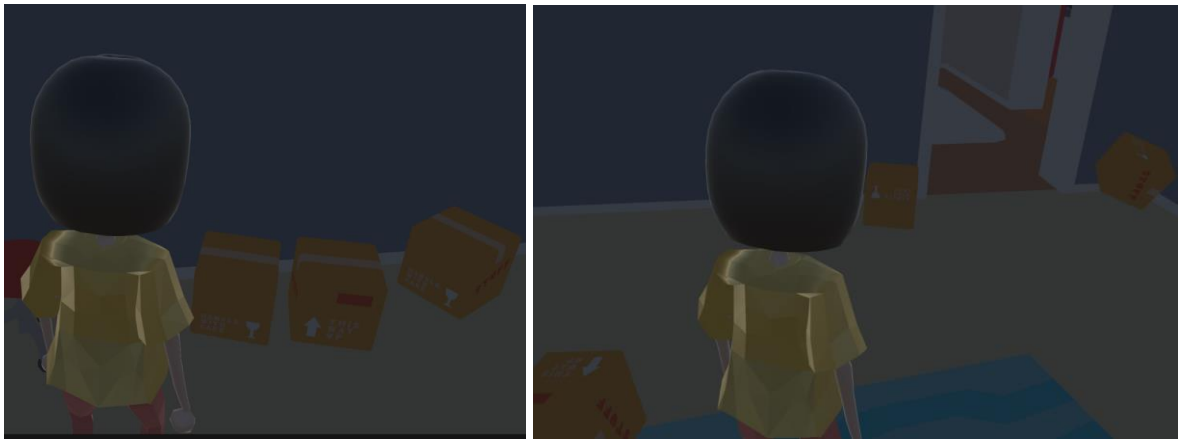

Figura 87: Col·lisió amb objectes mòbils

Un cop que s'han agafat els tres objectes necessaris, al tocar la porta de sortida el jugador passa al següent nivell.

Figura 88: Inici de l'escenari 2

Una vegada el jugador s'apropa a les tombes, la càmera canvia de posició, agafa millor angle i el joc SimonDice comença a mostrar la seqüència. La mecànica explicada es troba en l'apartat 8.2.3 Escenari 2.

Figura 89: Angle de visió per veure la seqüència de SimonDice

Figura 90: Jugador accionant les tombes en l'ordre de la seqüència

Un cop s'ha imitat la seqüència correctament la pedra que bloqueja el camí s'aparta i deixa pas perquè el jugador passi al següent nivell. El canvi de nivell està explicat detalladament en l'apartat 8.2.1 Game_Manager.

El jugador apareix en el tercer mapa, i té una llanterna que a l'il·luminar un enemic, aquest deixa de moure's. Aquesta mecànica apareix detallat en 8.7 Enemics.

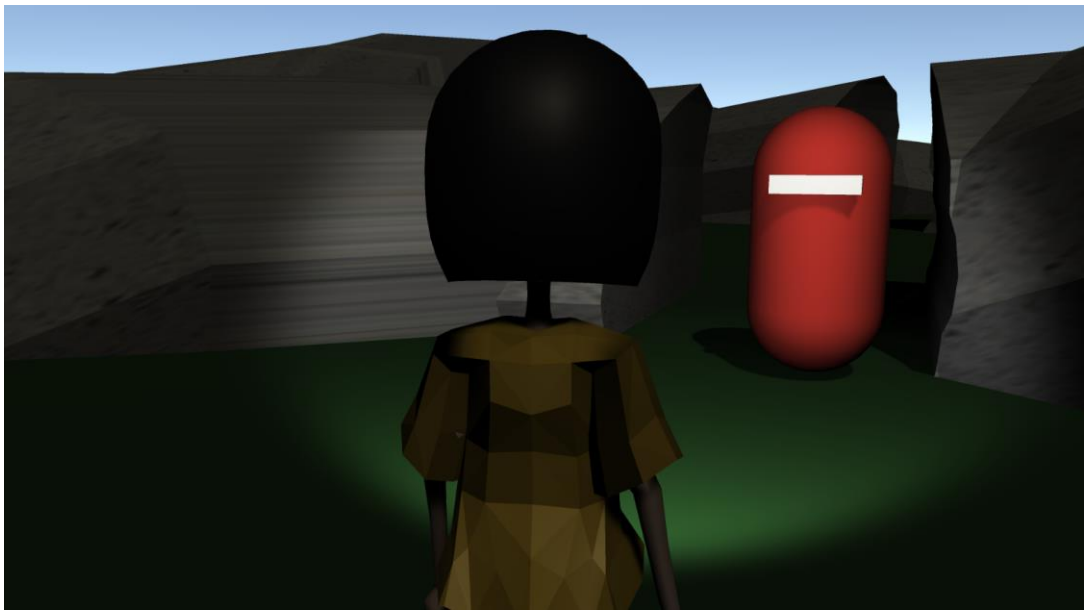

Figura 91: Jugador il·luminant a l'enemic per deixar-lo quiet

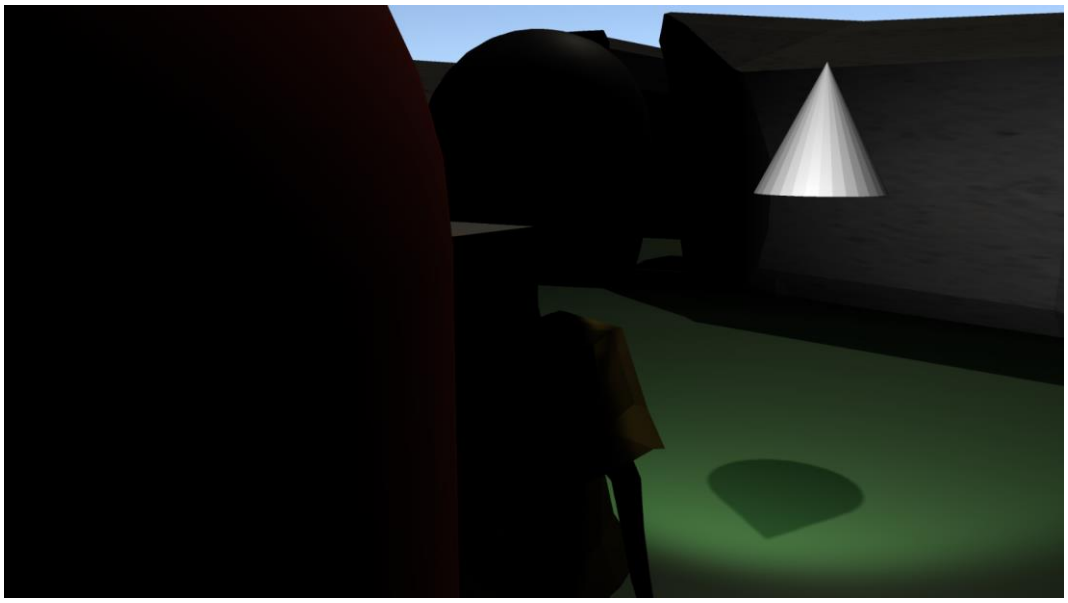

Figura 92: Recàrrega de bateria

La recàrrega de bateria suma temps al comptador per no quedar-se sense llum. Es parla d'aquesta mecànica en l'apartat 8.2.4 Escenari 3.

Si el jugador aconsegueix arribar a la sortida del laberint sense quedar-se sense llum i sense entrar en contacte amb cap enemic s'ha passat la demo del joc i apareix la pantalla final tractat en 8.8 Interfícies.

Figura 93: Pantalla final a l'acabar el joc

10 Conclusions

El desenvolupament d'aquest projecte ha sigut un repte pel poc coneixement de base amb el qual es partia, però a mesura que ha anat avançant s'ha agafat soltesa i ha incrementat el coneixement sobre les eines noves que s'ha fet servir, com és el cas de Unity. Un aprenentatge constant de nous conceptes que no s'han vist durant la carrera, sobretot en l'àmbit artístic com el modelatge dels personatges o la creació de mapes. En altres parts del projecte si s'ha pogut aplicar mètodes apresos durant el grau, repassant i millorant l'enteniment d'aquests, com podrien ser el mètode de planificació, el disseny de codi, diagrames de casos d'ús, entre altres.

La planificació amb el mètode Scrum ha ajudat a posar objectius a curt termini i avançar el projecte setmana a setmana, reduint la carga durant el desenvolupament del videojoc. Les tasques i el temps d'implementació estimat, que s'havien plantejat han sigut encertades amb l'excepció d'algunes que han tardat bastant més del que s'havia plantejat.

S'ha assolit gran part de l'objectiu que s'havia proposat. Les mecàniques principals, com els canvis d'escena, la interacció del jugador amb el personatge principal, els objectes interactius i les mecàniques dels primers nivells, funcionen adequadament. L'estètica ha sigut assolida però podria millorar en alguns aspectes com serien millorar el fons del Skybox, o la il·luminació.

El projecte queda obert a futures implementacions argumentats en l'apartat 11 Treball futur.

11 Treball futur

Al ser per nivells i estar inacabat, el projecte té la possibilitat d'expandir-se. A continuació es presenten parts del projecte que poden millorar o afegir-hi nou contingut.

- **Afegir vinyetes de text**: La falta de text impedeix que la història s'expliqui com és degut. Aquestes vinyetes apareixen quan s'agafa un objecte o simula els pensaments de la protagonista.

- **Mecàniques del mapa 3**: tant el temporitzador com les càrregues de bateria no han estat implementades pel que deixa el tercer nivell sense acabar.

- **Millorar les animacions**: Les animacions del personatge principal tenen alguns defectes visuals que es poden arreglar. Els enemics no tenen animacions el qual provoca una sensació molt estàtica, cosa que amb una animació s'evitaria.

- **Afegir música**: Música de fons i so ambiental poden incrementar els estímuls que el jugador rep quan juga.

- **Possibilitat de guardar diferents sessions**: Actualment, només guarda l'última sessió que s'ha jugat. Creant un fitxer per cada sessió evitaria el problema.

- **Millorar d'intel·ligència dels enemics**: La IA dels enemics és molt bàsica, de manera que es vol millorar canviant-la per un arbre de decisions. D'aquesta manera es poden fer mecàniques més complicades.

- **Optimització codi**: Millorar i optimitzar el codi per millorar-ne la claredat i evitar codi redundant.

- **Nivells**: El disseny del videojoc permet crear més nivells obrint les portes a noves mecàniques i enemics.

Bibliografia

Unity. *Unity User Manual*. (2023)

https://docs.unity3d.com/Manual/index.html

Blender. *Blender Reference Manual*. (2023)

https://docs.blender.org/manual/en/latest/

John French. *How to move objects in Unity*. (oct.2021)

https://gamedevbeginner.com/how-to-move-objects-in-unity/

EOM. *Evolución del mercado de los videojuegos*. (nov.2022)

https://elordenmundial.com/mapas-y-graficos/evolucion-mercado-videojuegos/

Talent.com, tokioshool.com. *Salaris feines de programador*. (2023)

https://es.talent.com/salary?job=dise%C3%B1ador+gr%C3%A1fico

https://www.tokioschool.com/formaciones/cursos-videojuegos/diseno/sueldo/

Itch.io. *Indie games online distributor*. (2023)

https://itch.io/developers

Gdevelop. *How to publish your game on itch.io*. (nov.2023)

https://gdevelop.io/es-es/page/how-to-publish-your-game-on-itch-io-and-why-you-should

Samyam. *Cinemachine blend between cameras*. (dic.2020)

https://www.youtube.com/watch?v=Ri8PEbD4w8A&t=75s

Aaron Rod. *Save/Load Json File*. (feb.2022)

https://www.youtube.com/watch?v=KotprYDkUV8

Free 3D. *3D models for free or comercial use*. (nov.2023)

https://free3d.com/3d-models/lowpoly-house

Annexos

Encara que la majoria de la informació sobre el projecte ha estat tractada en aquesta memòria no surt en la seva totalitat.

El projecte complet estarà accessible en una carpeta de Google Drive com també un executable del videojoc i un vídeo mostrant la jugabilitat.

El link a la carpeta és la següent:

https://drive.google.com/drive/folders/1isUGPVrktWkLxyy6DalUHJWd8wE7J8P4?usp=sharing

Manual d'usuari

Instal·lació

Requisits: és necessari que el sistema operatiu sigui Windows 10.

Per iniciar el joc cal:

- Descomprimir la carpeta demoGame.zip
- Executar el demoGame.exe

Controls

Visió:

- Ratolí

Moviment:

- Endavant: "W"
- Endarrere: "S"
- Dreta: "D"
- Esquerra: "A"

Accions:

- Interacció: "E"
- Ajupir-se: "R"
- Pausa: "ESC"