



**Universitat**  
de les Illes Balears

## **TREBALL DE FI DE GRAU**

**#ZOMBUIB**

**Xavier Vives Marcus**

**Grau d'Enginyeria Informàtica**

**Escola Politècnica Superior**

**Any acadèmic 2023-24**



**#ZOMBUIB**

**Xavier Vives Marcus**

**Treball de Fi de Grau**

**Facultat de: Escola Politècnica Superior**

**Universitat de les Illes Balears**

**Any acadèmic 2023-24**

Paraules clau del treball:

laberint, joc, realitat virtual, zombi

*Nom del tutor: Antoni Oliver Tomàs*

*Nom del segon tutor: Antoni Bibiloni Coll*

Autoritz la Universitat a incloure aquest treball en el repositori institucional per consultar-lo en accés obert i difondre'l en línia, amb finalitats exclusivament acadèmiques i d'investigació

Autor/a		Tutor/a	
Sí	No	Sí	No
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>



*En dedicació a:*

*Joan “poques feines” López Ferrer,*

*Marc “macaco” Melià Flexas*

*I Miquel “mosquitripo” Vidal Cortés.*



## Taula de continguts

Taula de continguts .....	i
Índex de figures .....	v
Índex de codis.....	vii
Nomenclatura.....	viii
Resum.....	1
Capítol 1. Introducció .....	3
1.1. Contextualització.....	3
1.2. Motivació.....	3
1.3. Objectius .....	4
1.4. Abast del projecte .....	4
1.5. Estructura del document.....	5
Capítol 2. Estat de l'art .....	7
2.1. Solucions existents a problemes semblants.....	7
2.1.1. Jocs similars existents .....	7
2.1.1.1. Call of Duty: Black Ops – Zombies.....	7
2.1.1.2. Risk of Rain 2 .....	8
2.1.1.3. Enter the Gungeon .....	9
2.1.1.4. Pavlov .....	9
2.2. Eines que poden fer-se servir per resoldre aquest problema. Anàlisi i selecció....	11
2.2.1. Kit de desenvolupament de jocs en RV.....	11
2.2.1.1. A-Frame .....	11
2.2.1.2. WonderlandEngine.....	11
2.2.2. Generació de laberints .....	11
2.2.2.1. Algorisme de Prim .....	11
2.2.2.2. Algorisme d'arbre binari.....	12
2.2.3. Objectes, models i animacions 3D .....	12
2.2.3.1. Mixamo .....	12
2.2.3.2. Sketchfab.....	12
2.2.4. Selecció.....	13

## Taula de continguts

---

2.2.4.1. Kit de desenvolupament de jocs en RV .....	13
2.2.4.2. Generació de laberints .....	13
2.2.4.3. Objectes, models i animacions 3D .....	14
Capítol 3. Desenvolupament del projecte.....	15
3.1. Gestió del projecte .....	15
3.1.1. Metodologia de desenvolupament.....	15
3.1.2. Planificació temporal.....	15
3.2. Anàlisi .....	16
3.2.1. Usuaris.....	16
3.2.2. Requisits d'usuari .....	16
3.2.2.1. Requisits funcionals.....	16
3.2.2.2. Requisits no funcionals.....	16
3.3. Disseny .....	17
3.3.1. Model de dades.....	17
3.3.2. Disseny arquitectònic .....	18
3.3.3. Disseny gràfic.....	19
3.3.3.1. Ambient.....	19
3.3.3.2. Armament .....	20
3.3.3.3. Zombis .....	20
3.3.4. Cicle de joc.....	21
3.4. Implementació .....	21
3.4.1. Base de dades.....	22
3.4.2. Servidor .....	22
3.4.2.1. Creació del servidor.....	22
3.4.2.2. Variables d'entorn.....	23
3.4.2.3. Accés a base de dades.....	24
3.4.3. Client .....	25
3.4.3.1. A-Frame .....	25
3.4.3.2. Càrrega de models 3D .....	26
3.4.3.3. Generació i dibuix del laberint .....	27
3.4.3.4. Navmesh del laberint .....	27
3.4.3.5. Components .....	29
3.4.3.5.1. Component "player" .....	29



---

3.4.3.5.2. Component “zombie” .....	30
3.4.3.5.3. Component “zombie-game” .....	32
3.4.3.6. Controls .....	33
3.5. Instal·lació .....	34
Capítol 4. Resultats.....	37
4.1. Pàgina principal .....	37
4.2. El joc .....	37
Capítol 5. Conclusions .....	40
5.1. Línies futures .....	42
Referències.....	44
Apèndix .....	46
Algorisme de Prim per Generar laberints .....	46
Generació de navmesh.....	48
A-Frame irregular-polygon custom component .....	57
Eliminar model 3D de memòria .....	58



## Índex de figures

Figura 1: Black Ops 3 - Zombies. Kino der Toten.....	7
Figura 2: Risk of Rain2. Mithrix logbook.....	8
Figura 3: Risk of Rain 2. Gameplay .....	8
Figura 4: Enter the Gungeon .....	9
Figura 5: Pavlov .....	9
Figura 6: Mixamo.....	12
Figura 7: Sketchfab.....	13
Figura 8: Cronograma.....	15
Figura 9: Model de dades.....	17
Figura 10: Model client-servidor .....	18
Figura 11: Model Arquitectònic .....	19
Figura 12: Escenaris de <i>Call of Duty: Black Ops – Zombies</i> (a) <i>Nacht der Untoten</i> , (b) <i>Der Riese</i> , (c) <i>Dead of the Night</i> , (d) <i>Mob of the Dead</i> .....	19
Figura 13: Models d'armes (a) <i>Colt M1911</i> [9], (b) <i>PPSH</i> [10], (c) <i>Raygun</i> [11], (d) <i>Moixipistola</i> [12] .....	20
Figura 14: Models de zombi .....	20
Figura 15: Diagrama de flux del joc.....	21
Figura 16: Esquema d'implementació Client.....	25
Figura 17: Animacions amb Blender .....	26
Figura 18: Laberint senzill.....	28
Figura 19: Laberint senzill. Navmesh (a), (b), (c) i (d).....	29
Figura 20: Zombi amb <i>hitboxes</i> .....	30
Figura 21: Pàgina principal .....	37
Figura 22: El joc .....	38
Figura 23: Escollir classe.....	38
Figura 24: Dins una partida .....	39
Figura 25: Vestíbul post-partida.....	39
Figura 26: Ciència per a Tothom .....	41

## Índex de figures

---

Figura 27: Begudes potenciadores. (a) Juggernog, (b) Speed Cola, (c) Quick Revive, (d) Deadshot Daiquiri.....42

Figura 28: Armes especials. (a) Apothicon Servant, (b) Wunderwaffe DG-2, (c) Monkey Bomb .....43

## Índex de codis

Codi 1: Implementació SQL de la base de dades .....	22
Codi 2: Creació del servidor .....	23
Codi 3: Configuració del servidor .....	24
Codi 4: Servidor emprant fitxer de configuració.....	24
Codi 5: Servidor. Accés a base de dades .....	25
Codi 6: A-Frame. Escena simple .....	25
Codi 7: A-Frame. Models animats .....	26
Codi 8: Algorisme de Prim per generar laberints .....	27
Codi 9: Pathfinding periòdic .....	31
Codi 10: Zombie hitPlayer().....	31
Codi 11: Detecció d'impactes .....	32
Codi 12: Controls .....	33
Codi 13: Controls en RV .....	33
Codi 14: Instal·lació Node.js i MySQL .....	34
Codi 15: Dependències Node.js .....	34
Codi 16: Configuració supervisor .....	35
Codi 17: Implementació en javascript de l'algorisme de Prim per generar laberints.....	48
Codi 18: Implementació en javascript per generar la navmesh del laberint .....	57
Codi 19: Eliminació de models 3D en memòria principal .....	59

<b>LTIM</b>	Laboratori de Tecnologies de la Informació Multimèdia.
<b>Roguelike</b>	Gènere de videojoc caracteritzat per tenir habilitats i ítems aleatoris, generació procedural de nivells, i mort permanent. És a dir, quan el jugador perd la partida, es perd tot el progrés.
<b>UIB</b>	Universitat de les Illes Balears.
<b>VR</b>	Realitat virtual ( <i>virtual reality</i> )
<b>AR</b>	Realitat augmentada ( <i>augmented reality</i> )
<b>XR</b>	Realitat estesa ( <i>extended reality</i> ). Agrupa una sèrie de tecnologies immersives com VR i AR entre d'altres.

## Resum

Aquest TFG tracta com s'ha plantejat, dissenyat i implementat un joc de trets en realitat virtual basat en web fortament inspirat en la saga de *Call of Duty: Black Ops – Zombies* on el jugador s'enfronta a hordes de zombis organitzats en rondes amb l'objectiu de sobreviure el màxim temps possible. Per fer-ho s'ha emprat A-Frame, un kit de desenvolupament de codi obert dedicat a crear experiències RV en web.

Es tracten temes com: algorismes de generació de laberints de forma aleatòria, desenvolupament d'escenes 3D, animacions i modelatge d'objectes 3D, desenvolupament d'experiències de RV, entre d'altres.

Ha estat desenvolupat majoritàriament en JavaScript, HTML i CSS i La metodologia emprada per la gestió d'aquest projecte és una metodologia incremental basada en prototipus.

Després d'exposar el projecte a la fira organitzada per la UIB: *Ciència per a tothom*, els usuaris ens transmeteren les seves impressions que foren majoritàriament positives i es va observar el canvi de comportament entre estudiants de diferents edats.

També es tracten possibles ampliacions que es podrien fer al joc en un futur per expandir l'experiència de joc.

El resultat final es troba disponible a través de l'adreça [zombis.ltim.uib.es](http://zombis.ltim.uib.es) i és compatible tant amb equipament de RV com amb un PC tradicional.





## Capítol 1. Introducció

En aquest capítol es dona una visió general del projecte explicant el context d'aquest Treball de Fi de Grau, els motius personals que han duit a realitzar-lo, els objectius que es volen aconseguir, l'abast del projecte i l'estructura del document.

### 1.1. Contextualització

La realitat virtual ha emergit com una tecnologia amb un impacte significatiu en diversos sectors, com ara la indústria, l'educació i l'entreteniment [1]. Aquesta tecnologia no només ha canviat la manera en què interactuem amb el món digital, sinó que també ha obert noves oportunitats per a la innovació i la creativitat. Amb l'avenç constant de les aplicacions web, la integració de la realitat virtual amb les tecnologies web s'ha convertit en una àrea d'investigació molt prometedora i fascinant. Aquest Treball de Fi de Grau (TFG) se centra específicament en el desenvolupament d'un joc de realitat virtual basat en web.

El joc que es desenvoluparà en aquest TFG serà un joc de supervivència en realitat virtual basat en web, estructurat per rondes i amb un enfocament en els trets. Els jugadors hauran d'enfrontar-se a diverses rondes d'enemics, utilitzant diferents tipus d'armes de foc per eliminar-los. A mesura que el jugador avanci i elimini més enemics, la dificultat del joc augmentarà progressivament, fent que cada ronda sigui més desafiant que l'anterior. L'objectiu principal del joc és sobreviure el màxim nombre de rondes possibles, demostrant habilitat i estratègia en un entorn virtual cada vegada més hostil.

En resum, aquest TFG oferirà una visió sobre com la realitat virtual amb combinació a les tecnologies web, poden crear noves formes d'entreteniment en l'àmbit digital.

### 1.2. Motivació

*Call of Duty: Black Ops III – Zombies* és el meu joc preferit per la immersiva narrativa, escenaris intrigants i el desafiament que presenta als jugadors aquest mode de joc. No obstant això, la saga ha patit, des d'un punt de vista personal, un gran declivi en qualitat. Les noves entregues de la saga no han pogut satisfer les meves expectatives i m'han duit a cercar noves alternatives. Ara bé, tot i existir alternatives interessants, cap ofereix la mateixa experiència que *Call of Duty: Black Ops – Zombies* presenta.

Aquesta situació ha estat el motiu principal pel qual he decidit que, si el que el mercat actual ofereix no és del meu gust, millor crear jo mateix un joc que ho sigui. Evidentment, el producte creat en aquest TFG no serà comparable al producte que una companyia milionària amb múltiples treballadors i diversos anys de desenvolupament ofereixi doncs només soc un desenvolupador, dispòs de molt menys temps i no tenc un pressupost milionari.

Adicionalment, desenvolupar una aplicació en realitat virtual és una cosa que no he fet mai i sembla interessant.

## 1. Introducció

---

### 1.3. Objectius

L'objectiu principal és desenvolupar un joc de trets en primera persona en el que el jugador s'enfronta a hordes de zombis organitzats en rodes. A mesura que el jugador progressi, més enemics apareixeran i més difícils d'eliminar seran. El joc no té final com a tal sinó que el jugador ha de sobreviure el màxim temps possible. El joc s'acabarà quan el jugador es vegi sobrepassat i els zombis acabin amb ell.

El joc estarà pensat per ser jugat amb equipament de VR, però també hauria de ser accessible a través d'un PC corrent.

De cada partida es recopilarà informació per poder elaborar una taula de puntuacions que mostri el top 10 millors jugadors. Aquesta taula ajudarà a crear una competició entre els jugadors per estar-hi de la mateixa manera que en les màquines arcade.

Finalment, el darrer objectiu és tenir una versió jugable per exposar a la fira *Ciència per a tothom* que organitza la UIB. Per complir aquest objectiu haurem de dur una gestió del calendari adequada.

### 1.4. Abast del projecte

El joc no cerca ser un AAA (videojocs produïts o distribuïts pels editors amb més renom dins la indústria) [2] per les evidents limitacions que tenim. Per tant, moltes característiques presents en jocs similars, no estaran presents en aquest projecte. Especialment en l'àmbit del modelatge i animacions 3D.

No s'esperen incloure animacions de recàrrega d'armes per la falta de coneixement per crear-les. Les armes tendran una certa quantitat de munició. Una vegada s'hagi esgotat, el jugador no tindrà res a fer.

Tampoc s'espera una gran varietat de models 3D. Ni en l'àmbit d'armament ni en els zombis. Una altra vegada, això és deu al desconeixement de tècniques de modelatge i a què aquest projecte tampoc té com a objectiu oferir una gran varietat.

La trajectòria dels trets serà en línia recta i no serà afectada ni pel fregament de l'aire ni per la gravetat. Això serà així perquè no té sentit implementar un sistema de físiques per aquest detall que serà imperceptible a causa de les curtes distàncies del joc. Tampoc hi haurà destrucció de l'entorn quan el jugador dispari a les parets de l'escenari.

Les armes no tendran cap dispersió de bala. És a dir, tots els trets sortiran en línia recta des del canó en la direcció que el jugador està apuntant. Això serà així perquè apuntar amb els comandaments de RV ja és prou complicat per a afegir-hi una dispersió aleatòria addicional.

Que l'algorisme de *pathfinding* que emprin els zombis per trobar el jugador no trobi el camí més curt serà acceptable perquè elaborar un algorisme de *pathfinding* en un entorn 3D on l'objectiu es va movent no és una tasca senzilla. Addicionalment, tampoc és l'enfocament que aquest projecte té.

S'elaboraran dues versions. Una per PC i un altre per equipament de RV. Ara bé, dispositius mòbils, tauletes, etc. no tendran suport.

No es té intenció d'implementar un mode multijugador en línia que permeti a diversos jugadors jugar una mateixa partida de forma cooperativa.

### 1.5. Estructura del document

Aquest document es troba estructurat per capítols. El segon capítol és Estat de l'art on s'exposen i descriuen tècniques i solucions ja existents a problemes similars que poden resultar útils. El tercer capítol és Desenvolupament del projecte format per tots els apartats de gestió, anàlisi, disseny, implementació i instal·lació. El quart capítol és Resultats on es comenten els productes obtinguts. El cinquè i darrer capítol compté les conclusions que s'han obtingut i possibles millores per al futur.



## Capítol 2. Estat de l'art

Aquest capítol exposa i descriu les tècniques i solucions existents que poden resultar d'utilitat a l'hora de desenvolupar el projecte.

### 2.1. Solucions existents a problemes semblants.

Abans d'entrar en el procés de desenvolupament, podem observar què han fet altres amb l'objectiu d'aprendre d'ells. Què fan bé? Quines coses podem evitar?

#### 2.1.1. Jocs similars existents

A continuació es descriuen un conjunt de jocs amb característiques similars al joc que volem desenvolupar.

##### 2.1.1.1. *Call of Duty: Black Ops – Zombies*

És un mode de joc dins la franquícia *Call of Duty*, en concret, de la saga *Black Ops*. En aquesta experiència de joc, els jugadors s'enfronten a onades de zombis organitzades en rondes de forma cooperativa fins a 4 jugadors.

A mesura que progressen en la partida, van aconseguint punts que poden intercanviar per millor armament, i altres millores que incrementen les seves probabilitats de supervivència. L'objectiu del joc és, en definitiva, sobreviure el màxim temps possible. Doncs, com més temps passi, més zombis apareixen per ronda i més difícils d'eliminar són.

Ofereix una àmplia gamma de mapes ambientats en diferents ubicacions i ambientacions. Cadascun inclou característiques úniques com poden ser objectes construïbles, armament, diferents enemics, narrativa de la història, etc. La Figura 1 mostra una captura d'un dels mapes més emblemàtics de la saga. Kino der Toten.



Figura 1: Black Ops 3 - Zombies. Kino der Toten

## 2. Estat de l'art

### 2.1.1.2. Risk of Rain 2

És un joc *roguelike* multijugador de ritme intens. Els jugadors juguen com supervivents que han aterrat forçosament a *Petrichor V*, un planeta extraterrestre amb criatures hostils.

L'objectiu és progressar a través dels diferents nivells mentre aconsegueixen ítems que augmenten les seves probabilitats d'èxit. Una vegada arriben al darrer nivell on s'hauran d'enfrontar a *Mithrix* (Figura 2), un ciclops humanoide amb poders divins que impedeix que els supervivents escapin del planeta.

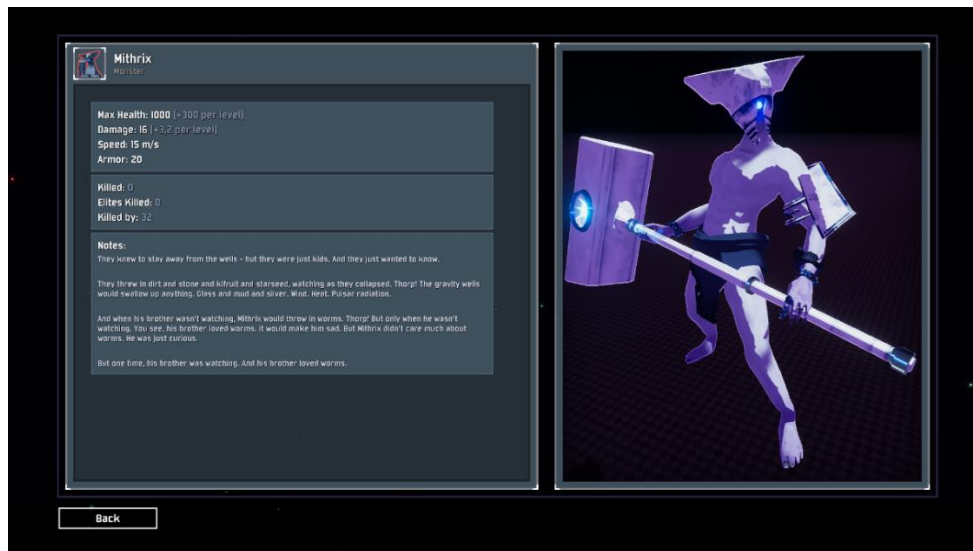


Figura 2: Risk of Rain2. Mithrix logbook

Els jugadors poden elegir entre diferents classes de personatges diferents (supervivents). Cadascun compta amb habilitats i estils de joc únics que, combinat a el fort component aleatòria d'obtenció d'ítems fa que cada partida se senti diferent respecte a les altres.



Figura 3: Risk of Rain 2. Gameplay

## 2.1. Solucions existents a problemes semblants.

### 2.1.1.3. *Enter the Gungeon*

És un *roguelike* on els jugadors tenen diferents personatges jugables per elegir, cadascun amb les seves peculiaritats i estils de joc únics.

Cada nivell està generat de forma procedimental formant un laberint d'habitacions plenes d'enemics.

L'objectiu del joc és progressar a través dels diferents nivells (anomenats càmeres) aconseguint objectes i armes que assegurin l'èxit de la partida. El propòsit final consisteix a viatjar al passat i completar una missió específica que dependrà del personatge que el jugador hagi escollit.

La principal dificultat d'aquest joc radica en la gran quantitat de projectils que ha d'esquivar el jugador per no ser ferit.



Figura 4: Enter the Gungeon

### 2.1.1.4. *Pavlov*

És un joc multijugador de trets en RV. Els jugadors formen equips i s'enfronten en diferents escenaris i missions. La característica distintiva de *Pavlov* és que permet interactuar de forma realista amb l'entorn mitjançant dispositius de RV com ara el moviment, llençar objectes, recarregar armes, etc.



Figura 5: Pavlov

## 2. Estat de l'art

### 2.1.2. Selecció de característiques

Després d'haver vist diferents jocs similars, toca valorar què trobam que fan bé i què fan no tan bé. La Taula 1 mostra les característiques, el joc d'origen i una valoració sobre si la característica en qüestió és interessant pel nostre joc.

Joc/s	Característica	Valoració (S/N)
<i>Risk of Rain 2</i> <i>Call of Duty: Black Ops – Zombies</i> <i>Enter the Gungeon</i>	Els enemics són controlats pel mateix programa i aquests cerquen el jugador	S
<i>Pavlov</i>	L'origen i direcció del tret depèn de la posició i inclinació de la mà del jugador com si aquest tingués una arma real.	S
<i>Call of Duty: Black Ops - Zombies</i>	Ús de l'escenari, il·luminació i música per crear un ambient de terror.	S
<i>Risk of Rain 2</i> <i>Enter the Gungeon</i>	Permetre a l'usuari elegir el seu estil de joc mitjançant l'ús de classes.	S
<i>Call of Duty: Black Ops – Zombies</i> <i>Risk of Rain 2</i> <i>Enter the Gungeon</i>	Si el jugador mor, es perd tot el progrés de la partida.	S
<i>Risk of Rain 2</i>	Existeix un enemic final que, en ser derrotat, acaba la partida.	N
<i>Risk of Rain 2</i> <i>Enter the Gungeon</i>	La generació del mapa és aleatòria.	S
<i>Call of Duty: Black Ops - Zombies</i>	Sobreviure a onades d'enemics organitzades en rondes els quals es faran més forts a mesura que avanci el temps.	S
<i>Call of Duty: Black Ops – Zombies</i> <i>Enter the Gungeon</i> <i>Pavlov</i>	El jugador ha de recarregar les armes.	N
<i>Call of Duty: Black Ops – Zombies</i>	Multijugador en línia	N

Taula 1: Característiques de jocs



## 2.2. Eines que poden fer-se servir per resoldre aquest problema. Anàlisi i selecció.

En aquest apartat, s'exploren diverses eines disponibles que podran ser emprades per abordar els principals problemes que aquest projecte presenta. Finalment, procedirem a la selecció de les eines que més s'ajustin a les nostres necessitats.

### 2.2.1. Kit de desenvolupament de jocs en RV

Un kit de desenvolupament de jocs és un programari que facilita la creació i renderització de gràfics generats per ordinador així com proporcionar eines i funcionalitats per al desenvolupament com ara gestió d'*assets*, optimització de rendiment, motor de física, etc.

Dins l'àmbit de generació de gràfics per ordinador basats en web, hem trobat dues opcions bastant prometedores:

#### 2.2.1.1. A-Frame

A-Frame [3] és un *framework* de codi obert per crear escenes 3D a través de web i poder interactuar amb elles ja sigui mitjançant teclat i ratolí o bé, emprant equipament de realitat virtual. Està construït damunt HTML de manera que s'abstrau de possibles problemes i complicacions que pot tenir fer feina amb dispositius VR permetent als desenvolupadors treballar emprant un llenguatge de marques com HTML i JavaScript. A-Frame és també amb una gran varietat navegadors i equipament VR.

#### 2.2.1.2. WonderlandEngine

WonderlandEngine [4] és un motor de gràfics lleugers principalment dissenyat per treballar en web. Permet crear aplicacions XR basades en web altament optimitzades i accessibles a gairebé tots els dispositius.

Adicionalment, posen a disposició dels desenvolupadors Wonderland Editor, un editor d'aplicacions 3D que permet al desenvolupador crear, optimitzar i modificar *assets*.

### 2.2.2. Generació de laberints

Per a la generació de nivells, és interessant que cada vegada que comença una partida, el mapa sigui diferent. Per complir-ho, es podria generar un laberint de forma completament aleatòria en cada partida. Per fer-ho, ens podem ajudar de diferents algorismes com els descrits a continuació. Ara bé, aquests només són uns quants d'entre tots els algorismes que existeixen.

#### 2.2.2.1. Algorisme de Prim

- Es comença amb una quadrícula plena de parets.
- S'escull una cel·la i es marca com a part del laberint. Les parets de les cel·les veïnes que no s'han visitat s'afegeixen al conjunt de veïns.
- Mentre hi continuï havent parets a la llista: Selecciona una paret de forma aleatòria. Si una de les dues cel·les que divideix aquesta paret ha estat visitada, llavors, crea un camí entre ambdues i marca la cel·la no visitada com a part del laberint, afegeix les parets al conjunt de veïns i elimina la paret que separava les cel·les del conjunt de veïns.

## 2. Estat de l'art

### 2.2.2.2. Algorisme d'arbre binari

Aquest algorisme destaca per la seva simplicitat i facilitat per ser implementat, ja que per ser implementat només cal fer el següent:

- Per cada cel·la de la quadrícula, es crea un camí de forma aleatòria o bé sud o bé est.

Cal notar que un laberint generat amb aquest mètode no ens assegura que totes les caselles estiguin connectades entre si i és possible que es creïn "habitacions" tancades.

### 2.2.3. Objectes, models i animacions 3D

Posat que no es tracta d'un projecte de disseny gràfic i que tampoc es tenen les habilitats ni el temps per crear els diferents models i objectes i animar-los en cas que sigui necessari, es farà ús de contingut generat per tercers.

#### 2.2.3.1. Mixamo

És un servei propietat d'Adobe que ofereix eines i recursos per a l'animació i *rigging* de personatges en 3D. És especialment valuós per desenvolupadors que necessiten models animats, però manquen els coneixements tècnics per crear-los ells mateixos.

Mixamo (vegeu Figura 6) [5] proporciona una gran biblioteca de models 3D per elegir als quals se'ls poden incorporar animacions.

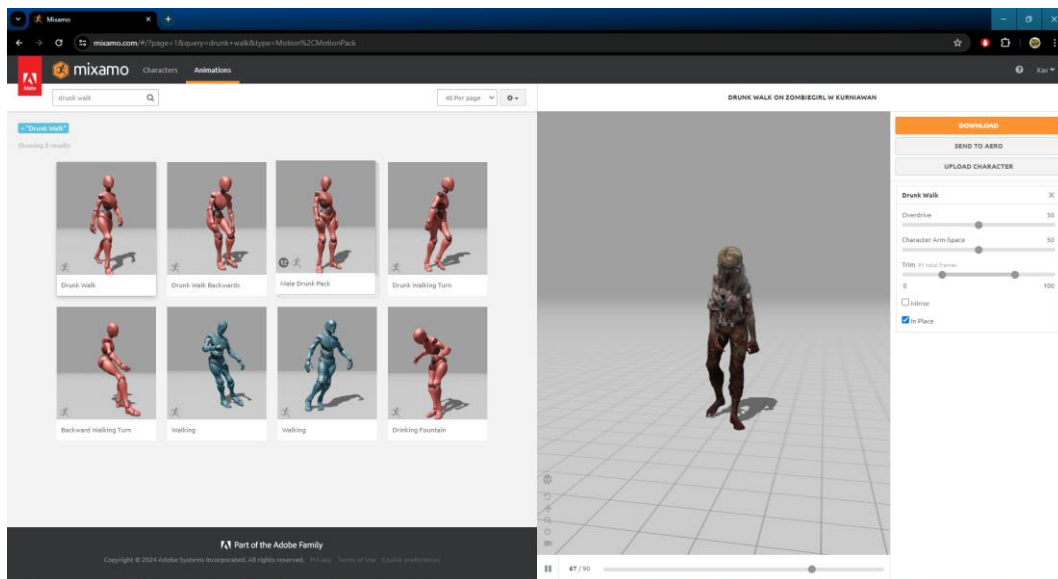


Figura 6: Mixamo

#### 2.2.3.2. Sketchfab

Sketchfab (vegeu Figura 7) [6] és una plataforma que permet als usuaris carregar, visualitzar i compartir els seus models 3D. D'aquesta manera, altres usuaris poden emprar aquests models pels seus projectes sigui de forma gratuïta o bé, comprant la llicència d'ús segons ho hagi especificat l'autor.

## 2.2. Eines que poden fer-se servir per resoldre aquest problema. Anàlisi i selecció.

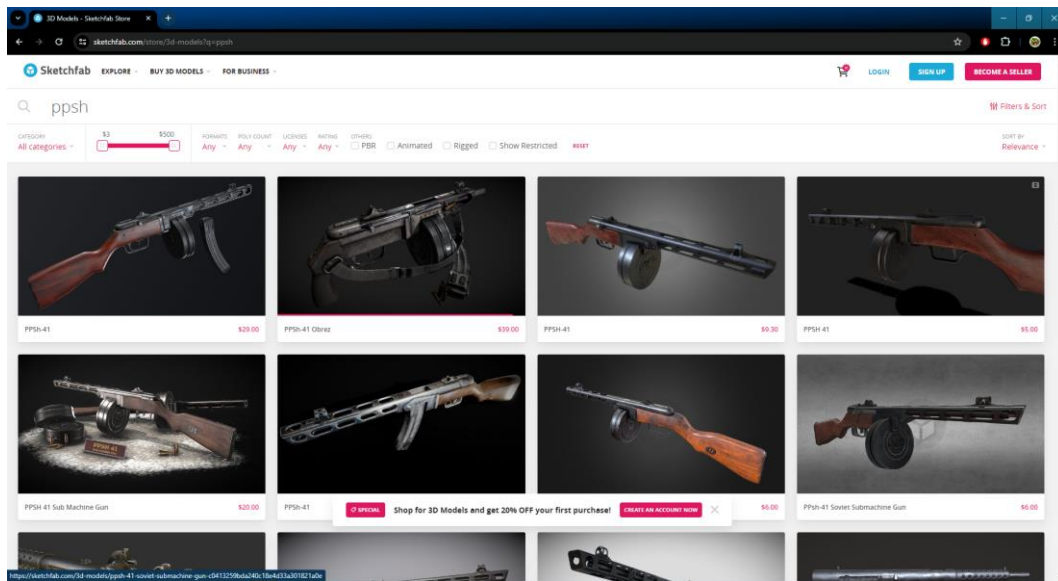


Figura 7: Sketchfab

### 2.2.4. Selecció

Una vegada vist els principals problemes que presenta el projecte i les eines disponibles per resoldre-los, decidim quines emprarem.

#### 2.2.4.1. Kit de desenvolupament de jocs en RV

S'ha decidit finalment emprar A-Frame per diversos motius:

- És l'eina líder en l'àmbit de web XR ,per tant, hi ha molta més comunitat que es tradueix en més documentació, articles i llibreries fetes per la comunitat.
- Altres companys que també estan fent el TFG amb aplicacions similars i fan servir A-Frame per el que és més senzill ajudar-nos entre nosaltres.
- El mateix Wonderland Engine reconeix que A-Frame és la millor eina en un article [6].

#### 2.2.4.2. Generació de laberints

S'ha emprat l'algorisme de Prim perquè aquest és molt senzill d'implementar en comparació a altres algorismes i, sobretot, els laberints generats d'aquesta manera tenen unes propietats que ens resultaran molt útils:

- A diferència d'altres algorismes, l'algorisme de Prim no crea zones aïllades. És a dir, no genera espais tancats que no són accessibles de cap costat.
- Totes les parets estan connectades. És a dir, suposem que ens situam a la casella superior esquerra del laberint. Posam la mà tocant la paret i caminam fora separant la mà de la paret. Havent passat prou temps, tornariem al punt de partida i hauríem tocat totes les parets.

Això ens serà útil perquè podrem representar el laberint com un polígon irregular de 'n' vèrtexs i així definir la zona per la qual ens podem moure.

## 2. Estat de l'art

---

### *2.2.4.3. Objectes, models i animacions 3D*

S'emprarà Mixamo per obtenir models animats i Sketchfab per altres objectes que siguin necessaris

## Capítol 3. Desenvolupament del projecte

Aquest capítol detalla els aspectes més importants de la fase de desenvolupament. Primerament es detallen aspectes relacionats amb la gestió del projecte, després una anàlisi dels requisits seguit de la implementació i les proves realitzades.

### 3.1. Gestió del projecte

Aquest primer apartat descriu la gestió del desenvolupament del projecte. S'explicarà la metodologia de desenvolupament que s'ha seguit i la planificació temporal que s'ha fet.

#### 3.1.1. Metodologia de desenvolupament

En el desenvolupament d'aquest projecte s'ha utilitzat una metodologia basada en el desenvolupament incremental per prototipus. La idea darrera aquest mètode és desenvolupar un projecte a través de la creació de successius prototipus funcionals amb l'objectiu de millorar i refinar el producte final.

Aquesta metodologia es basa principalment en els principis d'iteració i millora progressiva (en lloc de desenvolupar tot el sistema de cop, el projecte es divideix en petites iteracions que cadascuna produirà un prototipus), flexibilitat pels canvis (com el sistema es construeix gradualment, és més senzill realitzar ajustaments i incorporar noves funcionalitats) i validació primerenca (amb prototipus funcionals es pot validar i verificar la solució abans d'arribar a la versió final).

#### 3.1.2. Planificació temporal

La planificació d'aquest projecte s'ha dividit per mesos contemplant l'any acadèmic. Com es pot veure a la Figura 8, els primers mesos no hi haurà gaires avenços perquè hi ha altres tasques alienes a aquest projecte que hi interfereixen. La major part de la feina s'haurà fet entre febrer i maig.

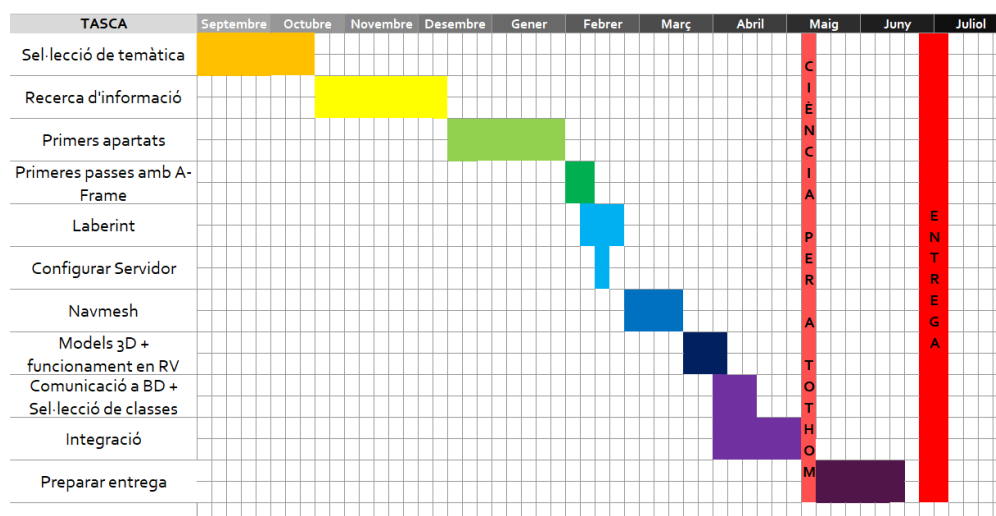


Figura 8: Cronograma

### 3. Desenvolupament del projecte

---

Cal notar com hi ha una primera fita a principis de maig que ens forçarà a tenir una versió funcional de l'aplicació per poder presentar-la a la fira *Ciència per a tothom* [7].

#### 3.2. Anàlisi

Abans d'entrar en el disseny i la implementació de la solució, cal fer una anàlisi que tracti temes com els diferents usuaris i els requisits tant funcionals com no funcionals de l'aplicació.

##### 3.2.1. Usuaris

L'aplicació compta amb un sol tipus d'actor: el jugador. Aquest, de manera trivial, juga independentment de la plataforma que faci servir per fer-ho. És a dir, un usuari que accedeixi des d'un PC és un jugador de la mateixa manera que un altre usuari que hi accedeixi amb equipament VR.

##### 3.2.2. Requisits d'usuari

Això són els requisits finals després d'haver fet totes les iteracions del projecte. S'han dividit en requisits funcionals i requisits no funcionals.

###### 3.2.2.1. Requisits funcionals

- RF01 - Disparar
  - Un jugador ha de poder disparar.
- RF02 - Moviment
  - El jugador s'ha de poder moure lliurement per l'escena.
- RF03 - Inici de joc
  - El jugador ha de poder iniciar una partida.
- RF04 - Classe
  - El jugador ha de poder triar la classe i arma que prefereixi.
- RF05 - Llista de marcadors
  - El jugador ha de poder consultar el resultat del top 10 millors partides
- RF06 - Game Over
  - El jugador ha de poder iniciar una nova partida quan sigui abatut.

###### 3.2.2.2. Requisits no funcionals

- RNF01 - Taula de puntuacions
  - La taula de puntuacions es limitarà al top 10 millors resultats obtinguts pels jugadors.
- RNF02 - Generació de nivells
  - El mapa de cada partida es generarà de forma procedimental i aleatòria. De manera que cada partida sigui diferent.
- RNF03 - Armes en RV
  - Quan es jugui en RV, les armes han de funcionar com si realment es tingués una arma a la mà. Això implica que si es dispara, l'origen i direcció del tret dependran de la posició i orientació de la mà que tengui l'arma.
- RNF04 - Col·lisions
  - Ni el jugador ni els enemics poden travessar les parets. Si aquests s'hi apropen molt haurien de col·lisionar i no poder continuar avançant en aquella direcció.

- RNF05 - Identificació
  - El jugador s'ha d'identificar amb un nom o àlies per poder iniciar la partida.
- RNF06 - Compatibilitat de plataformes
  - El joc ha de ser jugable tant amb equipament de RV com a través d'un PC tradicional.
- RNF07 - Controls
  - El joc s'ha de poder manejar amb comandaments de videoconsola, comandaments de RV, teclat i ratolí.
- RNF08 - Tutorial
  - Abans d'anar al joc, es mostrarà una pantalla que expliqui breument com s'hi juga.
- RNF09 - RV amb 6-DoF
  - El joc ha de ser jugable amb dispositius de RV amb 6-DoF.
- RNF10 - Generació de nivells
  - El mapa de cada partida es generarà de forma procedimental i aleatòria. De manera que cada partida sigui diferent.
- RNF11 - Efectes de so
  - El joc comptarà amb efectes de so que es reproduiran quan es realitzin certes accions com disparar, ser ferit, ferir un zombi, etc.

### 3.3. Disseny

El projecte necessita una fase de disseny que té com a objectiu definir i presentar el model de dades, el disseny arquitectònic i el disseny gràfic. Aquests elements seran les entrades per a la fase d'implementació.

#### 3.3.1. Model de dades

Com l'únic que cal guardar a la base de dades és la informació sobre les partides que altres jugadors han fet, el model de dades resulta molt simple. Una sola taula amb els quatre atributs que fan falta per descriure la puntuació obtinguda en la partida és tot el que necessitam.

Leaderboards	
playerName	varchar
round	integer
kills	integer
headshotKills	integer

Figura 9: Model de dades

### 3. Desenvolupament del projecte

---

- **playerName**: El nom a través del qual el jugador ha decidit identificar-se quan ha començat la partida.
- **round**: La ronda fins a la qual ha arribat el jugador.
- **kills**: Quantitat total d'enemics que ha eliminat el jugador.
- **headshotKills**: Quantitat total d'enemics que el jugador ha eliminat disparant al cap. Aquest camp sempre serà menor o igual que **kills**.

Cal notar com cap d'aquests camps (o combinacions d'aquests) són adequats per convertir-se en clau primària de la taula. En canvi, es definirà un camp *id* autoincremental que servirà com clau primària. Per més informació, vegeu 3.4.1 Base de dades.

#### 3.3.2. Disseny arquitectònic

En aquest apartat, es presenta l'arquitectura del sistema que s'ha implementat seguint un model client-servidor. Aquest model divideix el sistema en dues parts principals: el client, que és la interfície a través de la qual l'usuari accedeix al sistema i, el servidor, és qui dona servei a les peticions dels clients tal i com mostra la Figura 10.

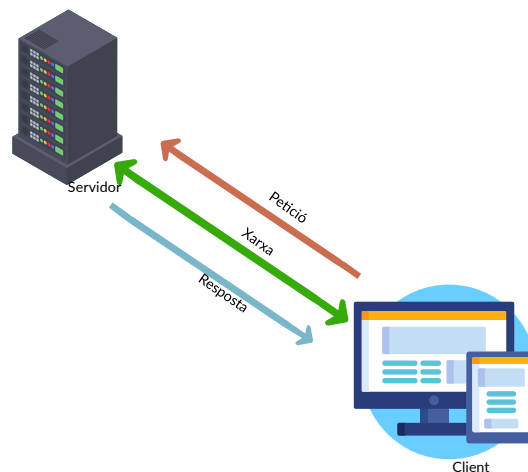


Figura 10: Model client-servidor

Un sistema que implementa aquest model, presenta el següent esquema de funcionament:

1. El client sol·licita recursos al servidor.
2. La petició es rebuda pel servidor.
3. La petició es processada.
4. El servidor envia al client el resultat.
5. El client processa el resultat.

La Figura 11 mostra com s'implementarà el model descrit anteriorment en aquest projecte amb més detall.



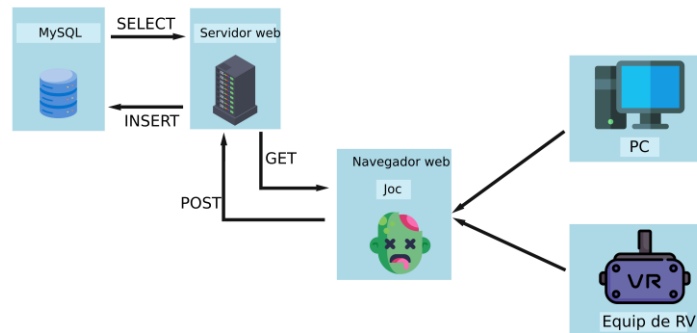


Figura 11: Model Arquitectònic

### 3.3.3. Disseny gràfic

Aquesta secció està dedicada a descriure disseny gràfic realitzat per construir una atmosfera de terror.

#### 3.3.3.1. Ambient

Per crear un ambient de terror, el fonamental és l'escenari i és jugar amb la il·luminació, especialment amb l'absència d'aquesta així com amb l'ambientació geogràfica de l'escenari (per exemple: una presó, un manicomi, una fàbrica, etc.). La Figura 12 diferents exemples sobre com la saga *Call of Duty: Black Ops – Zombies* aconsegueix crear aquest efecte.

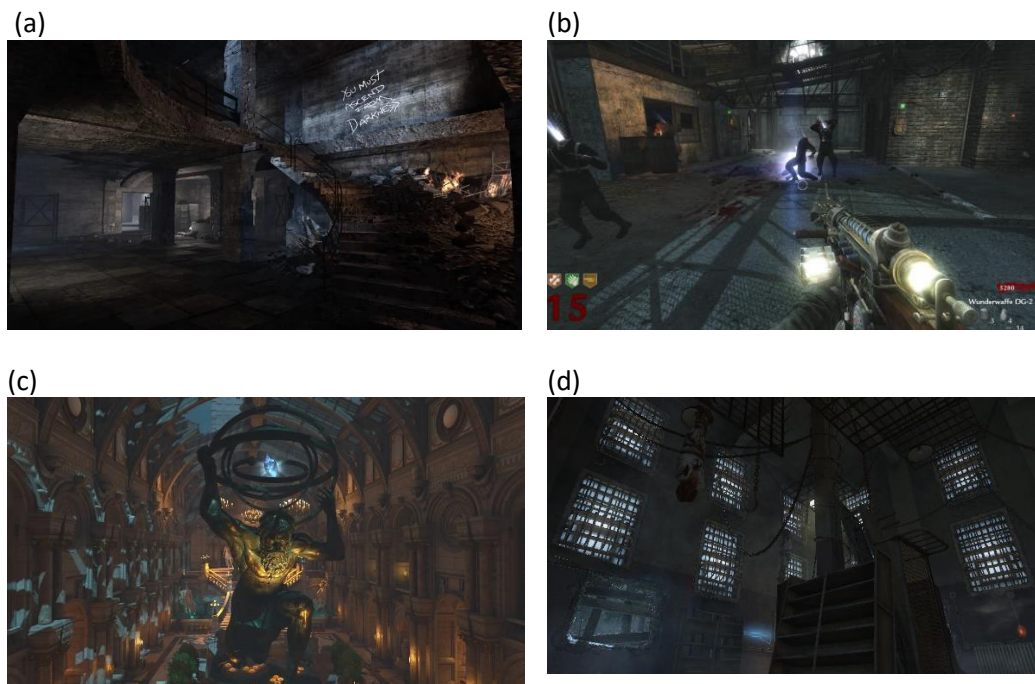


Figura 12: Escenaris de *Call of Duty: Black Ops – Zombies* (a) *Nacht der Untoten*, (b) *Der Riese*, (c) *Dead of the Night*, (d) *Mob of the Dead*

### 3. Desenvolupament del projecte

En el nostre cas, farem servir la component *a-frame-environment* [8] que ens ajudarà a definir el cel, terra, quantitat d'il·luminació i direcció d'aquesta en la nostra escena. També farem servir una textura fosca per les parets del laberint que acompanyarà a la foscor general de l'escena.

#### 3.3.3.2. Armament

Respecte als models de les armes que s'han escollit, s'han seleccionat armes icòniques per la seva importància històrica Figura 13 (a) i (b), la *Raygun* Figura 13 (c) (una pistola de raigs emblemàtica de la saga *Call of Duty: Black Ops – Zombies*) i, addicionalment, s'ha afegit un moix-pistola Figura 13 (d) amb intencions humorístiques.

(a)



(b)



(c)



(d)

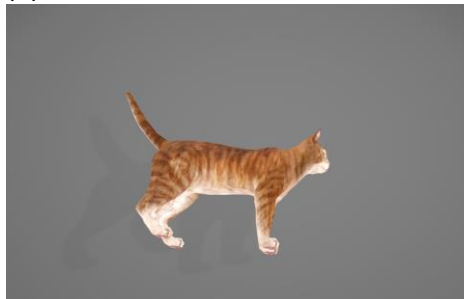


Figura 13: Models d'armes (a) *Colt M1911* [9], (b) *PPSH* [10], (c) *Raygun* [11], (d) Moix-pistola [12]

#### 3.3.3.3. Zombis

Sobre els models dels zombis, s'han elegit tres models diferents que els zombis poden adoptar en el joc (vegeu Figura 14).



Figura 14: Models de zombi

Malauradament, per problemes de rendiment amb el primer i tercer model, s'ha acabat emprant únicament el model del zombi dona, ja que aquest no era tan pesat.

#### 3.3.4. Cicle de joc

Una vegada el jugador iniciï el joc, seguirà un a sèrie de passes per començar una partida fins a acabar-la.

Per començar, el jugador no podrà iniciar la partida fins que s'hagi identificat amb un àlies i hagi seleccionat la classe amb la que vol jugar. Una vegada hagi complert aquests dos requisits, podrà prémer el botó d'iniciar partida i la partida iniciarà. Una vegada el jugador sigui abatut, serà transportat al vestíbul post-partida on se li donarà la opció d'iniciar una nova partida. Si així ho decideix, el cicle tornarà a començar. La Figura 15 mostra el diagrama de flux que el jugador seguirà.

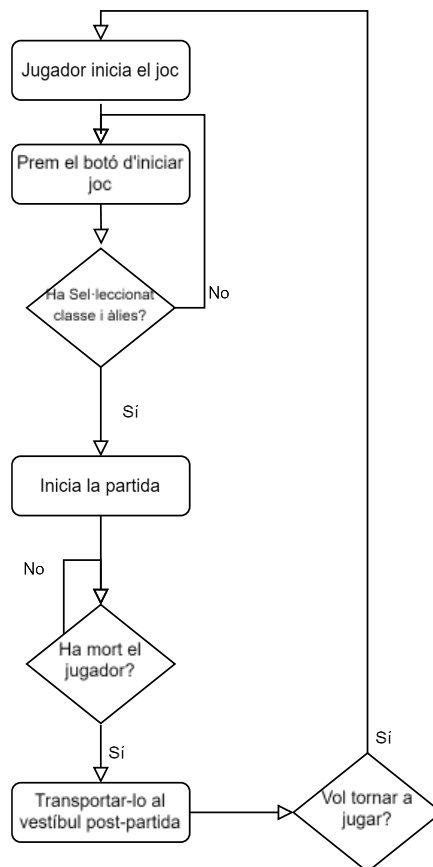


Figura 15: Diagrama de flux del joc

#### 3.4. Implementació

En aquest apartat es descriu la implementació realitzada. S'ha dividit en tres subapartats (un per cada entitat definida): 3.4.1 Base de dades, 3.4.2 Servidor i 3.4.3 Client. Cada subapartat es desglossa en els diferents mòduls que conformen cada entitat.

### 3. Desenvolupament del projecte

---

#### 3.4.1. Base de dades

Com s'ha esmentat anteriorment, per aquest només necessitam una taula per emmagatzemar els resultats de les partides. Per tant, la implementació és senzilla (vegeu Codi 1)

```
CREATE DATABASE tfg;
USE tfg;

CREATE TABLE `leaderboards` (
  `playerName` varchar(20) DEFAULT NULL,
  `round` int DEFAULT NULL,
  `kills` int DEFAULT NULL,
  `headshotKills` int DEFAULT NULL,
  `id` int NOT NULL AUTO_INCREMENT,
  PRIMARY KEY (`id`)
)

ALTER USER [USER] IDENTIFIED WITH mysql_native_password BY [PASSWORD];
FLUSH PRIVILEGES;
```

Codi 1: Implementació SQL de la base de dades

Tot i que la implementació d'aquesta entitat és molt senzilla, hi ha dos detalls dignes de menció:

- La clau primària de la taula és un autoincremental. És així perquè no tendria sentit que cap de les altres columnes o combinacions d'aquestes ho fossin ja que, per exemple, un jugador podria jugar més d'una partida amb el mateix nom, a la mateixa ronda, etc.
- Les darreres dues comandes ens serveixen per establir el mode d'autenticació per accedir a la base de dades. Són necessàries perquè per defecte no ens deixa accedir-hi. Això és així perquè MySQL 8 per defecte empra el mètode d'autenticació *caching\_sha2\_password* però mysqljs (el paquet de Node.js que es fa servir per establir una connexió amb una base de dades) no soporta aquest mètode. [13]

#### 3.4.2. Servidor

S'ha emprat Node.js per crear la part del servidor. Com s'hi han hagut de muntar dos servidors: un accessible a través d'internet ([zombis.ltim.uib.es](http://zombis.ltim.uib.es)) i un en local per poder donar accés durant la fira *Ciència per a tothom*, s'han fet servir les variables d'entorn per configurar el servidor depenent d'en quin entorn es trobi fora necessitat de tenir un fitxer per cada entorn en el qual vulguem arrancar el servidor.

##### 3.4.2.1. Creació del servidor

S'ha fet servir l'entorn de treball Express.js [14] juntament amb les llibreries HTTP [15] i HTTPS [16] de Node.js per desenvolupar la gestió de sol·licituds, respostes i rutes; i middlewares. El Codi 2 mostra com s'ha implementat.

```
// Instancia d'express
const app = express();

// Middlewares
```

```

app.use(express.json());

var options = {
  key: fs.readFileSync('[key]'),
  cert: fs.readFileSync('[cert]')
};

// Block petitions to some files
app.use((req, res, next) => {
  if (config.blockedPaths.includes(req.path)) {
    // Send 404 Not Found response
    res.status(404).send('Not Found');
  } else {
    next();
  }
});

app.use('/', express.static(path.join(__dirname, '')));

// Create an HTTP service.
http.createServer(app).listen(80);
// Create an HTTPS service identical to the HTTP service.
https.createServer(options, app).listen(443);

```

Codi 2: Creació del servidor

Cal notar com hem bloquejat accés a alguns fitxers.

#### 3.4.2.2. Variables d'entorn

Les variables d'entorn són parells clau-valor emmagatzemats dins el sistema operatiu i poden ser utilitzades per programes per obtenir informació sobre l'entorn en què s'estan executant canviant així el curs d'execució si fos necessari.

Hem emprat aquesta eina per crear un sol programa servidor i que es comporti de forma diferent depenent de l'entorn en què es troba. Per fer-ho, hem creat un fitxer de configuració (vegeu Codi 3) que establirà la configuració que ha de tenir el servidor en funció del valor de la variable d'entorn que hem creat anteriorment.

```

var config = {
  port: 80,
  db: {
    host: ''
    , user: ''
    , password: ''
    , database: ''
  } , blockedPaths: ['[blockedPaths]']
}
if (process.env.NODE_ENV === "production") {
  config.db.host = '[host]',
  config.db.user = '[user]',
  config.db.password = '[password]',
  config.db.database = '[database]'
}else if (process.env.NODE_ENV=="fira"){
  config.db.host = '[host]',

```

### 3. Desenvolupament del projecte

```
    config.db.user = '[user]',
    config.db.password = '[password]',
    config.db.database = '[database]'
  }
  else {
    config.db.host = '[host]',
    config.db.user = '[user]',
    config.db.password = '[password]',
    config.db.database = '[database]'
  }
  module.exports = config;
```

Codi 3: Configuració del servidor

Finalment, només cal incloure aquest fitxer de configuració al nostre programa servidor i emprar-lo. El Codi 4 mostra resumidament una manera de fer-ho.

```
const config = require('./config.js')
// Database connection
const db = mysql.createConnection({
  host: config.db.host,
  user: config.db.user,
  password: config.db.password,
  database: config.db.database
});
```

Codi 4: Servidor emprant fitxer de configuració

El principal avantatge de fer-ho d'aquesta manera és que, com podem veure en el Codi 4, no codifiquem en dur res del servidor sinó que tot depèn del fitxer de configuració. Resultant en un codi més llegible i versàtil.

#### 3.4.2.3. Accés a base de dades

El servidor també s'encarrega de fer d'intermediari entre el client i la base de dades. De manera que, quan el client envii el resultat de la seva partida, o bé, consulti la taula de puntuacions ho farà a través del servidor. Per fer-ho, s'ha emprat el mòdul de MySQL [17] de Node.js. El Codi 5 especifica com s'ha fet:

```
// Client posts their result
app.post('/gameOver', (req, res) => {
  let sql = `INSERT INTO leaderboards [...]`;
  db.query(sql, (err, result) => {
    if (err) throw err;
    console.log('Data saved successfully');
    res.json({ message: 'Data saved successfully' });
  });
});
// Get top 10 from leaderboards
app.get('/leaderboards', (req, res) => {
  db.query(`SELECT [...]`, (err, results) => {
    if (err) {
      console.error('Error en la consulta 1:', err);
      res.status(500).json({ error: 'Error en la consulta 1' });
    } else {
      res.json(results);
    }
  });
});
```

```

    }
  });
});

```

Codi 5: Servidor. Accés a base de dades

D'aquesta manera, el client pot interactuar amb la base de dades simplement fent peticions al servidor. També s'ha de dir que el client no pot fer vulgui amb la base de dades. Només pot realitzar aquestes dues interaccions que hem definit.

### 3.4.3. Client

Aquesta secció explica els detalls de la implementació en la part del client. Com en aquesta entitat és centra la gran part de la implementació, s'ha dividit en diverses subseccions. La mostra de forma esquemàtica les diferents parts que formen aquesta secció.

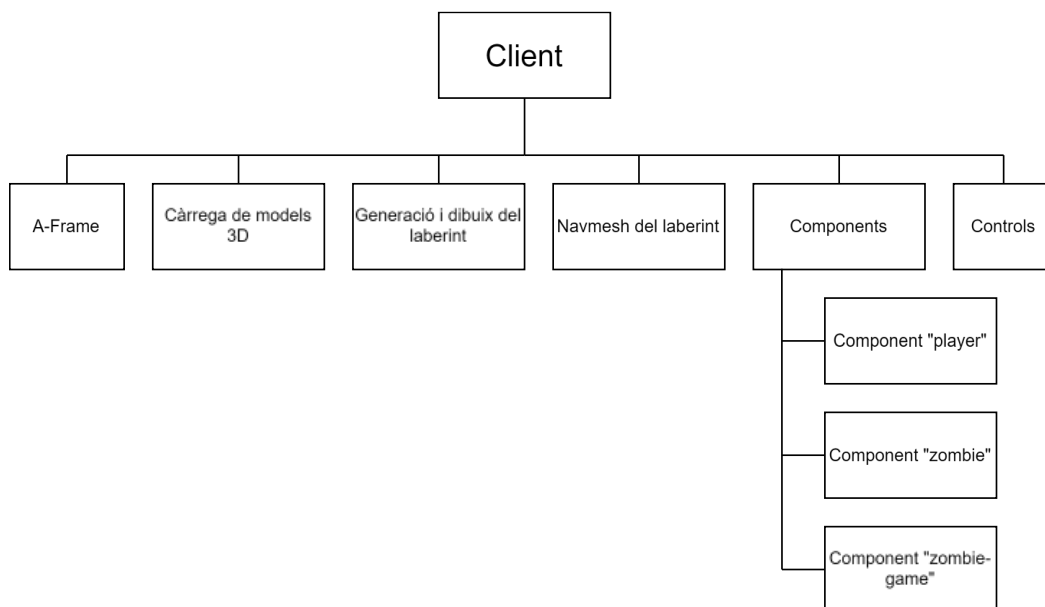


Figura 16: Esquema d'implementació Client

#### 3.4.3.1. A-Frame

És un *framework* construït damunt HTML que facilita la creació d'escenes en RV i és el que hem emprat per a la realització d'aquest projecte. Per emprar-lo, només cal afegir-lo en el nostre document HTML.

A-Frame empra etiquetes com les que empra HTML per representar els objectes de l'escena. Per exemple, una escena simple podria ser implementada com mostra el Codi 6.

```

<a-scene>
  <a-box position="0 1.5 -5" rotation="0 45 0"></a-box>
  <a-sphere position="-2 1.25 -5" radius="1.25"></a-sphere>
  <a-plane position="0 0 0" width="10" height="10"></a-plane>
</a-scene>

```

Codi 6: A-Frame. Escena simple

### 3. Desenvolupament del projecte

#### 3.4.3.2. Càrrega de models 3D

Per carregar models 3D amb A-Frame, es pot fer servir el component *gltf-model*. Això ens permet carregar objectes com les armes. Ara bé, per defecte, A-Frame no té una manera de carregar models 3D amb animacions de manera que, en el nostre cas, si volem crear un món més inmersiu on, per exemple, quan un zombi es mogui, tenguim una animació de caminar o que quan ataquem al jugador, en tenguim una d'atac, etc. aquesta forma de carregar els models no ens serà suficient.

És per això que ens serà necessari l'ús d'*a-frame-extras* [18], una llibreria que incorpora un conjunt de components, primitives i sistemes addicionals que no venen per defecte a A-Frame. D'aquesta manera, incloure models animats es redueix al que descriu el Codi 7.

```
<a-entity gltf-model="#zombie_model" animation-mixer="clip:Punch"
position="3 0 0"></a-obj-model>
```

Codi 7: A-Frame. Models animats

Cal notar com dins un sol model podem tenir més d'una animació i anar canviant la animació dinàmicament. Per combinar múltiples animacions en un sol fitxer .glb ens podem ajudar d'eines com *Blender* [19]. La Figura 17 mostra com s'han fusionat diverses animacions dins un sol fitxer .glb.

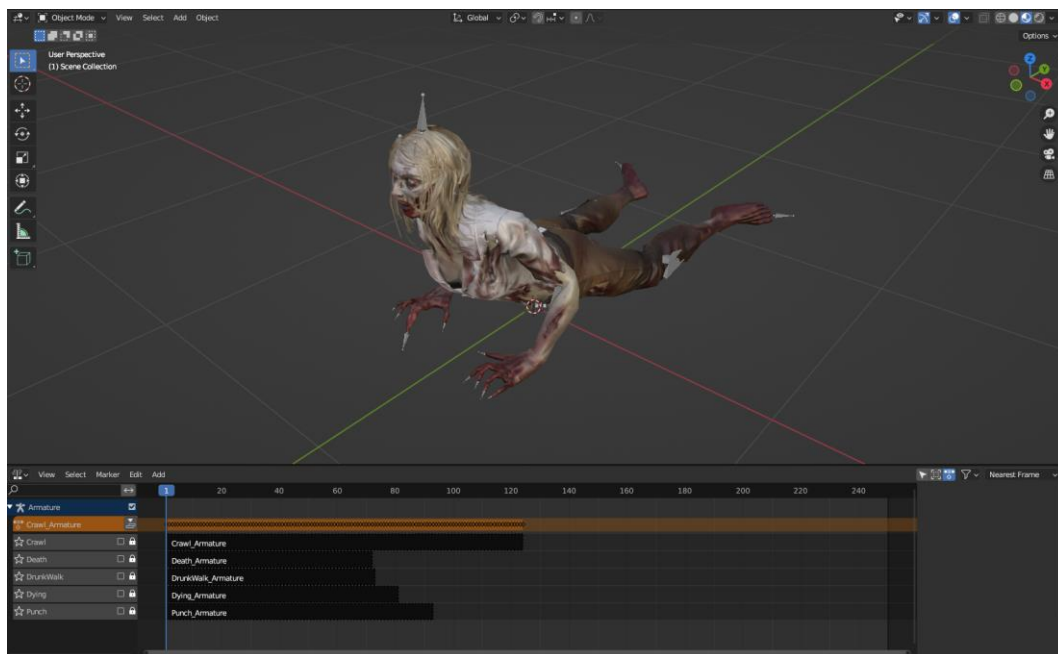


Figura 17: Animacions amb Blender

Si ens fixam en el menú inferior, podem veure com l'objecte amb el qual estam treballant té cinc animacions (*Crawl*, *Death*, *DrunkWalk*, *Dying* i *Punch*) associades a ell.

A-Frame [3] té la peculiaritat que cada vegada que elimina un objecte de l'escena, no allibera la memòria que ocupava aquest. Això és així perquè Three.js [20], la biblioteca sobre la qual A-Frame [3] es sustenta, no ho fa. Això no suposa un gran problema si feim servir objectes simples ja que aquests ocupen pocs bytes en memòria. Ara bé, emprant objectes més pesats, com ara podria ser un model 3D d'un zombi, pot arribar a causar un



“memory leak”. A-Frame [3] és conscient d’aquesta problemàtica [21] i ells mateixos aporten una solució (vegeu Apèndixs>Eliminar model 3D de memòria) que consisteix en sobreesciure la component *glTF-model* per tal de canviar el funcionament del mètode *remove()*, encarregat d’eliminar de l’escena un objecte. El que fa resumidament és, recórrer tots els nodes tant de malla com de textura que conformen el model 3D i els allibera un a un i, finalment, elimina les referències del renderitzador.

#### 3.4.3.3. Generació i dibuix del laberint

Per representar el laberint, es fa servir una matriu. Cada casella té 4 caselles veïnes (nord, est, sud i oest) i 4 parets que la tanquen. Si existeix camí entre dues caselles, llavors, la paret que les separa haurà desaparegut.

L’algorisme fet servir per generar laberints és l’algorisme de Prim. El Codi 8 mostra una simplificació del que s’ha implementat. Si voleu veure la implementació completa, aneu a Apèndixs>Algorisme de Prim per Generar laberints.

```
var laberint = inicialitzar_matriu_amb_totes_les_parets();
var visited = new Set();
var frontier = new Set();

var start = laberint [0][0];

visited.add(start);
// Add all unvisited cells that are adjacent to the current cell to
the frontier set
add_frontiers(start);

while (frontier.size > 0) {
  let current = frontier.getRandomCell();

  frontier.delete(current);
  visited.add(current);
  current.visited = true;
  // Remove the wall between the current cell and a random adjacent cell
  that
  // belongs to the visited set.
  remove_random_wall(current);
  add_frontiers(current);
}
```

Codi 8: Algorisme de Prim per generar laberints

Una vegada s’ha generat el laberint, dibuixar-lo a l’escena només suposa recórrer la matriu cel·la per cel·la i dibuixar les parets corresponents.

#### 3.4.3.4. Navmesh del laberint

Per definir la zona navegable, es construeix una malla de navegació (*navmesh*) l’àrea de la qual serà tota la superfície a la qual restringirem el moviment tant del jugador com dels zombis. La definirem de manera que si hi ha una paret, no hi haurà *navmesh* i el jugador (o el zombi) no podrà continuar caminant en aquella direcció. Crear aquesta malla és clau tant per restringir el moviment del jugador, és a dir, impedir que aquest travessi parets, com pel *pathfinding* dels zombis (vegeu 3.4.3.5.2 Component “zombie”).

### 3. Desenvolupament del projecte

---

Com el laberint és diferent en cada partida que es juga, no es pot emprar sempre la mateixa *navmesh* sinó que aquesta s'haurà de generar cada vegada.

Per solucionar aquest problema s'ha escrit un algorisme que recorr tot el laberint i genera un polígon irregular de tants vèrtexs com siguin necessari per fer-lo servir com *navmesh*. Cal notar que s'ha pogut fer així perquè l'algorisme que s'ha fet servir per generar el laberint ho permet, doncs tots els laberints generats amb l'algorisme de Prim tenen la següent propietat: Suposem que ens situam a una casella arbitrària. Posam la mà a la paret i caminam seguint la paret fora desferrant la mà. Donat suficient temps, haurem recorregut tot el laberint havent i haurem fregat la mà per totes les parets. Aquesta propietat es coneix com la *regla de la mà dreta/esquerra* [22].

En termes de teoria de grafs, podem representar el nostre laberint com  $G = (V, E)$  on  $V$  és el conjunt de caselles i  $E$  és el conjunt de passadissos entre caselles. La connexitat de  $G$  implica que partint de qualsevol node  $v_0$  puc arribar a qualsevol altre node  $v$ .

$$\forall u, v \in V, \exists (u = v_0, v_1, v_2, \dots, v_k = v) \text{ tals que } \forall i \in \{0, 1, 2 \dots k - 1\}, (v_i, v_{i+1}) \in E$$

Emprant aquesta propietat, hem escrit un algorisme que fa això mateix. Comença al cantó superior esquerre i segueix la paret. Quan es troba que ha de fer un canvi de direcció, col·loca un vèrtex i continua avançant. Una vegada arribi al punt de partida, obtindrem un polígon irregular que emprem com *navmesh*.

A continuació es mostra el resultat amb un laberint senzill (Figura 18):

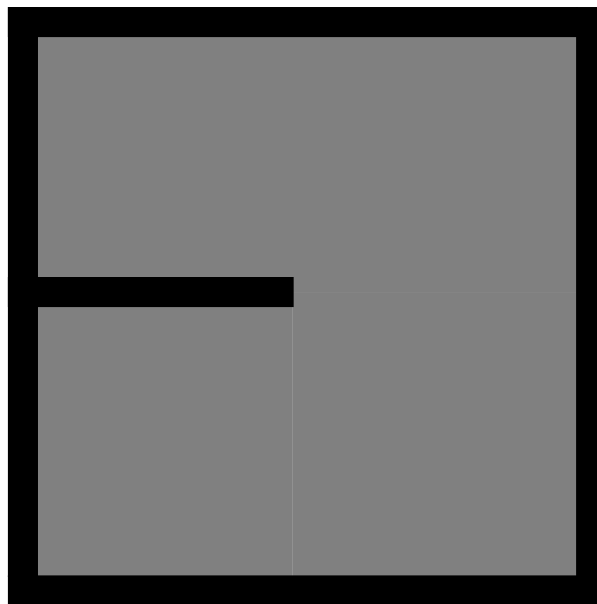


Figura 18: Laberint senzill

1. Començam al cantó superior esquerre Figura 19(a).
2. En trobar un canvi de direcció, col·loca un vèrtex Figura 19(b).
3. Continua fins arribar al punt de partida Figura 19(c).
4. Connecta els vèrtexs en l'ordre que s'han visitat i s'obté la *navmesh* Figura 19(d).

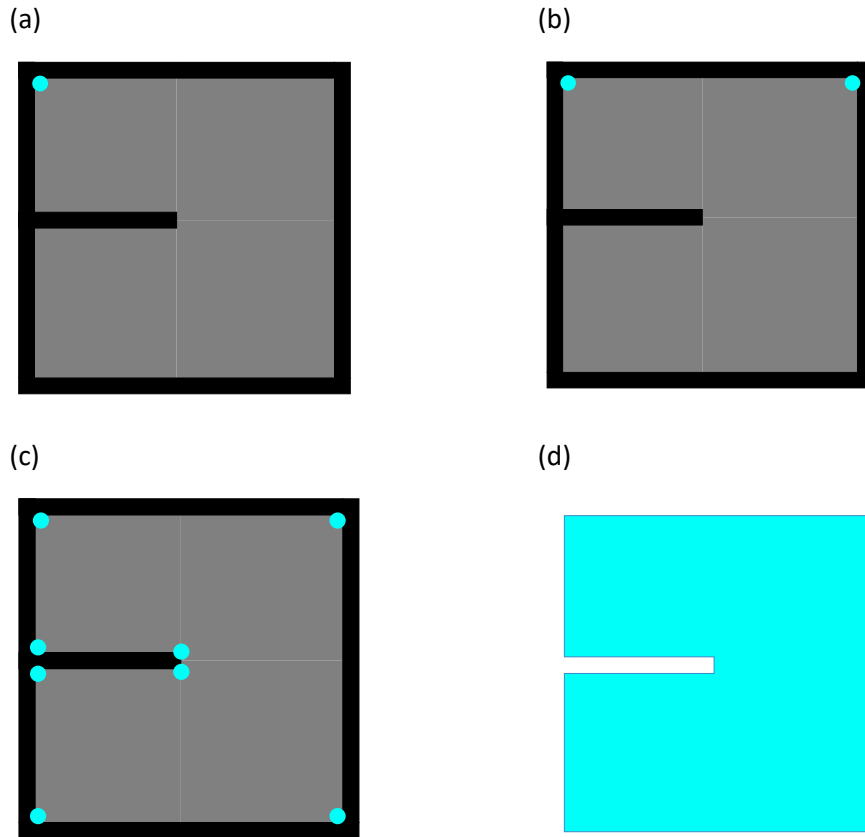


Figura 19: Laberint senzill. Navmesh (a), (b), (c) i (d)

Si es vol consultar la implementació d'aquest algorisme en Javascript, aneu a [Apèndixs>Generació de navmesh](#)

#### 3.4.3.5. Components

Un component de A-Frame és un fragment de codi reusable i modular que s'afegeix a una entitat per afegir-hi una aparença, comportament o funcionalitat. Per exemple, la component *gltf-model* de la qual hem xerrat anteriorment, ens permet carregar un model gltf o la component *animation-mixer* que ens permet tenir models animats a la nostra escena.

A-Frame també permet al desenvolupador crear les seves pròpies components. Per aquest projecte, s'han creat tres components clau pel funcionament del joc: *Player*, *zombie* i *zombie-game*.

##### 3.4.3.5.1. Component "player"

S'encarrega principalment de gestionar l'estat del jugador. Monitora quanta munició i punts de vida li manquen i actuar de manera diferent depenent d'aquests valors. És a dir, si el jugador té menys de 0 punts de vida, llavors envia una notificació al sistema que aquest ha mort. De la mateixa manera que si s'ha quedat sense munició, no podrà disparar.

### 3. Desenvolupament del projecte

#### 3.4.3.5.2. Component “zombie”

Defineix completament un zombi i implementa totes les funcionalitats d’aquest. És a dir: s’encarrega que les entitats amb aquesta component tinguin la geometria que definirà un zombi (model 3D i *hitboxes*. Vegeu Figura 20) i inclou les funcionalitats de *pathfinding*, interacció amb el jugador, eliminació de l’entitat i monitora dels punts de vida del mateix zombi així com enviar una notificació al sistema quan aquest mor.



Figura 20: Zombi amb *hitboxes*

Per la funcionalitat de *pathfinding* s’ha fet servir el que ja implementa la llibreria *a-frame-extras*. Per fer-lo servir es necessiten dues coses: *navmesh* i destinació final. Per la *navmesh*, es pot fer servir la mateixa que empram per restringir el moviment del jugador (vegeu 3.4.3.4 Navmesh del laberint). Per la destinació final farem servir la posició del jugador ja que volem que els zombis el persegueixin (vegeu Codi 9). Ara bé, com el jugador es pot moure lliurement, haurem d’anar actualitzant periòdicament aquesta destinació final a fi que els zombis sempre el puguin trobar. Tot i que pot parèixer que el fet d’anar recalculant el camí cada poc temps pot causar greus problemes de rendiment, en la pràctica no és així, ja que l’algorisme que s’empra és poc pesat. Així i tot, s’han pres mesures per mitigar aquest possible risc i és que els zombis recalculen el camí en instants de temps diferents de manera que no ho fan tots alhora.

```
AFRAME.registerComponent('zombie', {
  schema: {
    [...],
    pathfindProcess: { type: 'number' },
  },
  init: function () {
    [...][...][...]
    // Zombie Pathfinding
    this.el.setAttribute('nav-agent');
    this.el.addEventListener('navigation-end', () =>
    this.hitPlayer(this.el));
    this.pathfindProcess = setInterval(this.pathfind,
    cfg.ZOMBIE_PATHFIND_TIME, this.el);
```

```

    },
    pathfind: function (element) {
        [...];
        element.setAttribute('nav-agent', {
            active: true,
            destination: cameraRig.getAttribute('position')
        });
    },
},

```

Codi 9: Pathfinding periòdic

Per la interacció amb el jugador, es mira la distància entre el zombi i el jugador. Si aquesta és menor a cert valor llindar, el zombi ataca el jugador (vegeu Codi 10).

```

// Enemies will be able to attack the player once they have reach
their destination
hitPlayer: function (element) {
    // If the enemy is close to the player, then punch them
    if
(element.getAttribute('position').distanceTo(cameraRig.getAttribute('p
osition')) < 1.5) {
        element.querySelector('.model').setAttribute('animation-
mixer', 'clip:Punch; timeScale:1.5;
loop:once;clampWhenFinished:true;');
        cameraRig.components['player'].getHit();
    }
    // However if the enemy is not close to them, it may have
glitched out of the navmesh and will never be able to find them again
    else {
        // Check if the enemy has glitched out of the navmesh
        let zombiePos = element.getAttribute('position').clone()
        zombiePos.add(new THREE.Vector3(0, 1, 0));
        let intersect = new THREE.Raycaster(zombiePos, new
THREE.Vector3(0, -1, 0)).intersectObject(floor.object3D);
        // If so, kill it
        if (intersect.length === 0) {
            element.components['zombie'].updateHP(1000000, false);
            console.log('death by suffocation');
        }
    }
},

```

Codi 10: Zombie hitPlayer()

Com podem veure, un zombi ataca al jugador quan arriba a la seva destinació i aquest segueix aprop. Si és aprop, llavors el zombi fa l'animació d'atac (*Punch*) i envia una notificació al component *player* que ha estat atacat. Cal notar que també comprovam si el zombi segueix estant damunt la *navmesh* i, si no hi és, el matam. Això és així per un hipotètic cas que es pot donar degut al funcionament del *pathfinding* on el zombi “escapa” d'aquesta: Quan l'algorisme troba un camí, el zombi va avançant fins arribar-hi. La forma en què ho fa és canviant la seva posició en cada fotograma. Ara bé, es pot donar el cas en què el zombi no està lo suficientment aprop de la paret com per detectar-la i quan faci una passa, aquesta el transporti fora dins aquesta (i, per tant, fora de la *navmesh*) i ja no pugui tornar a trobar el camí.

### 3. Desenvolupament del projecte

Per registrar els trets que el jugador efectua contra el zombi, es fan servir les *hitboxes* (les caixes blava i vermella vistes a la Figura 20). Si un tret col·lisiona amb alguna *hitbox* del zombi (vegeu Codi 11), enviarà una notificació al zombi i aquest actualitzarà el valor dels punts de vida que li manquen. En cas d'haver mort, enviarà una notificació al sistema.

```
function shootRay() {
  [...]
  var raycaster = new THREE.Raycaster(origin, direction);
  // Has it impacted any target?
  var targets =
Array.from(document.querySelectorAll('[target]')).map(obj =>
obj.object3D);
  var impacts = raycaster.intersectObjects(targets);
  [...]
  if (impacts.length > 0) {
    checkImpacts(impacts);
  }
}
function checkImpacts(impacts) {
  for (let i = 0; i < impacts.length; i++) {
    [...]
    if (impacts[i].object.el.classList.contains("headbox")) {
      damage_given = damage_given * 1.5;
      // Apply damage to every zombie hit
      impacts[i].object.el.parentElement.components['zombie'].updateHP(damage_given, true);
    } else {
      // Apply damage to every zombie hit
      impacts[i].object.el.parentElement.components['zombie'].updateHP(damage_given, false);
    }
  }
}
```

Codi 11: Detecció d'impactes

El Codi 11 és una simplificació de la implementació real per resumir com es du a terme la detecció d'impactes.

#### 3.4.3.5.3. Component "zombie-game"

S'encarrega del funcionament del joc. Organitza les rondes i hordes de zombis, escolta notificacions dels zombis per saber com progressa el jugador, escolta notificacions del jugador per saber quan s'acaba la partida i duu el recompte de la puntuació de la partida.

El funcionament d'aquesta component es resumeix bàsicament en les següents accions:

1. Escolta la mort del jugador
2. Comença una ronda
3. Genera zombis
4. Escolta la seva mort per dur el recompte de la puntuació i per generar-ne més / començar la següent ronda
5. En acabar la partida, envia els resultats de la partida al servidor.

## 3.4.3.6. Controls

Dins l'escena 3D, el jugador controla l'entitat *cameraRig*. Aquesta representa tot el cos del jugador dins l'escena i, com podem veure en el Codi 12, incorpora altres entitats dins ella mateixa per especificar altres parts del cos com el cap i les mans.

```
<!-- CAMERA RIG -->
<a-entity id="cameraRig" player movement-controls="controls: keyboard,
gamepad; constrainToNavMesh: true [...]>
  <!-- HEAD -->
  <a-entity [...] look-controls="pointerLockEnabled: true" camera
cursor="rayOrigin: mouse;" id="myCamera" position="0 1.75 0">
    <!-- CURSOR CROSSHAIR -->
    <a-cursor id="cursorSights" overlay color="white"></a-cursor>
    <!-- Gun holding hand -->
    <a-entity id="rightHand" position="0.15 -0.15 -0.2"
rotation="90 0 0"></a-entity>
  </a-entity>
</a-entity>
```

Codi 12: Controls

Cal notar, que cal especificar a través d'una component, els mètodes d'entrada que es faran servir per moure el *cameraRig* mentre que el cap (*myCamera*) implementa a través de la component *look-controls* la funció de rotar la vista.

També cal notar com la mà dreta (*rightHand*) és filla del cap (*myCamera*) i com no incorpora cap component relacionada amb els controls. Això és així perquè el Codi 12 fa referència a la versió per defecte del joc. És a dir, a quan s'hi juga a través de teclat i ratolí. En cas de que s'accedeixi a la funció de RV, es canvien dinàmicament els controls per ajustar-se al nou mode tal i com mostra el .

```
myScene.addEventListener('enter-vr', () => {
  inVR = true;

  cursorSights.remove();

  var leftHand = document.createElement('a-entity');
  leftHand.setAttribute('id', 'leftHand');
  leftHand.setAttribute('cursor');
  leftHand.setAttribute('raycaster', 'far: 5;');
  leftHand.setAttribute('laser-controls', 'hand: left;');
  cameraRig.appendChild(leftHand);

  cameraRig.remove(rightHand);
  myCamera.removeChild(rightHand);
  var rh = document.createElement('a-entity');
  rh.setAttribute('hand-controls', 'hand: right;');
  rh.setAttribute('id', 'rightHand');
  rh.setAttribute('visible', 'true');
  cameraRig.appendChild(rh);
});
```

Codi 13: Controls en RV

### 3. Desenvolupament del projecte

Com podem veure, en el Codi 13, apart de reorganitzar el *cameraRig*, també s'hi afegeix la mà esquerra (*leftHand*) i els controls per ambdues mans a través de les components *hand-controls* i *laser-controls*. S'empra *laser-controls* per la mà esquerra ja que aquesta component ens permet interactuar amb elements de l'escena. D'aquesta manera, el jugador pot fer servir la mà esquerra per prémer botons.

#### 3.5. Instal·lació

Aquest darrer apartat del capítol de desenvolupament s'encarrega d'explicar, de manera ordenada, el procediment a seguir per fer públic el joc. Tot i que a l'apartat 3.4.2 Servidor s'ha explicat com s'ha implementat la part del servidor en diferents àmbits, aquest apartat va més dedicat a la instal·lació de programari necessari per poder muntar-lo i fer-lo públic a Internet.

El primer pas és obtenir accés al servidor del LTIM. Això se'ns va donar fet i vam poder accedir-hi a connectant-nos al nostre subdomini per SSH. En aquest cas, el servidor estava emprant un contenidor *docker* [23] que feia servir *Ubuntu* [24].

Sobre aquest contenidor, s'ha instal·lat el servei de Node.js [17] i el gestor de base de dades MySQL [25] (vegeu Codi 14).

```
curl -fsSL https://deb.nodesource.com/setup_21.x | sudo -E bash - &&\
sudo apt-get install -y nodejs

sudo apt install mysql-server
```

Codi 14: Instal·lació Node.js i MySQL

A continuació s'han instal·lat totes les dependències i paquets necessaris de Node.js. El Codi 15 mostra resumidament una manera de fer-ho.

```
npm install express
npm install fs
npm install http
npm install https
npm install mysql
npm install path
```

Codi 15: Dependències Node.js

I, finalment, s'ha configurat el supervisor [26]. El supervisor és una eina de gestió de processos per sistemes operatius de tipus UNIX que té l'objectiu de controlar i supervisar l'execució de processos i aplicacions per garantir que funcionin correctament. Per fer-ho simplement hem de modificar el fitxer de configuració. El Codi 16 mostra resumidament una com s'ha configurat en el nostre cas.

```
[supervisord]
nodaemon=true
[program:sshd]
command=/usr/sbin/sshd -D

# SI FAS SERVIR MYSQL, DESCOMENTA AIXÒ
[program:mysql]
command=/usr/bin/pidproxy /var/run/mysqld/mysqld.pid
/usr/bin/mysqld_safe
```



```
autorestart=true

# PROGRAMA NODEJS
[program:zombis] # posa-li el nom que toqui
directory=[DIRECTORI] # directori on s'ha d'executar
command=[PROGRAMA] # ajusta l'ordre per iniciar-lo
autorestart=true # volem que s'iniciï tot sol, no?
user=[USUARI] # usuari que l'ha d'executar
group=[GRUP] # grup que l'ha d'executar
stdout_logfile=[RUTA] # on s'escriurà el log de sortida
stderr_logfile=[RUTA] # on s'escriurà el log d'error
environment=NODE_ENV="production"# variables d'entorn
```

Codi 16: Configuració supervisor



## Capítol 4. Resultats

El resultat principal obtingut en aquest projecte és una aplicació web que consisteix en un joc accessible per a tots els usuaris ja sigui amb un PC o bé amb equipament de RV. En segon lloc, s'ha dissenyat una petita pàgina web que serveix com presentació i tutorial del joc.

### 4.1. Pàgina principal

Podeu accedir a la pàgina principal a través de la direcció <https://zombis.ltim.uib.es>. La pàgina principal explica resumidament els controls del joc i com jugar (vegeu Figura 21). També aporta una breu descripció del projecte i les xarxes socials del creador.

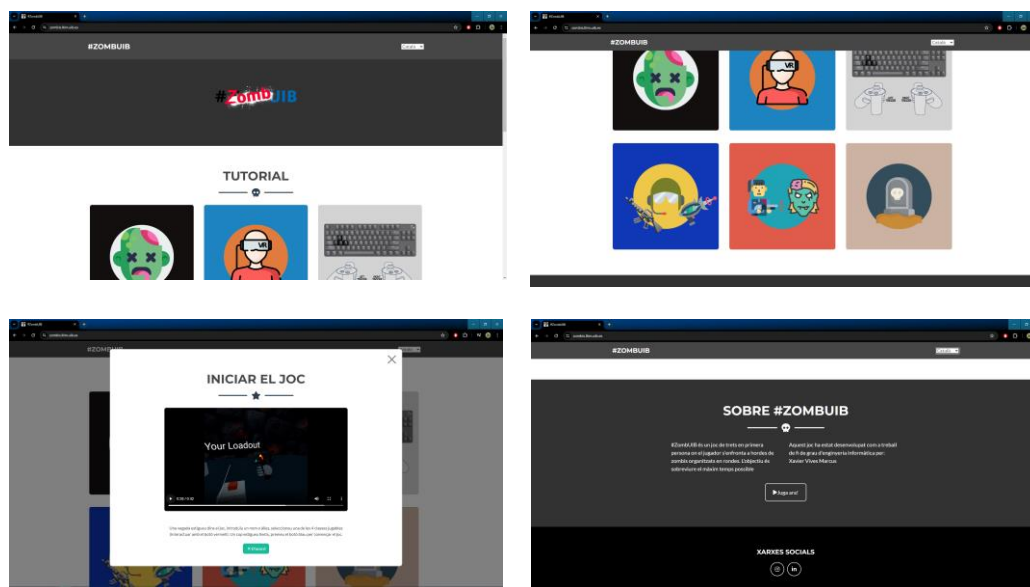


Figura 21: Pàgina principal

### 4.2. El joc

S'ha aconseguit crear una atmosfera tètrica i de terror tal i com es pretenia des del principi del projecte. Per aconseguir aquest efecte immersiu, s'han fet ús de diverses tècniques de generació d'entorns detallades a la secció 3.3.3 Disseny gràfic.

A més de l'ambient visual, s'ha creat també una banda sonora específica per al joc que contribueix significativament a l'atmosfera d'intranquil·litat. Aquesta es reproduïx en el canvi de ronda per reforçar el sentiment de progrés i tensió al jugador.

S'han implementat diversos efectes especials de so com ara: les armes emeten so quan son disparades (diferents armes emeten diferents sons), quan el jugador és ferit pels zombis, aquest gemega, quan un tret fereix un zombi, un so de confirmació es reproduïx.

#### 4. Resultats

També s'han implementat diferents animacions als zombis que accentuen la immersió en el joc. Com ara: caminar, arrossegar-se, atac i diferents animacions per la mort. Per veure com s'ha fet, vegeu 3.4.3.2 Càrrega de models 3D.

Podeu accedir al joc a través de la direcció <https://zombis.ltim.uib.es/game> o des de la pàgina principal <https://zombis.ltim.uib.es>.

A continuació es mostra com es veu el cycle de joc. La Figura 22 mostra el que veu el jugador només entra al joc. Com podem veure, té un camp de text on ha d'establir el seu àlies. També hi ha un teclat perquè el jugador pugui escriure si fa servir equipament de RV. En aquest cas, el jugador ha escollit l'àlies "Atomic Lettuce". A l'esquerra es pot veure la taula de marcadors.



Figura 22: El joc

La Figura 23 mostra com el jugador ha escollint la classe com la que vol jugar. Per fer-ho, ha interactuat amb un dels quatre botons vermells que hi ha dins el vestíbul. En aquest cas, el jugador ha escollit la classe "Infantry" l'arma de la qual és la PPSH. Com el jugador ja ha escollit àlies i classe, haurà de prémer el botó blau que té davant per poder començar la partida.

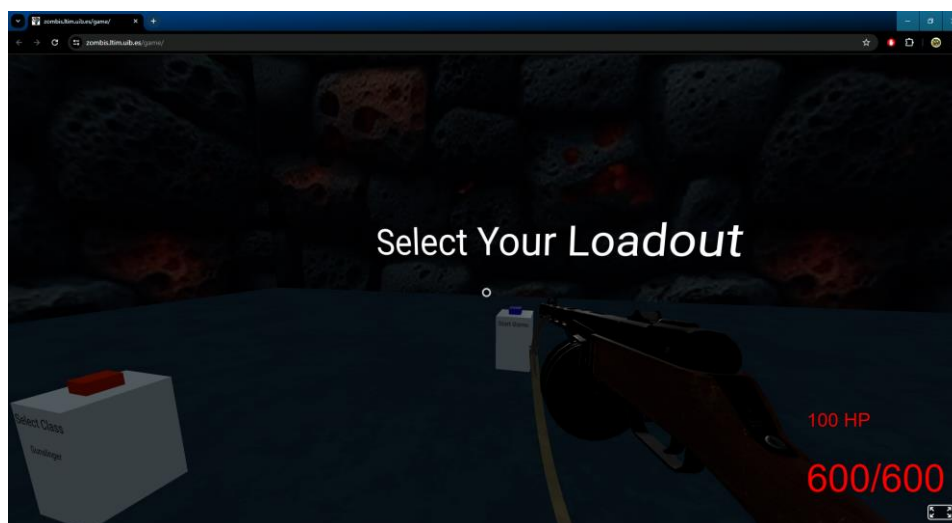


Figura 23: Escollir classe

La Figura 24 mostra el jugador dins del laberint enfrontant-se als zombis que van apareixent.

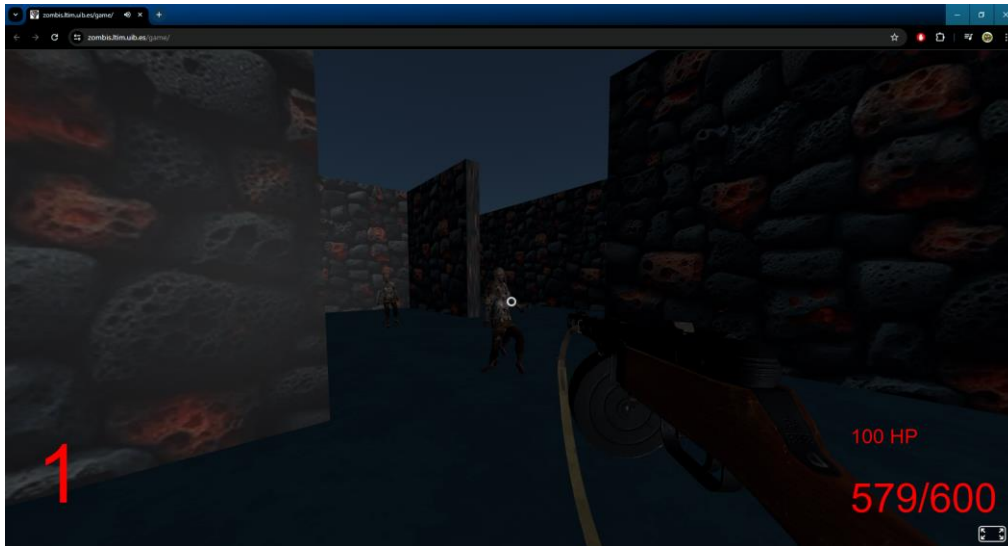


Figura 24: Dins una partida

La Figura 25 mostra el jugador en el vestíbul post-partida després d'haver estat abatut pels zombis. Com podem veure, té un botó lila davant que, quan sigui premut, reiniciarà el joc.

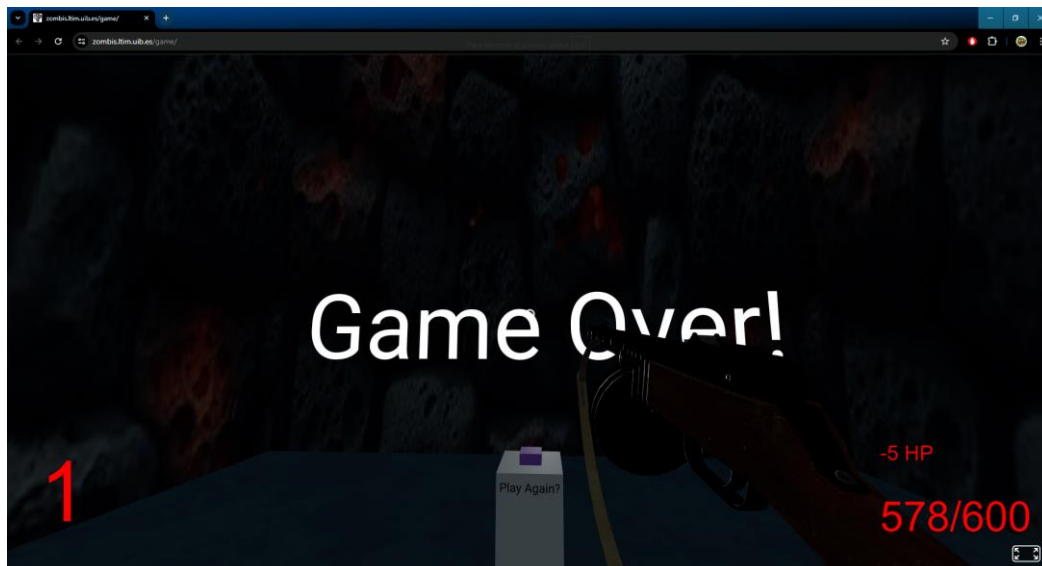


Figura 25: Vestíbul post-partida

### Capítol 5. Conclusions

Per concloure aquest TFG, es revisaran els objectius proposats a l'inici del document justificant i comprovant que s'hagin complert. També es compartiran opinions personals sobre el mateix TFG, dificultats trobades i altres aspectes rellevants.

En primer lloc, l'objectiu de desenvolupar un joc de trets en primera persona en el que el jugador s'enfronta a hordes de zombis organitzats en rondes. Aquest objectiu s'ha complert amb èxit i aquest document és la prova, ja que l'objectiu descriu quina serà l'essència del TFG.

En segon lloc, tenim l'objectiu de que el joc sigui compatible tant amb equipament de RV com amb un PC tradicional. Això s'ha aconseguit gràcies a que s'ha fet servir A-Frame [3] i ha facilitat considerablement la feina.

En tercer lloc, vàrem establir el següent objectiu: "recopilar informació de cada partida per poder elaborar una taula de puntuacions que mostri el top 10 millors jugadors". Aquest objectiu tenia la finalitat de generar un esperit competitiu entre els jugadors, fet que s'ha complert a la fira de *Ciència per a tothom* [7].

Finalment, el darrer objectiu, era tenir una versió jugable per exposar a la fira *Ciència per a tothom* [7]. Per complir aquest, es va haver de dur una bona gestió del temps i, fins i tot, fer "hores extres". Però, finalment, es va poder complir i fou un èxit. Durant la fira alumnes de primària fins a batxillerat tengueren la oportunitat de provar el joc. Aprofitarem això per refinar alguns detalls com, per exemple: en la versió de RV no es veu el HUD sempre ja que obstrueix massa el camp visual sino que el jugador l'activa o el desactiva a voluntat o, una vegada el jugador està en el laberint només veu la mà que té l'arma per facilitar l'experiència als jugadors més novats. La Figura 26 mostra fotografies al respecte.

En aquesta fira també es va poder observar els diferents comportaments que tenen alumnes de diferents edats mentre els seus companys proven el joc. Per lo general podem distingir com els alumnes de primària i primer d'ESO tendien a fer comentaris positius i de recolzament als seus companys. Addicionalment, eren més propensos a no voler provar el joc per por a tenir malsons aquell mateix vespre mentre que els alumnes més majors es dedicaven (en un context amigable) a molestar-se els uns als altres i fer bromes més agressives.

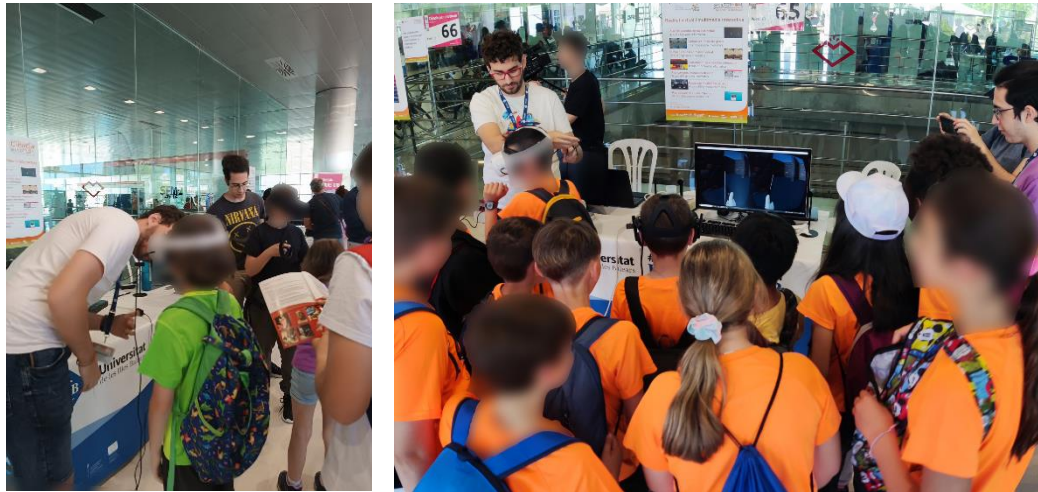


Figura 26: Ciència per a Tothom

En l'àmbit personal, sent que aquest TFG ha complert amb les meves expectatives que eren: "Fer alguna cosa, si t'agrada millor i si a més a més, aprens noves lliçons, espectacular." Jo vaig tenir la sort de tenir com a opció fer una versió personalitzada del meu joc preferit, per tant, era segur que m'agradaria i, ara que l'he acabat, puc dir que he après moltes lliçons noves en l'àmbit tècnic com ara: geometria, treballar amb servidors, informàtica gràfica, modelatge 3D, etc. I, també, lliçons no tècniques com veure la diferència de comportament d'alumnes de primària fins a ESO provant #ZombUIB a la fira *Ciència per a Tothom* [7].

Respecte a les dificultats del projecte, la complicació més gran va ser generar la navmesh dinàmicament. S'intentaren diferents mètodes que es basaven a fusionar les geometries de diferents objectes però fracassaren tots. Finalment, es va pensar que es podria emprar la regla de la mà dreta [22], es va demostrar que qualsevol laberint generat amb l'algorisme de prim es podia resoldre d'aquesta manera i es va desenvolupar el codi que teniu disponible a Apèndix>Generació de navmesh. Addicionalment, convertir un conjunt de vèrtexs a una figura plana tampoc va ser senzill per mor que A-Frame no té una primitiva per fer-ho directament i es va haver de crear una component personalitzada per fer-ho. Aquesta darrera complicació es deu no tant a la complexitat de la tasca com a tal sinó més aviat en la inexperiència de l'eina. Teniu el codi per crear aquesta component disponible a Apèndix>A-Frame irregular-polygon custom component.

Una altra complicació considerable va ser el *memory leak* (vist anteriorment en 3.4.3.2 Càrrega de models 3D). La problemàtica no va ser arreglar el problema com el procés d'investigar i descobrir què l'ocasionava.

Sobre les assignatures cursades que més han importància han tengut en aquest projecte han estat: Informàtica gràfica, Tecnologies Multimèdia i Base de dades I. Hagués estat també interessant haver cursat Gestió i Distribució de la Informació Empresarial, ja que serveix com a continuació de Tecnologies Multimèdia i expandeix el contingut de la part del servidor. Addicionalment, Antoni Oliver Tomàs (tutor d'aquest TFG i professor de l'assignatura) ens va facilitar material de l'assignatura per poder realitzar més fàcilment algunes tasques com ara la configuració del servidor.



## 5. Conclusions

### 5.1. Línies futures

Tot i que l'aplicació compleix amb els requisits i objectius establerts a l'inici del projecte, és cert que es podrien implementar certes actualitzacions de cara al futur.

Com ja s'ha explicat anteriorment, aquest projecte té com a font d'inspiració la saga *Call of Duty: Black Ops – Zombies*. Una característica emblemàtica de la saga és el sistema de progrés. És a dir, els jugadors comencen la partida amb armes poc potents i conforme avancen en la partida aconsegueixen punts que poden emprar com a moneda per aconseguir millores com ara, nou armament, millorar l'armament actual, obrir portes que duen a noves parts del mapa i begudes (vegeu Figura 27) que aporten millores del personatge com ara més punts de vida, més rapidesa en la recàrrega, etc.

(a)



(b)



(c)



(d)



Figura 27: Begudes potenciadores. (a) Juggernog, (b) Speed Cola, (c) Quick Revive, (d) Deadshot Daiquiri

Una altre de les característiques més emblemàtiques de la saga són les armes especials (vegeu Figura 28). Aquestes son armes molt poderoses capaces d'eliminar una gran quantitat de zombis amb un sol tret. Cada mapa sol tenir la seva pròpia arma especial i acostumen a tenir poders de ciència ficció com raigs, forats negres, etc. Sovint es poden obtenir si el jugador completa una missió secundària i hagués estat una bona addició per #ZombUIB una missió secundària d'aquest estil.



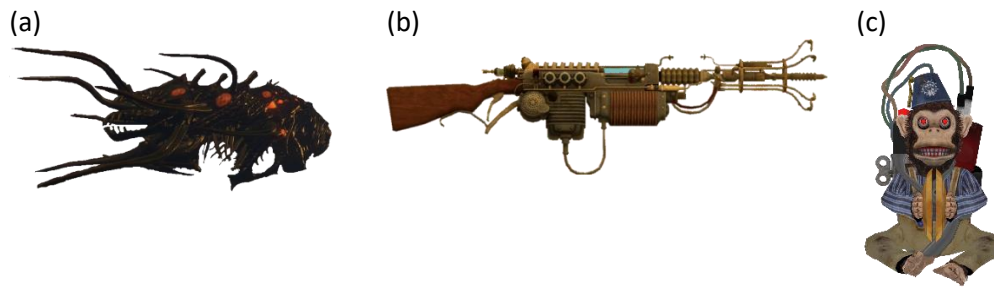


Figura 28: Armes especials. (a) Apothicon Servant, (b) Wunderwaffe DG-2, (c) Monkey Bomb

Finalment, es podrien elaborar més mapes que no necessàriament es generin dinàmicament sinó que sempre siguin iguals. Algunes idees podrien camps a la perifèria, una fàbrica abandonada, un pantà, etc. I, sobretot, jugar més amb la verticalitat. #ZombUIB només té un mapa (el laberint) i aquest no té mai cap desnivell.

- [1] Hillary, «Virtual Reality: 10 Ways It's Shifting Our Perspectives,» *TechBullion*, pp. <https://techbullion.com/virtual-reality-10-ways-its-shifting-our-perspectives/>, 25 May 2024.
- [2] W. contributors, «Wikipedia,» 12 May 2024. [En línia]. Available: [https://en.wikipedia.org/wiki/AAA\\_\(video\\_game\\_industry\)](https://en.wikipedia.org/wiki/AAA_(video_game_industry)).
- [3] «A-Frame - make WebVR,» [En línia]. Available: <https://aframe.io>.
- [4] «Wonderland Engine,» [En línia]. Available: <https://wonderlandengine.com>.
- [5] Wonderland Engine, «Top 5 web XR frameworks- Comparison,» [En línia]. Available: <https://wonderlandengine.com/news/top-5-webxr-frameworks-comparison/>.
- [6] «Sketchfab,» [En línia]. Available: <https://sketchfab.com/>.
- [7] UIB, «Ciència per a Tothom 2024,» [En línia]. Available: <https://seras.uib.cat/ciencia>.
- [8] «A-Frame-Environment,» [En línia]. Available: <https://www.npmjs.com/package/aframe-environment-component>.
- [9] «Sketchfab M1911 3D model,» [En línia]. Available: <https://sketchfab.com/3d-models/pistol-1911-8e0860d318314e9286be34f4ab7d2207>.
- [10] «Sketchfab PPSH 3D model,» [En línia]. Available: <https://sketchfab.com/3d-models/ppsh-41-low-poly-d9ee1742732c4aec9c98aeddff93071c>.
- [11] «Sketchfab Raygun 3D model,» [En línia]. Available: <https://sketchfab.com/3d-models/raygun-call-of-duty-zombies-927ef1106e5f4ddd9aa45d64ce072991>.
- [12] «Sketchfab Cat 3D model,» [En línia]. Available: <https://sketchfab.com/3d-models/the-cats-body-04ae93d572904be8b7d53a6ccb71c433>.
- [13] StackOverflow, «MySQL 8.0 - Client does not support authentication protocol requested by server; consider upgrading MySQL client,» [En línia]. Available: <https://stackoverflow.com/questions/50093144/mysql-8-0-client-does-not-support-authentication-protocol-requested-by-server>.
- [14] «Express.js,» [En línia]. Available: <https://expressjs.com/>.
- [15] «Node.js HTTP library,» [En línia]. Available: <https://nodejs.org/api/http.html>.

- 
- [16] «Node.js HTTPS library,» [En línia]. Available: <https://nodejs.org/api/https.html>.
- [17] « Node.js MySQL package,» [En línia]. Available: <https://www.npmjs.com/package/mysql>.
- [18] «A-Frame-Extras,» [En línia]. Available: <https://www.npmjs.com/package/aframe-extras/v/4.2.0>.
- [19] Blender, «Home of the Blender project - Free and Open 3D Creation Software,» [En línia]. Available: <https://www.blender.org/>.
- [20] «Three.js,» [En línia]. Available: <https://threejs.org/>.
- [21] A-Frame, «Destroying a scene or switching scene without memory leak,» [En línia]. Available: <https://aframe.wiki/en/#!/pages/memory.md>.
- [22] Wikipedia, «Maze-solving algorithm. Hand on wall rule,» [En línia]. Available: [https://en.wikipedia.org/wiki/Maze-solving\\_algorithm#Hand\\_On\\_Wall\\_Rule](https://en.wikipedia.org/wiki/Maze-solving_algorithm#Hand_On_Wall_Rule).
- [23] «Docker,» [En línia]. Available: <https://www.docker.com/>.
- [24] «Ubuntu,» [En línia]. Available: <https://ubuntu.com/>.
- [25] «MySQL,» [En línia]. Available: <https://www.mysql.com/>.
- [26] «Supervirord,» [En línia]. Available: <http://supervisord.org/>.
- [27] «Mixamo,» [En línia]. Available: <https://www.mixamo.com/#/>.

## Algorisme de Prim per Generar laberints

```
const MAZE_ROWS = 10;
const MAZE_COLS = 10;
var MAZE_MATRIX = [];
var visited = new Set();
var frontier = new Set();

function remove_random_wall(cell) {
  let neighbours = [];
  // Northern neighbour
  if (cell.row - 1 >= 0) {
    if (MAZE_MATRIX[cell.row - 1][cell.col].visited) {
      neighbours.push(MAZE_MATRIX[cell.row - 1][cell.col])
    }
  }
  // Eastern neighbour
  if (cell.col + 1 < MAZE_COLS) {
    if (MAZE_MATRIX[cell.row][cell.col + 1].visited) {
      neighbours.push(MAZE_MATRIX[cell.row][cell.col + 1]);
    }
  }
  // Southern neighbour
  if (cell.row + 1 < MAZE_ROWS) {
    if (MAZE_MATRIX[cell.row + 1][cell.col].visited) {
      neighbours.push(MAZE_MATRIX[cell.row + 1][cell.col]);
    }
  }
  // Western neighbour
  if (cell.col - 1 >= 0) {
    if (MAZE_MATRIX[cell.row][cell.col - 1].visited) {
      neighbours.push(MAZE_MATRIX[cell.row][cell.col - 1]);
    }
  }
  let random_neighbour = neighbours[Math.floor(Math.random() *
neighbours.length)];

  if (random_neighbour.col == cell.col && random_neighbour.row <
cell.row) {
    random_neighbour.south = false;
    cell.north = false;
  } else if (random_neighbour.col == cell.col && random_neighbour.row
> cell.row) {
    random_neighbour.north = false;
    cell.south = false;
  } else if (random_neighbour.col < cell.col && random_neighbour.row
== cell.row) {
    random_neighbour.east = false;
    cell.west = false;
  } else {
    random_neighbour.west = false;
```

```

    cell.east = false;
  }
}

function add_frontiers(cell) {
  let cell_row = cell.row;
  let cell_col = cell.col;

  // Northern neighbour
  if (cell.row - 1 >= 0) {
    if (!MAZE_MATRIX[cell.row - 1][cell.col].visited) {
      frontier.add(MAZE_MATRIX[cell.row - 1][cell.col]);
    }
  }
  // Eastern neighbour
  if (cell.col + 1 < MAZE_ROWS) {
    if (!MAZE_MATRIX[cell.row][cell.col + 1].visited) {
      frontier.add(MAZE_MATRIX[cell.row][cell.col + 1]);
    }
  }
  // Southern neighbour
  if (cell.row + 1 < MAZE_ROWS) {
    if (!MAZE_MATRIX[cell.row + 1][cell.col].visited) {
      frontier.add(MAZE_MATRIX[cell.row + 1][cell.col]);
    }
  }
  // Western neighbour
  if (cell.col - 1 >= 0) {
    if (!MAZE_MATRIX[cell.row][cell.col - 1].visited) {
      frontier.add(MAZE_MATRIX[cell.row][cell.col - 1]);
    }
  }
}

function generate_maze() {
  for (let i = 0; i < MAZE_ROWS; i++) {
    MAZE_MATRIX.push([]);
    for (let j = 0; j < MAZE_COLS; j++) {
      MAZE_MATRIX[i].push({
        row: i,
        col: j,
        visited: false,
        north: true,
        east: true,
        south: true,
        west: true
      });
    }
  }
}

// Choose a cell as the starting point and add it to the visited set
let start = MAZE_MATRIX[0][0];
start.visited = true;
visited.add(start);

```

```
// Add all unvisited cells that are adjacent to the current cell to
the frontier set
add_frontiers(start);

// Choose a cell randomly from the frontier set and make it the
current cell,
// removing it from the frontier set and adding it to the visited
set
let i = 0;
while (frontier.size > 0) {
    let current = Array.from(frontier)[[Math.floor(Math.random() *
frontier.size)]];
    frontier.delete(current);
    visited.add(current);
    current.visited = true;
    // Remove the wall between the current cell and a random adjacent
cell that
    // belongs to the visited set.
    remove_random_wall(current);
    add_frontiers(current);
}
}

generate_maze();
console.log(MAZE_MATRIX);
```

Codi 17: Implementació en javascript de l'algorisme de Prim per generar laberints

## Generació de navmesh

```
function createFloor() {
    var floor = document.createElement('a-entity');
    floor.setAttribute('id', 'floor');
    floor.setAttribute('irregular-polygon', "vertices: 0 0 0, 1 0
4, 3 0 5, 3 0 3, 4 0 3, 4 0 0;");
    floor.setAttribute('position', '0 0 0');
    floor.setAttribute('rotation', '90 0 0');

    var vertices = [];
    // Save first vertex
    var first_vertex = (-(cfg.MAZE.MAZE_CELL_SIZE -
cfg.MAZE.MAZE_WALL_THICKNESS) / 2).toString() + " 0 " + (-(
cfg.MAZE.MAZE_CELL_SIZE - cfg.MAZE.MAZE_WALL_THICKNESS) /
2).toString();
    vertices.push(first_vertex);

    var current_vertex = '';
    var direction = 'east';
    var current_cell = [0, 0];
    while (!(current_vertex === first_vertex)) {
        switch (direction) {
            case 'north':
                while
```



```

        */
        else {
            var z = current_cell[0] *
cfg.MAZE.MAZE_CELL_SIZE;
            z = z -
((cfg.MAZE.MAZE_CELL_SIZE) / 2);
            var x = current_cell[1] *
cfg.MAZE.MAZE_CELL_SIZE;
            x = x -
((cfg.MAZE.MAZE_CELL_SIZE - cfg.MAZE.MAZE_WALL_THICKNESS) / 2);
            current_vertex = x.toString() +
" 0 " + z.toString();
            vertices.push(current_vertex);
            current_cell[1]--;
            x = current_cell[1] *
cfg.MAZE.MAZE_CELL_SIZE;
            x = x +
((cfg.MAZE.MAZE_CELL_SIZE - cfg.MAZE.MAZE_WALL_THICKNESS) / 2);
            current_vertex = x.toString() +
" 0 " + z.toString();
            vertices.push(current_vertex);
            direction = 'south'
        }
    }
    // SCENARIO where we no longer go north
    because we found an obstacle
    /*
    SCENARIO
        */
        else {
            var z = current_cell[0] *
cfg.MAZE.MAZE_CELL_SIZE;
            z = z - ((cfg.MAZE.MAZE_CELL_SIZE -
cfg.MAZE.MAZE_WALL_THICKNESS) / 2);
            var x = current_cell[1] *
cfg.MAZE.MAZE_CELL_SIZE;
            x = x - ((cfg.MAZE.MAZE_CELL_SIZE -
cfg.MAZE.MAZE_WALL_THICKNESS) / 2);
            current_vertex = x.toString() + " 0 "
+ z.toString();
            vertices.push(current_vertex);

```



```

        direction = 'east';
    }
    break;
case 'east':
    while
(!MAZE_MATRIX[current_cell[0]][current_cell[1]].east &&
MAZE_MATRIX[current_cell[0]][current_cell[1]].north) {
        current_cell[1]++;
    }
    // SCENARIOS where we don't continue going
    east because we no longer have a wall to follow
    if
(!MAZE_MATRIX[current_cell[0]][current_cell[1]].north) {
        current_cell[1]--;
        /*
        SCENARIO

        ┌───┐
        │   │
        │───┤
        │   │

        */
        if (MAZE_MATRIX[current_cell[0] -
1][current_cell[1] + 1].west) {
            var z = current_cell[0] *
cfg.MAZE.MAZE_CELL_SIZE;
            z = z -
((cfg.MAZE.MAZE_CELL_SIZE - cfg.MAZE.MAZE_WALL_THICKNESS) / 2);
            var x = current_cell[1] *
cfg.MAZE.MAZE_CELL_SIZE;
            x = x +
((cfg.MAZE.MAZE_CELL_SIZE) / 2);

            current_vertex = x.toString() +
" 0 " + z.toString();
            vertices.push(current_vertex);

            current_cell[0]--;
            current_cell[1]++;

            z = current_cell[0] *
cfg.MAZE.MAZE_CELL_SIZE;
            z = z +
((cfg.MAZE.MAZE_CELL_SIZE - cfg.MAZE.MAZE_WALL_THICKNESS) / 2);
            x = current_cell[1] *
cfg.MAZE.MAZE_CELL_SIZE;
            x = x -
((cfg.MAZE.MAZE_CELL_SIZE - cfg.MAZE.MAZE_WALL_THICKNESS) / 2);


            current_vertex = x.toString() +
" 0 " + z.toString();
            vertices.push(current_vertex);

            direction = 'north';

```

```

    }


    /*
    SCENARIO
    
    */
    else {
        var z = current_cell[0] *
cfg.MAZE.MAZE_CELL_SIZE;
        z = z -
((cfg.MAZE.MAZE_CELL_SIZE - cfg.MAZE.MAZE_WALL_THICKNESS) / 2);
        var x = current_cell[1] *
cfg.MAZE.MAZE_CELL_SIZE;
        x = x +
((cfg.MAZE.MAZE_CELL_SIZE) / 2);

        current_vertex = x.toString() +
" 0 " + z.toString();
        vertices.push(current_vertex);

        current_cell[0]--;
        z = current_cell[0] *
cfg.MAZE.MAZE_CELL_SIZE;
        z = z +
((cfg.MAZE.MAZE_CELL_SIZE - cfg.MAZE.MAZE_WALL_THICKNESS) / 2);

        current_vertex = x.toString() +
" 0 " + z.toString();
        vertices.push(current_vertex);

        direction = 'west'
    }

}
// SCENARIO where we no longer go east
because we found an obstacle
/*
SCENARIO

*/
    else {
        var z = current_cell[0] *
cfg.MAZE.MAZE_CELL_SIZE;
        z = z - ((cfg.MAZE.MAZE_CELL_SIZE -
cfg.MAZE.MAZE_WALL_THICKNESS) / 2);

```

```

        var x = current_cell[1] *
cfg.MAZE.MAZE_CELL_SIZE;
        x = x + ((cfg.MAZE.MAZE_CELL_SIZE -
cfg.MAZE.MAZE_WALL_THICKNESS) / 2);

        current_vertex = x.toString() + " 0 "
+ z.toString();
        vertices.push(current_vertex);

        direction = 'south';
    }

    break;
    case 'south':
        while
(!MAZE_MATRIX[current_cell[0]][current_cell[1]].south &&
MAZE_MATRIX[current_cell[0]][current_cell[1]].east) {
            current_cell[0]++;
        }

        // SCENARIOS where we don't continue going
south because we no longer have a wall to follow
        if
(!MAZE_MATRIX[current_cell[0]][current_cell[1]].east) {
            current_cell[0]--;
            /*
            SCENARIO
            | |
            | |
            | |
            | |
            */
            if (MAZE_MATRIX[current_cell[0] +
1][current_cell[1] + 1].north) {
                var z = current_cell[0] *
cfg.MAZE.MAZE_CELL_SIZE;
                z = z +
                var x = current_cell[1] *
                x = x +
                ((cfg.MAZE.MAZE_CELL_SIZE - cfg.MAZE.MAZE_WALL_THICKNESS) / 2);

                current_vertex = x.toString() +
" 0 " + z.toString();
                vertices.push(current_vertex);

                current_cell[0]++;
                current_cell[1]++;

                z = current_cell[0] *
cfg.MAZE.MAZE_CELL_SIZE;
                z = z -

```

```

((cfg.MAZE.MAZE_CELL_SIZE - cfg.MAZE.MAZE_WALL_THICKNESS) / 2);
    x = current_cell[1] *
cfg.MAZE.MAZE_CELL_SIZE;
    x = x -
((cfg.MAZE.MAZE_CELL_SIZE - cfg.MAZE.MAZE_WALL_THICKNESS) / 2);

    current_vertex = x.toString() +
" 0 " + z.toString();
    vertices.push(current_vertex);
    direction = 'east';
}
/*
SCENARIO
    | | |
    | | |
    | | |
*/
else {
    var z = current_cell[0] *
cfg.MAZE.MAZE_CELL_SIZE;
    z = z +
((cfg.MAZE.MAZE_CELL_SIZE) / 2);
    var x = current_cell[1] *
cfg.MAZE.MAZE_CELL_SIZE;
    x = x +
((cfg.MAZE.MAZE_CELL_SIZE - cfg.MAZE.MAZE_WALL_THICKNESS) / 2);

    current_vertex = x.toString() +
" 0 " + z.toString();
    vertices.push(current_vertex);

    current_cell[1]++;
    x = current_cell[1] *
cfg.MAZE.MAZE_CELL_SIZE;
    x = x -
((cfg.MAZE.MAZE_CELL_SIZE - cfg.MAZE.MAZE_WALL_THICKNESS) / 2);

    current_vertex = x.toString() +
" 0 " + z.toString();
    vertices.push(current_vertex);
    direction = 'north'
}
}
// SCENARIO where we no longer go south
because we found an obstacle
/*
SCENARIO
    | - |
    |   |
    |   |

```




```

vertices.push(current_vertex);

current_cell[0]++;
current_cell[1]--;

z = current_cell[0] *
cfg.MAZE.MAZE_CELL_SIZE;
z = z -
((cfg.MAZE.MAZE_CELL_SIZE - cfg.MAZE.MAZE_WALL_THICKNESS) / 2);
x = current_cell[1] *
cfg.MAZE.MAZE_CELL_SIZE;
x = x +
((cfg.MAZE.MAZE_CELL_SIZE - cfg.MAZE.MAZE_WALL_THICKNESS) / 2);

current_vertex = x.toString() +
" 0 " + z.toString();
vertices.push(current_vertex);

direction = 'south';
}
/*
SCENARIO

*/
else {
var z = current_cell[0] *
cfg.MAZE.MAZE_CELL_SIZE;
z = z +
((cfg.MAZE.MAZE_CELL_SIZE - cfg.MAZE.MAZE_WALL_THICKNESS) / 2);
var x = current_cell[1] *
cfg.MAZE.MAZE_CELL_SIZE;
x = x -
((cfg.MAZE.MAZE_CELL_SIZE) / 2);

current_vertex = x.toString() +
" 0 " + z.toString();
vertices.push(current_vertex);

current_cell[0]++;
z = current_cell[0] *
cfg.MAZE.MAZE_CELL_SIZE;
z = z -
((cfg.MAZE.MAZE_CELL_SIZE - cfg.MAZE.MAZE_WALL_THICKNESS) / 2);

current_vertex = x.toString() +
" 0 " + z.toString();
vertices.push(current_vertex);

direction = 'east'
}
}

```

```

        // SCENARIO where we no longer go west
        because we found an obstacle
        /*
        SCENARIO

        */
        else {
            var z = current_cell[0] *
cfg.MAZE.MAZE_CELL_SIZE;
            z = z + ((cfg.MAZE.MAZE_CELL_SIZE -
cfg.MAZE.MAZE_WALL_THICKNESS) / 2);
            var x = current_cell[1] *
cfg.MAZE.MAZE_CELL_SIZE;
            x = x - ((cfg.MAZE.MAZE_CELL_SIZE -
cfg.MAZE.MAZE_WALL_THICKNESS) / 2);

            current_vertex = x.toString() + " 0 "
+ z.toString();
            vertices.push(current_vertex);

            direction = 'north';
        }
        break;
    }
}

// Delete last vertex for good measure (last vertex is equal to
first vertex)
vertices.pop();

floor.setAttribute('irregular-polygon', "vertices: " +
vertices.toString());
floor.setAttribute('nav-mesh', '');
floor.setAttribute('visible', 'false');
floor.setAttribute('class', 'navmesh');

return floor;
}

```

Codi 18: Implementació en javascript per generar la navmesh del laberint

### A-Frame irregular-polygon custom component

```

AFRAME.registerComponent('irregular-polygon', {
  schema: {
    vertices: { type: 'string', default: '' } // Comma-separated
list of vertex coordinates
  },
  init: function () {
    const vertices = this.data.vertices.split(',').map(coord =>
coord.trim().split(' ').map(parseFloat));

```

```

    const shape = new THREE.Shape(vertices.map(v => new
THREE.Vector3(v[0], v[2]))); // Use X and Z coordinates
    const geometry = new THREE.ShapeGeometry(shape);
    const material = new THREE.MeshBasicMaterial({ color:
0x00ff00, side: THREE.DoubleSide });
    const mesh = new THREE.Mesh(geometry, material);
    this.el.setObject3D('mesh', mesh);
    //this.el.object3D.add(mesh);
  }
});

```

### Eliminar model 3D de memòria

```

function disposeTextures(material) {
  // Explicitly dispose any textures assigned to this material
  for (const propertyName in material) {
    const texture = material[propertyName];
    if (texture instanceof THREE.Texture) {
      const image = texture.source.data;
      if (image instanceof ImageBitmap) {
        image.close && image.close();
      }
      texture.dispose();
    }
  }
}

function disposeNode(node) {
  if (node instanceof THREE.Mesh) {
    const geometry = node.geometry;
    if (geometry) {
      geometry.dispose();
    }

    const material = node.material;
    if (material) {
      if (Array.isArray(material)) {
        for (let i = 0, l = material.length; i < l; i++) {
          const m = material[i];
          disposeTextures(m);
          m.dispose();
        }
      } else {
        disposeTextures(material);
        material.dispose(); // disposes any programs associated with
the material
      }
    }
  }
}

AFRAME.registerComponent("gltf-model", {
  [...]
  remove: function () {
    if (!this.model) {
      return;
    }
  }
}

```



---

```
    this.el.removeObject3D("mesh");
    // New code to remove all resources
    this.model.traverse(disposeNode);
    this.model = null;

    THREE.Cache.clear();
    // Empty renderLists to remove references to removed objects for
    garbage collection
    this.el.sceneEl.renderer.renderLists.dispose();
  },
});
```

Codi 19: Eliminació de models 3D en memòria principal