Universitat de les Illes Balears

TREBALL DE FI DE GRAU

BELLVER DEFENSE: TOWER DEFENSE A LA INVERSA EN REALITAT VIRTUAL

Alexandre Joan Morro Santandreu

Grau d'Enginyeria Informàtica

Escola Politècnica Superior

Any acadèmic 2024-25

BELLVER DEFENSE: TOWER DEFENSE A LA INVERSA EN REALITAT VIRTUAL

Alexandre Joan Morro Santandreu

Treball de Fi de Grau

Escola Politècnica Superior

Universitat de les Illes Balears

Any acadèmic 2024-25

Paraules clau del treball:

Joc, tir amb arc, realitat virtual, tower defense invers

Nom del tutor: Antoni Oliver Tomàs

Nom del segon tutor: Antonio Bibiloni Coll

Agraïments

Amb agraïment a la meva família, que m'ha brindat el seu suport econòmic i emocional durant tota la meva vida. Sense la seva ajuda, paciència i confiança en mi, aquest projecte no hauria estat possible.

Al meu tutor Antoni Oliver, pel seu suport, consells i paciència que han ajudat a fer que el projecte sigui el millor possible.

Taula de continguts

Agraïments i
Taula de continguts v
Índex de figures ix
Índex de taules xi
Nomenclatura xiii
Resum xv
Capítol 1. Introducció 1
1.1. Contextualització 1
1.2. Adaptació 1
1.3. Objectius 1
1.4. Abast del projecte 2
1.5. Estructura del document 3
Capítol 2. Estat de l'art 5
2.1. Jocs similars existents 5
2.1.1. Royal Revolt! 5
2.1.2. Bloons TD 6 6
2.1.3. Temple run 6
2.1.4. The Lab – Longbow 7
2.2. Característiques interessants 8
2.2.1. Característiques escollides 9
Capítol 3. Desenvolupament del projecte 11
3.1. Gestió del projecte 11
3.1.1. Metodologia de desenvolupament 11
3.1.2. Planificació temporal 11
3.2. Anàlisi 12
3.2.1. Usuaris 12

3.2.2. Requisits funcionals 12
3.2.2.1. Requisits funcionals d'usuari 12
3.2.2.2. Requisits funcionals de sistema 12
3.2.3. Requisits no funcionals 13
3.3. Disseny 13
3.3.1. Eines 13
3.3.1.1. Desenvolupament de jocs web en RV 14
3.3.1.1.1. Three.js 14
3.3.1.1.2. A-Frame 14
3.3.1.1.3. WonderlandEngine 14
3.3.1.2. Objectes i models 3D o animacions 14
3.3.1.2.1. Blender 15
3.3.1.2.2. Maya 16
3.3.1.3. Gestió de versions 16
3.3.1.3.1. Git 16
3.3.1.3.2. GitHub 16
3.3.1.3.3. GitLab 16
3.3.1.4. IDE 17
3.3.1.4.1. Visual Studio Code 17
3.3.1.4.2. Eclipse 17
3.3.1.4.3. IntelliJ IDEA 17
3.3.1.5. Il·lustracions 17
3.3.1.5.1. Inkscape 18
3.3.1.5.2. Adobe Illustrator 18
3.3.1.5.3. CorelDRAW 18
3.3.2. Eines seleccionades 18
3.3.2.1. Desenvolupament de jocs web en RV 18
3.3.2.2. Models 3D 18
3.3.2.3. Gestió de versions 19
3.3.2.4. IDE 19

3.3.2.5. Il·lustracions 19
3.3.3. Disseny gràfic 19
3.3.3.1. Ambientació i il·luminació 19
3.3.3.2. Estil gràfic 19
3.3.3.3. Interfície d'usuari 20
3.3.3.3.1. Controls 20
3.3.3.3.2. Enemics 21
3.3.4. Cicle jugable 22
3.3.5. Model de dades 23
3.3.6. Disseny arquitectònic 23
3.4. Implementació 24
3.4.1. Servidor 24
3.4.1.1. Codi del servidor 25
3.4.1.2. Variables d'entorn 25
3.4.1.3. Accés a la base de dades 25
3.4.2. Base de dades 26
3.4.3. Client 27
3.4.3.1. A-Frame 27
3.4.3.2. Components 27
3.4.3.2.1. Arc 28
3.4.3.2.2. Corda 30
3.4.3.2.3. Fletxa 31
3.4.3.2.4. Planta 36
3.4.3.2.5. Vagoneta 37
3.4.3.2.6. Terra 38
3.4.3.2.7. Diana 39
3.4.3.3. Funcions auxiliars 40
3.4.3.3.1. controladorEnemics(tipus) 40
3.4.3.3.2. generadorModals(tipus) 40
3.4.3.3.3. actualitzarVidaPunts() 42

3.4.3.3.4. partidaAcabada() 42
3.4.3.3.5. Funcions AJAX 42
3.4.3.4. Controls 43
3.4.3.5. Inici de sessió 45
3.4.3.6. Sons 45
3.5. Instal·lació 46
Capítol 4. Resultats 49
Capítol 5. Conclusions 53
5.1. Objectius 53
5.2. Possibles millores 54
5.3. Opinió personal 54
Referències 55
Apèndix A. 57
A.1. Fonts de l'especificació de l'arc 57
Apèndix B. 59
B.1. Arc 59
B.2. Fletxa 59
B.3. Planta 59
B.4. Vagoneta 59
B.5. Arbres 59
B.6. Diana 59
B.7. Cel 60
B.8. So partida perduda 60
B.9. So arrancar fruita 60
B.10. So perdre vida 60
B.11. So botó pressionat 60
B.12. So fletxa disparada 60
B.13. So col·lisió amb diana 60
B.14. Música de fons del menú 61
B.15. Música de fons de la partida 61

Índex de figures

Figura 2.1 Royal Revolt! 5
Figura 2.2 Bloons TD 6 6
Figura 2.3 Temple run 7
Figura 2.4 The Lab - Longbow 8
Figura 3.1 Planificació temporal 11
Figura 3.2 Comandament de RV 20
Figura 3.3 Planta alienígena 21
Figura 3.4 Cicle de jugabilitat 22
Figura 3.5 Diagrama arquitectònic 24
Figura 3.6 Model 3D de l'arc 29
Figura 3.7 Exemple de *gimbal lock* 29
Figura 3.8 Corba Catmull-Rom de three.js 30
Figura 3.9 Model 3D de la fletxa 32
Figura 3.10 Color de la fruita segons la vida 36
Figura 3.11 Model 3D de la vagoneta 37
Figura 3.12 Model 3D del camí inicial 38
Figura 3.13 Model 3D dels camins subseqüents 39
Figura 3.14 Model 3D de la diana 39
Figura 3.15 Modal del menú 40
Figura 4.1 Resultat final 49
Figura 4.2 Presentació a Ciència per a tothom 50

Índex de taules

Taula 2.1 Característiques dels diferents jocs 9
Taula 3.1 Model de dades 23
Taula 3.2 Sons usats 46

Nomenclatura

LTIM	Laboratori de Tecnologies de la Informació Multimèdia.
UIB	Universitat de les Illes Balears.
HTML	Hypertext Markup Language.
A-Frame	*Framework* usat per a la realització de la major part del projecte.
RV (VR)	Realitat virtual (Virtual Reality).
Tower defense	Gènere de videojocs en el qual es basa el projecte.

Resum

Aquest treball de fi de grau té com a objectiu el desenvolupament d'un joc en realitat virtual. L'objectiu és crear una experiència immersiva on els jugadors utilitzen un arc per a disparar a diversos enemics mentre segueixen un camí. Aquest enfocament combina la precisió i l'habilitat del tir amb arc amb l'emoció d'enfrontar-se a enemics en un entorn virtual. A mesura que els jugadors avancen pel camí, es troben amb diferents tipus d'enemics, cadascun amb les seves pròpies característiques i nivells de dificultat, la qual cosa afegeix varietat i desafiament al joc.

La utilització de diferents enemics assegura que els jugadors es mantinguin en moviment constant, la qual cosa augmenta la intensitat i la dinàmica del joc. La combinació de la tecnologia de realitat virtual amb el tir amb arc ofereix una experiència única i atractiva, que pot atreure tant als entusiastes dels videojocs com a aquells interessats en l'esport del tir amb arc.

El joc permet als jugadors interactuar amb un arc i unes fletxes per practicar el tir amb arc i entretenir-se disparant diferents enemics utilitzant les oportunitats que ofereix la realitat virtual.

El desenvolupament del projecte s'ha basat en una metodologia iterativa. La implementació del joc ha implicat l'ús de tecnologies com HTML, JavaScript, MariaDB i Node.js, així com la implementació de models 3D i entorns interactius amb A-Frame.

El projecte s'ha realitzat amb èxit, a més a més, s'ha presentat de manera accessible als assistents de la fira "Ciència per a tothom" de la UIB a on els va agradar molt als participants i es va poder obtenir retroalimentació per millorar el projecte.

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 dut a realitzar-lo, els objectius que es volen aconseguir, l'abast del projecte i l'estructura del document.

1.1. Contextualització

La Realitat Virtual (RV) ha generat molt d'interès recentment gràcies a l'aparició de multitud de dispositius per reproduir-la, com poden ser les Oculus Rift, Meta Quest, HTC Vive, Samsung Gear, Google Daydream, etc. Fins i tot han aparegut dispositius per a ser fets servir amb mòbils, arribant a les econòmiques Google Cardboard, que estan fetes de cartró.

El joc que es desenvoluparà en aquest TFG serà un joc d'atac a la torre invers, és una variant dels jocs del tipus *tower defense*[1], és a dir un joc a on enemics van avançant per destruir la torre que s'ha de defensar, en el qual en lloc de col·locar i millorar torres o estructures per evitar que les diferents ones d'enemics arribin a la base, el jugador assumeix el rol de l'atacant.

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. Adaptació

El gènere de jocs de *tower defense* solen ser una bona proposta com a joc, ja que són ambdós senzills d'aprendre i desafiants, però a l'hora de traslladar aquest tipus de jocs a un entorn de realitat virtual resulta un aprofitament molt pobre de les possibilitats que habiliten les ulleres de RV, degut a que funcionen molt bé en un tauler 2D.

En vers de fer un *tower defense* habitual es pot fer a la inversa, és a dir, el protagonista és el que ha d'anar superant les defenses. Aquesta proposta pot aprofitar millor les possibilitats de les ulleres RV, ja que es pot fer un joc molt més interactiu perquè pot ser en primera persona i la utilització d'armes de projectils, com un arc fa que s'aprofitin bé les possibilitats dels controls amb les mans.

1.3. Objectius

L'objectiu principal és desenvolupar un joc en primera persona a on el jugador avança de manera automàtica per un camí i s'enfronta a diferents tipus d'enemics

[1] https://ca.wikipedia.org/wiki/Tower_Defense

els quals haurà d'eliminar. 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'acaba quan al jugador no li quedin vides.

El joc està pensat per ser jugat només amb equipament de RV.

De cada partida es recopilarà informació per poder elaborar una taula de puntuacions que mostri els 10 millors jugadors. Aquesta taula ajudarà a crear una competició entre els jugadors per estar-hi de la mateixa manera que en els marcadors de millors jugadors en jocs competitius o el menor temps assolit per pilots en un circuit.

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 s'ha de dur una gestió del calendari adequada.

L'objectiu principal del joc consisteix en avançar per un camí infinit generat dinàmicament i anar superant les defenses enemigues, a l'estil *endless runner*[2], és a dir un joc de córrer sense fi, però amb l'afegit d'anar disparant als enemics. A mesura que el jugador avanci i elimini més enemics, la dificultat del joc augmentarà progressivament, fent que cada secció sigui més desafiant que l'anterior.

La puntuació principal del joc es mesurarà amb el màxim nombre d'enemics eliminats, demostrant habilitat i estratègia en un entorn virtual cada vegada més hostil.

1.4. Abast del projecte

El joc consistirà sobretot en la demostració del que es podria fer amb un equip de realitat virtual més que un joc amb característiques especials, sobretot per les limitacions de personal i coneixements en la creació de videojocs, sinó més aviat una prova del que es pot fer amb la RV junt amb diferents tecnologies web. Per tant, moltes característiques vistes en jocs similars, no estaran presents o se tractaran de manera més superficial en aquest projecte. Especialment en l'àmbit del modelatge i animacions 3D.

Tampoc s'espera una gran varietat de models 3D. Ni en l'àmbit d'armament ni en el dels enemics i defenses. 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. Així com tampoc hi haurà destrucció de l'entorn quan el jugador dispari a les parets de l'escenari.

La trajectòria del tir es veurà afectada per la gravetat, ja que com es dispara amb un arc té sentit i és natural que la fletxa vagi caient segons va recorrent metres.

[2] https://es.wikipedia.org/wiki/Corredor_sin_fin

Encara que no hi haurà dispersió, perquè no se cerca una simulació completa de les físiques i molts jugadors ho trobarien frustrant.

Els enemics quedaran estàtics a la posició a on apareguin o tinguin assignada, ja que implementar un algorisme de *pathfinding* o similar no és l'objectiu del projecte, i la dificultat del joc resideix sobretot en l'ús d'un arc i no s'afegirà dificultat fent que els enemics es moguin.

S'elaboraran dues versions. Una per PC i un altre per equipament de RV, els quals accediran a través del navegador. Ara bé, dispositius mòbils, tauletes, etc. no tindran 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, solucions i idees ja existents a problemes similars que poden resultar útils i poden millorar el resultat del projecte. El tercer capítol és Desenvolupament del projecte format per tots els apartats de gestió, anàlisi, disseny, implementació i instal·lació, és a on es concentra la major part de la realització del projecte. El quart capítol és Resultats on es comenten els productes obtinguts amb els objectius proposats. El cinquè i darrer capítol Conclusions conté les diferents conclusions que s'han obtingut, possibles millores per al futur i una opinió personal.

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 aquest projecte.

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. Jocs similars existents

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

2.1.1. Royal Revolt!

Royal Revolt![3] és un joc de *tower defense* invers en el qual el protagonista és un príncep que torna al seu regne, però descobreix que el tron del seu pare ha estat usurpat.

La missió principal del joc és liderar soldats per recuperar el regne. Per a aconseguir això, s'han de liderar a les diferents tropes a través d'una sèrie de nivells, cadascun amb un castell enemic que s'ha de conquistar.

A mesura que s'avança s'aconsegueixen diners per millorar tant el príncep com les diferents tropes i màgies que es poden invocar.

Figura 2.1 Royal Revolt!

[3] https://play.google.com/store/apps/details?id=com.flaregames.royalrevolt&hl=en-US

En la Figura 2.1 podem veure al príncep junt amb les seves tropes avançant pel territori hostil. Al cantó inferior esquerre es veuen els icones de les diferents tropes que es poden cridar i al cantó inferior dret les 2 màgies que es poden invocar.

En resum, el joc combina estratègia i acció, requerint l'ús tant de la força bruta com de la planificació tàctica per a avançar.

2.1.2. Bloons TD 6

Bloons TD 6[4] és un *tower defense* el qual l'objectiu és evitar que els globus arribin al final del camí. Per a això, s'han de col·locar estratègicament torres de micos i herois perquè destrueixin els globus abans que arribin al seu destí.

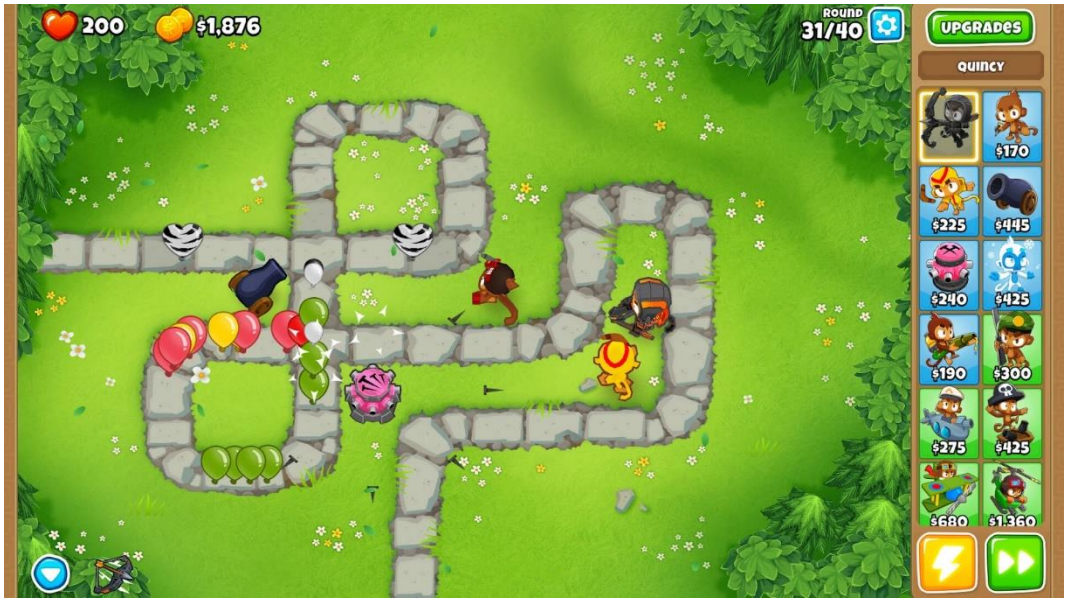

Figura 2.2 Bloons TD 6

Cada mico té una arma diferent i funciona especialment bé sobre un tipus de globus. La idea és cobrir tots els tipus de globus amb el menor nombre de micos i anar-los millorant perquè siguin més eficaços. A la Figura 2.2 es poden veure els diferents tipus de micos i globus, com es pot apreciar, els globus tenen diferents colors i formes, el color indica la quantitat de capes que li queden abans d'explotar i la forma indica el tipus, com per exemple, que sigui invisible a certs tipus de micos.

2.1.3. Temple run

Temple run[5] és un *endless runner*, que ràpidament es va convertir en un èxit degut a la seva jugabilitat addictiva i senzilla.

[4] https://store.steampowered.com/app/960090/Bloons_TD_6/
[5] https://play.google.com/store/apps/details?id=com.imangi.templerun&hl=en-US

L'objectiu és recórrer el màxim nombre de metres evitant obstacles, a mesura que es recorren metres, de manera progressiva, el personatge corr de cada vegada més ràpid i s'afegeixen més diferències en els obstacles, necessitant de cada vegada una reacció més ràpida. Si comets una errada el personatge corr més lent i permet que se li apropin els enemics, si comets una altra errada en poc temps aconsegueixen atrapar-lo.

Figura 2.3 Temple run

A la Figura 2.3 es poden veure les criatures malignes que segueixen al personatge principal, qualque obstacle i diferents biomes que es recorren.

2.1.4. The Lab – Longbow

The Lab[6] és un videojoc de realitat virtual desenvolupat per Valve. El joc ofereix vuit tipus de minijocs diferents que impliquen experiències de demostració curtes que utilitzen diferents aspectes de les capacitats de la realitat virtual. La varietat també s'ofereix més enllà de les mateixes experiències per la quantitat d'interacció amb objectes de l'entorn que s'inclou.

Més específicament la demostració anomenada *Longbow* un joc basat en tir amb arc. Els dos controladors de moviment de RV s'utilitzen per apuntar i disparar un arc, amb un controlador que actua com l'eix d'arc i l'altre com la fletxa. El jugador té la tasca de defensar una porta del castell contra una força invasora mentre s'està sobre la murada del castell.

La característica especial d'aquest joc és la interacció amb l'arc i les fletxes a través dels comandaments de RV.

[6] https://store.steampowered.com/app/450390/The_Lab/

A la Figura 2.4 es poden veure els enemics a la part superior de la imatge i la porta del castell a l'esquerra a part de diferents elements distribuïts per l'entorn que donen punts extra o activen trampes.

Figura 2.4 The Lab - Longbow

2.2. Característiques interessants

Després d'haver vist diferents jocs que assoleixen una part del que es vol fer, ara es valorarà que fan bé i encaixaria amb el joc que es vol fer.

La Taula 2.1 mostra el joc d'origen les característiques i una valoració de si encaixaria en el nostre joc.

Joc o jocs	Característica	Valoració
Royal Revolt!	Anar millorant el personatge a mida que es puja de nivell.	No
Royal Revolt!	Anar invocant tropes mentre s'avança dins un nivell.	No
Royal Revolt! i **Bloons TD 6**	Utilització de màgies o habilitats especials.	No
Bloons TD 6	Un tipus de mico és especialment fort contra cert tipus de globus.	Sí
Temple run	El jugador avança pel mapa fins que comet 2 errades.	Sí

Temple run	El jugador és perseguit per un enemic que limita les errades que pot fer.	No
The Lab - Longbow	Utilització d'un arc amb els comandaments de RV.	Sí
The Lab - Longbow	Elements distribuïts pel mapa que donen punts extra.	Sí
Bloons TD 6 i The Lab - Longbow	Utilització del mateix enemic, però amb modificacions que li donen més vida o armadura.	Sí

Taula 2.1 Característiques dels diferents jocs

2.2.1. Característiques escollides

Finalment, tenim un conjunt de característiques que sabem que fan bé altres jocs. Aquestes característiques s'han de modificar per saber de quina manera adaptar-les al nostre.

- Un tipus de personatge és especialment fort contra cert tipus d'enemic: això es podria adaptar posant diferents tipus de fletxes com fletxes explosives o de foc.
- El jugador avança pel mapa fins que comet 2 errades: Es podria posar un sistema de vida en el qual si els enemics superen una certa distància, en perds una.
- Utilització d'un arc amb els comandaments de RV: es pot implementar de la mateixa manera o molt similar.
- Elements distribuïts pel recorregut que donen punts extra: es pot implementar de la mateixa manera.
- Utilització del mateix enemic, però amb modificacions que li donen més vida o armadura: es pot implementar el fet que els enemics tinguin més vida a mida que es puja de ronda.

Capítol 3. Desenvolupament del projecte

El desenvolupament d'un projecte de joc és un procés integral que abasta diverses etapes, cadascuna de les quals és crucial per a garantir la qualitat i l'èxit del producte final. En aquest apartat, es detallaran les fases fonamentals que componen el desenvolupament del joc, incloent-hi la gestió del projecte, l'anàlisi de requisits, el disseny del joc i la implementació.

3.1. Gestió del projecte

Aquest apartat descriu la metodologia de desenvolupament que s'ha seguit junt amb la planificació temporal.

3.1.1. Metodologia de desenvolupament

En el desenvolupament d'aquest projecte s'ha utilitzat la metodologia de prototip evolutiu. Aquest mètode consisteix en crear un prototipus inicial el qual va evolucionant de manera incremental afegint funcionalitats, el producte final és el prototipus inicial amb totes les funcionalitats requerides.

Aquesta metodologia és particularment efectiva en el desenvolupament àgil a on la retroalimentació i les iteracions són parts clau, a més de ser altament adaptable gràcies al desenvolupament incremental sobre el qual està basat.

3.1.2. Planificació temporal

A la Figura 3.1 es pot veure la planificació del projecte que està distribuïda entre les diferents funcionalitats a grans trets.

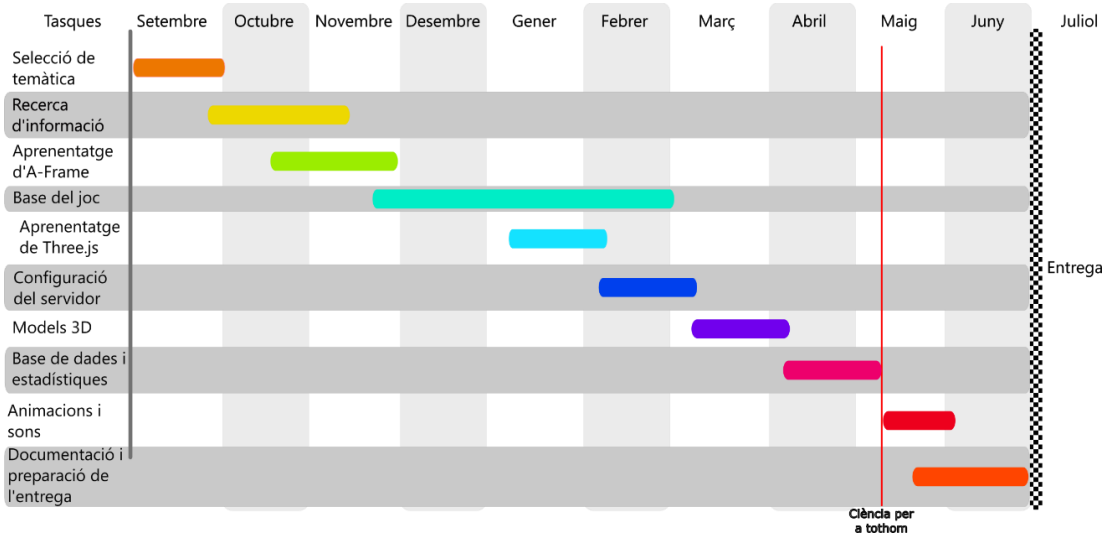

Figura 3.1 Planificació temporal

3.2. Anàlisi

En aquest apartat es detallaran els requisits tant funcionals com no funcionals i l'usuari que interactuarà amb el joc.

3.2.1. Usuaris

El joc compta amb un únic tipus d'usuari, el jugador, el qual interactuarà amb l'arc i els diferents elements del joc. Durant el joc, el jugador podrà agafar l'arc i anar disparant fletxes utilitzant els controls de realitat virtual. A més, es guardarà la seva puntuació dins la base de dades.

3.2.2. Requisits funcionals

Es pot distingir entre els requisits funcionals d'usuari que indiquen les accions que ha de poder fer un usuari i els de sistema que especifiquen que ha de fer el sistema.

3.2.2.1. Requisits funcionals d'usuari

- RFU-01 · Agafar un arc
  - El jugador ha de poder agafar i utilitzar un arc.
- RFU-02 · Moviment
  - El jugador s'ha de poder moure lliurement per la plataforma encara que no pugui sortir.
- RFU-03 · Disparar fletxa
  - El jugador ha de poder disparar fletxes en qualsevol direcció fent ús de l'arc.
- RFU-04 · Modes de joc
  - Abans de començar la partida, l'usuari ha de poder elegir entre el mode normal i un mode de pràctica.
- RFU-05 · Puntuació
  - El jugador ha de poder veure la vida restant i puntuació actual mentre juga.
- RFU-06 · Marcador
  - El jugador ha de poder veure un marcador amb les 10 millors puntuacions obtingudes pels jugadors.

3.2.2.2. Requisits funcionals de sistema

- RFS-01 · Recorregut del camí
  - El camí es recorrerà damunt una plataforma que es mourà de manera automàtica un cop engegada.
- RFS-02 · Sons
  - Han d'existir efectes de so que es reproduiran quan es realitzin certes accions com disparar, ferir un botó o un enemic...
- RFS-03 · Puntuacions
  - El joc ha de guardar la puntuació obtinguda si és major a l'anterior a cada punt de control i quan es quedi sense vides.

- RFS-04 · Generació del camí
    - El camí a recórrer es generarà dinàmicament mentre l'usuari vagi avançant.
- RFS-05 · Descans
    - El jugador ha de tenir un punt de control com a màxim cada 120 segons per poder descansar degut al cansament dels braços.

3.2.3. Requisits no funcionals

- RNF-01 · Immersió
    - S'ha de poder simular el tensament de la corda amb comandaments de RV, podent modificar la força aplicada segons la separació de les mans. A més a més, el comportament de la fletxa ha de seguir la trajectòria d'un projectil com si fos a la Terra, és a dir tenint en compte la gravetat terrestre.
- RNF-02 · Compatibilitat
    - El joc ha de ser compatible amb la majoria de dispositius de realitat virtual, però com a mínim amb l'Oculus i el Meta Quest.
- RNF-03 · Fluïdesa
    - El joc ha d'anar com a mínim a 30 fotogrames per segon.
- RNF-04 · Integritat
    - La puntuació de cada usuari s'ha de guardar dins una base de dades i només pot canviar la seva una vegada ha iniciat sessió.
- RNF-05 · Compatibilitat
    - El joc serà compatible amb tots els navegadors que suportin la API de WebXR.
- RNF-06 · Ús de dades
    - Només es recopilaran les següents dades: nom, edat, sexe, experiència prèvia en realitat virtual, nombre de disparaments, nombre d'encerts, puntuació i ronda. Aquestes dades només s'usaran amb fins estadístics.

3.3. Disseny

En aquest apartat, s'explica el disseny gràfic i l'atmosfera escollida, l'estil de la interfície d'usuari, el programari utilitzat, el disseny de la jugabilitat i el model de dades usat per a la realització del joc.

3.3.1. Eines

En aquest apartat, s'exploren diverses eines que es poden emprar per a la realització del projecte.

3.3.1.1. Desenvolupament de jocs web en RV

Aquí explorarem les diverses eines disponibles per al desenvolupament de jocs web en realitat virtual. A mesura que la tecnologia avança, sorgeixen múltiples plataformes i *frameworks* que faciliten la creació d'experiències immersives i atractives. Analitzarem com cadascuna d'aquestes eines contribueix al desenvolupament de jocs en realitat virtual, destacant les seves característiques, avantatges i casos d'ús.

3.3.1.1.1. Three.js

Three.js[7] és una biblioteca de JavaScript que permet crear i renderitzar gràfics 3D en un navegador web utilitzant WebGL. Proporciona una sèrie d'eines i funcionalitats que faciliten el desenvolupament d'escenes 3D, incloent-hi la creació de geometries, materials, llums, ombres i animacions.

3.3.1.1.2. A-Frame

A-Frame[8] és un *framework* web dissenyat per crear experiències de realitat virtual (RV). Està construït sobre HTML i JS, el que facilita la creació d'escenes virtuals. No obstant això, A-Frame va més enllà de ser simplement un gràfic d'escena 3D o un llenguatge de marques; en el seu nucli, presenta un sistema sòlid d'entitats-components que ofereix una arquitectura declarativa, flexible i modular per a *Three.js*.

3.3.1.1.3. WonderlandEngine

WonderlandEngine [9] és una plataforma de desenvolupament d'aplicacions i experiències en realitat virtual (RV) i realitat augmentada (AR) que permet als desenvolupadors crear entorns 3D interactius de manera eficient. Aquesta eina se centra en la creació de contingut web, cosa que significa que les aplicacions desenvolupades amb WonderlandEngine poden executar-se directament en navegadors sense necessitat d'instal·lacions addicionals, gràcies a la utilització de les APIs de WebGL [10] i WebXR [11] per generar el contingut i proporcionar la compatibilitat entre els navegadors.

3.3.1.2. Objectes i models 3D o animacions

Com que no es tracta d'un projecte de disseny 3D i que tampoc es tenen els coneixements 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 amb llicències que ens permetin utilitzar-los per a la realització d'aquest projecte.

[7] https://threejs.org/
[8] https://aframe.io/
[9] https://wonderlandengine.com/
[10] https://www.khronos.org/webgl/
[11] https://immersive-web.github.io/webxr/

Diferents llocs web de on s'han agafat els objectes:

- Models 3D: poly.pizza[12]
    - Poly.pizza és una plataforma que permet als usuaris crear i compartir experiències interactives en 3D i realitat virtual. Se centra en la creació d'entorns virtuals *low poly* que poden ser explorats i manipulats pels usuaris, facilitant la creació de contingut immersiu i col·laboratiu.
- Textures: polyhaven[13]
    - Poly Haven és una plataforma en línia que proporciona recursos gratuïts d'alta qualitat per a artistes i desenvolupadors que treballen en gràfics 3D i visualització. Ofereix una àmplia varietat d'actius, incloent-hi models 3D, textures i HDRIs (imatges de rang dinàmic alt) que poden ser utilitzats en projectes de disseny, animació i videojocs.
- Sons: Creative Commons[14]
    - Creative Commons disposa d'una eina de cerca que permet als usuaris trobar contingut amb llicència oberta i lliure de drets d'autor en diverses plataformes i repositoris en línia. Aquesta eina facilita la cerca d'imatges, vídeos, música i altres tipus de contingut que es poden utilitzar i compartir legalment, sempre que es respectin les condicions de les llicències corresponents.

A part d'aquests continguts també s'hauran de crear continguts propis o modificar els de tercers, per tant, caldrà utilitzar algun programa per modificar models 3D.

3.3.1.2.1. Blender

Blender[15] és un programari de codi obert i gratuït utilitzat per a la creació de gràfics en 3D. Ofereix una àmplia gamma d'eines i funcionalitats que permeten als usuaris modelar, esculpir, texturar, il·luminar i renderitzar objectes i escenes en tres dimensions. Aquest programa es farà servir per a la modificació i creació d'animacions sobre els models 3D definits en el punt anterior.

Entre les seves característiques més destacades es troben un potent motor de renderitzat, eines avançades de modelatge i escultura, capacitats d'animació, així com suport per a simulacions físiques i de fluids. Blender també compta amb una comunitat activa que contribueix al desenvolupament del programari i ofereix una àmplia gamma de tutorials i recursos en línia. La seva flexibilitat i versatilitat ho han convertit en una opció popular tant per a principiants com per a professionals en el camp de la creació de contingut en 3D.

[12] https://poly.pizza/
[13] https://polyhaven.com/
[14] https://search.creativecommons.org/
[15] https://www.blender.org

3.3.1.2.2. Maya

Maya [16] és un programari de modelatge, animació i renderitzat en 3D desenvolupat per Autodesk. És àmpliament utilitzat en la indústria del cinema, la televisió, els videojocs i la visualització arquitectònica. Maya ofereix una àmplia gamma d'eines i característiques que permeten als artistes i dissenyadors crear models tridimensionals detallats, animacions complexes i efectes visuals.

Entre les seves principals funcionalitats s'inclouen el modelatge poligonal, l'escultura digital, l'animació de personatges, la simulació de fluids i partícules, així com un potent motor de renderitzat. A més, Maya és conegut per la seva flexibilitat i capacitat de personalització, la qual cosa permet als usuaris adaptar el programari a les seves necessitats específiques mitjançant *scripts* i *plugins*.

3.3.1.3. Gestió de versions

La gestió de versions és un procés que permet controlar i rastrejar els canvis realitzats en documents, arxius o projectes al llarg del temps. Aquest enfocament és especialment útil en el desenvolupament de programari, on múltiples versions d'un codi poden existir simultàniament i on és crucial mantenir un registre de les modificacions per a facilitar el control dels canvis i la col·laboració entre diferents desenvolupadors.

3.3.1.3.1. Git

Git [17] és un sistema de control de versions distribuït que permet als desenvolupadors gestionar i realitzar un seguiment dels canvis en el codi font d'un projecte al llarg del temps. Permet proporcionar un historial detallat de les modificacions realitzades, la possibilitat de revertir canvis i la utilització de branques, la qual cosa possibilita el desenvolupament de noves característiques o correcció d'errors de manera aïllada. Va ser creat per Linus Torvalds en 2005 i s'ha convertit en una de les eines més populars per al desenvolupament de programari.

3.3.1.3.2. GitHub

GitHub[18] és una plataforma de desenvolupament col·laboratiu que utilitza Git com a sistema de control de versions. Permet als desenvolupadors emmagatzemar, gestionar i compartir el seu codi font en línia, facilitant la col·laboració en projectes de programari. Fundada en 2008, GitHub s'ha convertit en un dels serveis més populars per a l'allotjament de repositoris de codi.

3.3.1.3.3. GitLab

GitLab[19] és una plataforma integral que no sols permet la gestió de codi font, sinó que també proporciona eines per a la col·laboració, l'automatització de processos

[16] https://www.autodesk.com/products/maya/overview
[17] https://en.wikipedia.org/wiki/Git
[18] https://github.com/
[19] https://about.gitlab.com/

i la gestió de projectes. Igual que GitHub, GitLab utilitza Git com a sistema de control de versions.

3.3.1.4. IDE

Un IDE, o Entorn de Desenvolupament Integrat (per les seves sigles en anglès, Integrated Development Environment), és un programari que proporciona als desenvolupadors un conjunt d'eines i funcionalitats per a facilitar el procés de desenvolupament de programari. Un IDE combina diverses eines en una única interfície, la qual cosa permet als programadors escriure, depurar i compilar codi de manera més eficient.

3.3.1.4.1. Visual Studio Code

Visual Studio Code (VS Code) [20] és un editor de codi font desenvolupat per Microsoft que s'ha convertit en una de les eines més populars entre desenvolupadors de programari. És un editor lleuger i potent (a través de plugins) que ofereix una àmplia gamma de característiques i funcionalitats que faciliten el desenvolupament d'aplicacions en diversos llenguatges de programació.

3.3.1.4.2. Eclipse

Eclipse [21] és un entorn de desenvolupament integrat (IDE) de codi obert que s'utilitza principalment per al desenvolupament de programari en diversos llenguatges de programació, encara que és especialment popular per a Java. Originalment creat per IBM, Eclipse ha evolucionat al llarg dels anys i ara és mantingut per l'Eclipse Foundation, que gestiona una comunitat activa de desenvolupadors i contribuents.

3.3.1.4.3. IntelliJ IDEA

IntelliJ IDEA[22] és un entorn de desenvolupament integrat (IDE) desenvolupat per JetBrains, dissenyat principalment per al desenvolupament d'aplicacions en Java, encara que també ofereix suport per a altres llenguatges de programació com Kotlin, JavaScript, TypeScript, i més. És conegut pel seu enfocament en la productivitat del desenvolupador i la seva àmplia gamma de característiques avançades les quals fan que sigui un IDE dels més potents.

3.3.1.5. Il·lustracions

El programari d'il·lustració serveix per a crear i editar gràfics, imatges i dissenys visuals.

[20] https://code.visualstudio.com/
[21] https://eclipseide.org/
[22] https://www.jetbrains.com/idea/

3.3.1.5.1. Inkscape

Inkscape[23] és un programari de disseny gràfic de codi obert i gratuït que s'utilitza principalment per a la creació de gràfics vectorials. És una alternativa popular a programes comercials com l'Adobe Illustrator. Inkscape és conegut per la seva àmplia gamma d'eines i característiques que permeten als usuaris crear i editar il·lustracions, logotips, diagrames i altres tipus de gràfics.

3.3.1.5.2. Adobe Illustrator

Adobe Illustrator [24] és un programari de disseny gràfic i il·lustració vectorial desenvolupat per Adobe. És àmpliament utilitzat per dissenyadors, il·lustradors i artistes per a crear gràfics, logotips, tipografies, il·lustracions i altres elements visuals.

3.3.1.5.3. CorelDRAW

CorelDRAW [25] és un programari de disseny gràfic i il·lustració vectorial desenvolupat per Alludo. És àmpliament utilitzat per dissenyadors gràfics, il·lustradors i professionals de la impressió per a crear una varietat de projectes visuals, com a logotips, fullets, cartells, il·lustracions entre altres.

3.3.2. Eines seleccionades

Un cop vistes les diferents eines per a la realització del projecte, s'explicarà quines eines s'han seleccionat junt amb els diferents motius.

3.3.2.1. Desenvolupament de jocs web en RV

S'ha optat per utilitzar **A-Frame** ja que:

- Utilitza la biblioteca de Three.js la qual té molta comunitat, el que es tradueix en major documentació o pàgines de suport amb més informació.
- Proporciona un nivell d'abstracció sobre Three.js el qual fa més fàcil desenvolupar amb aquesta eina, a més de permetre poder accedir i modificar els components generats amb Three.js.
- WonderlandEngine reconeix que A-Frame és una de les millors eines en un article [1].

3.3.2.2. Models 3D

S'ha utilitzat el programa **Blender** perquè:

- Existeix una gran comunitat, per tant, hi ha molta documentació i tutorials disponibles.
- És gratuït i de codi obert i compatible amb Linux.

[23] https://inkscape.org/
[24] https://www.adobe.com/es/products/illustrator.html
[25] https://www.coreldraw.com/en/

- Ja s'havia utilitzat en el passat encara que només fos per crear uns models molt senzills.

3.3.2.3. Gestió de versions

Com que s'ha utilitzat **GitHub** durant tot el grau, ja se sabia com funciona i ja es tenia un compte fet, s'ha seleccionat aquest per comoditat i perquè les diferents eines aconsegueixen el mateix propòsit quasi de la mateixa manera.

3.3.2.4. IDE

S'ha elegit l'IDE de **IntelliJ IDEA** perquè:

- Ofereix un gran suport per JavaScript i HTML, els dos llenguatges principals del projecte.
- És un dels IDEs més potents.
- Es té una llicència gràcies al programa d'estudiants al qual la UIB està adherida.
- Per comoditat perquè ja s'havia usat.

3.3.2.5. Il·lustracions

S'ha elegit **Inkscape** degut als mateixos motius que Blender.

- Existeix una gran comunitat, per tant, hi ha molta documentació i tutorials disponibles.
- És gratuït, de codi obert i compatible amb Linux.
- Ja s'havia utilitzat en el passat.

3.3.3. Disseny gràfic

En aquest apartat, analitzarem com aquests elements visuals no sols contribueixen a l'estètica del joc, sinó que també influeixen en la jugabilitat i en l'experiència general de l'usuari.

3.3.3.1. Ambientació i il·luminació

L'ambientació i la il·luminació són elements crucials que determinen l'atmosfera i l'experiència del jugador. L'elecció d'una il·luminació brillant i natural és fonamental per a establir un ambient positiu, i per això s'ha optat per simular un dia clar amb molta llum solar.
Quant a la disposició dels elements en l'escena, un disseny net i organitzat, amb un enfocament en la simplicitat, ajuda a minimitzar distraccions i permet que el jugador es pugui concentrar fàcilment.

3.3.3.2. Estil gràfic

S'ha optat per l'estil *low poly*[26] en el disseny gràfic a causa de la seva notable eficiència de rendiment, especialment considerant l'alt consum de recursos que

[26] https://en.wikipedia.org/wiki/Low_poly

implica la realitat virtual. Aquest enfocament és particularment rellevant quan s'utilitzen ulleres de realitat virtual com les Meta Quest, les quals fan servir processadors ARM similars als d'un dispositiu mòbil. A més, l'ús de tecnologies web actuals impedeix aprofitar al màxim el potencial del maquinari disponible, la qual cosa fa que la reducció de la complexitat dels models i textures sigui encara més beneficiosa. Gràcies a adoptar aquest estil, s'aconsegueix una experiència més fluida i accessible, que permet que els usuaris gaudeixin d'entorns immersius sense comprometre la qualitat visual ni l'estabilitat del sistema.

3.3.3.3. Interfície d'usuari

En el disseny d'interfícies d'usuari per a experiències en realitat virtual, és fonamental evitar l'ús d'elements fixos pel fet que a moltes persones els hi provoca mareigs o qualque tipus de malestar. A més a més, poden limitar la immersió del jugador. En lloc d'això, és més convenient aprofitar l'entorn virtual per a integrar la interfície de manera dinàmica i contextual. En utilitzar objectes i espais de l'entorn com a punts d'interacció, es crea una experiència més natural.

Per tot això, s'ha optat per adoptar els mètodes d'interacció amb l'usuari directament utilitzant l'arc i les fletxes per disparar als botons. A més d'implementar la informació de la vida i la puntuació directament sobre la vagoneta sobre la qual existirà el jugador durant tota la sessió.

3.3.3.3.1. Controls

La manera principal d'interactuar amb el joc és a través dels comandaments de RV. Un comandament de realitat virtual és un dispositiu d'entrada que solen tenir un disseny ergonòmic i estan equipats amb diverses característiques que milloren l'experiència immersiva.

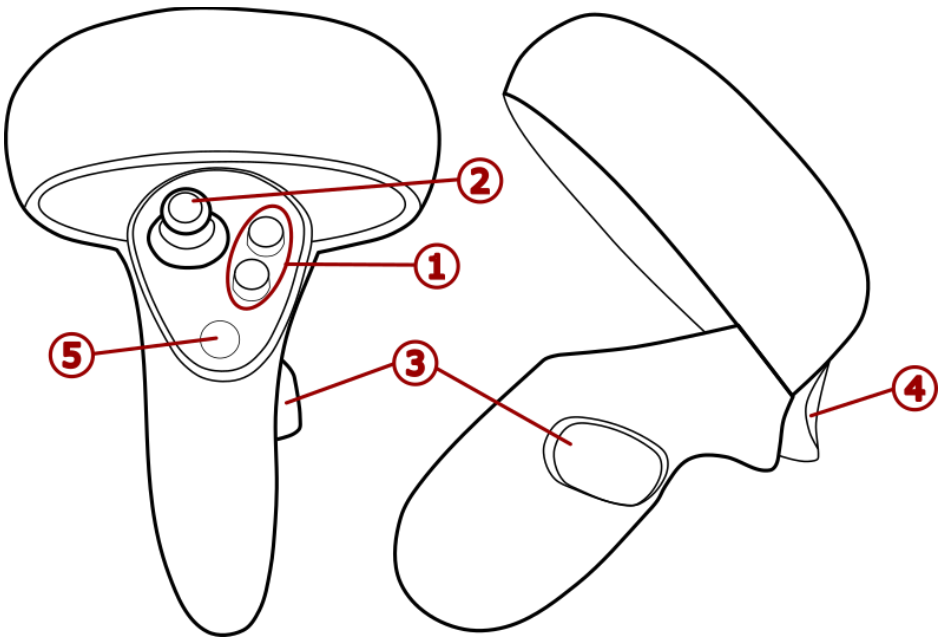

Figura 3.2 Comandament de RV

A partir de la Figura 3.2, s'explicaran les diferents maneres d'interacció.

1. Botons A, B (mà dreta) o X, Y (mà esquerra).
2. Palanca de moviment.
    a. Moviment del personatge en el comandament esquerra.
    b. Rotació de la càmera en el comandament dret.
3. Botó d'agafar, s'utilitza principalment per agafar els objectes virtuals.
4. Botó disparador, s'utilitza principalment per seleccionar elements.
5. Botó especial, aquests botons solen interactuar només amb el sistema operatiu de les ulleres.

Com el joc té interaccions bastant simples, només es fan servir els botons d'agafar i el disparador.

- El botó d'agafar serveix per agafar l'arc.
- El botó disparador serveix per seleccionar les lletres del teclat virtual, quan es té l'emissor de raigs, i per posar i tirar les fletxes de l'arc.

S'ha decidit utilitzar el botó disparador per les fletxes, en lloc del botó d'agafar, perquè en el joc mostrat anteriorment de The Lab – Longbow, fa servir aquest botó per la mateixa interacció i quan es va provar va resultar més còmode i intuïtiu aquest botó.

3.3.3.3.2. Enemics

Atès que el jugador ja es desplaça activament pel fet que està posicionat sobre una plataforma, és crucial que els enemics romanguin estàtics per a no complicar la mecànica de disparar amb l'arc.

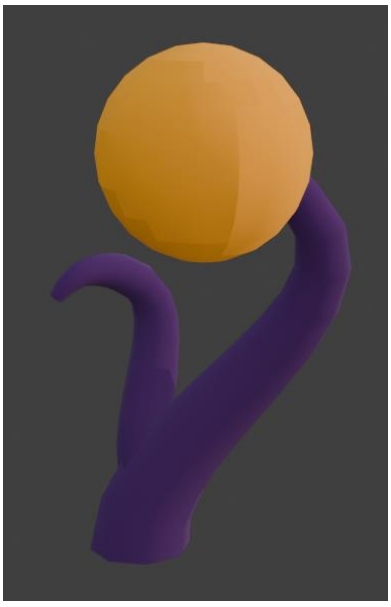

Figura 3.3 Planta alienígena

Això és especialment important en un joc dissenyat per a tots els públics, on l'accessibilitat i la diversió són primordials. Per aquesta raó, s'ha optat per introduir una espècie de planta alienígena que genera fruites en el seu extrem com es pot veure en la Figura 3.3. Els jugadors hauran d'apuntar i disparar a la fruita que es veu a la punta de la planta.

3.3.4. Cicle jugable

Una vegada el jugador inicií el joc, seguirà una sèrie de passes per començar una partida fins a acabar-la, les quals es mostren el diagrama de flux del cicle de jugabilitat en la Figura 3.4.

Per començar, el jugador no podrà iniciar la partida fins que s'hagi identificat amb el seu nom i l'edat. Una vegada hagi obert sessió, podrà agafar l'arc i se li mostrarà el menú per començar la partida o jugar en el mode de pràctica.

El mode de pràctica és simplement un entorn a on van sortint dianes a diferents altures a les quals el jugador haurà d'encertar. Aquest mode serveix principalment per familiaritzar-se amb l'ús de l'arc i la caiguda de la fletxa segons la força aplicada.

El mode normal consisteix a encertar a la fruita de la planta mencionada anteriorment per aconseguir punts i evitar perdre vida si el jugador s'allunya molt d'aquesta. Aquest cicle és infinit fins que el jugador es queda sense vides, una vegada el jugador es quedi sense, es guardaran en la base de dades les dades obtingudes durant la partida, establertes en el punt 3.3.5, i se li donarà l'opció d'iniciar una nova partida. Si així ho decideix, el cicle tornarà a començar.

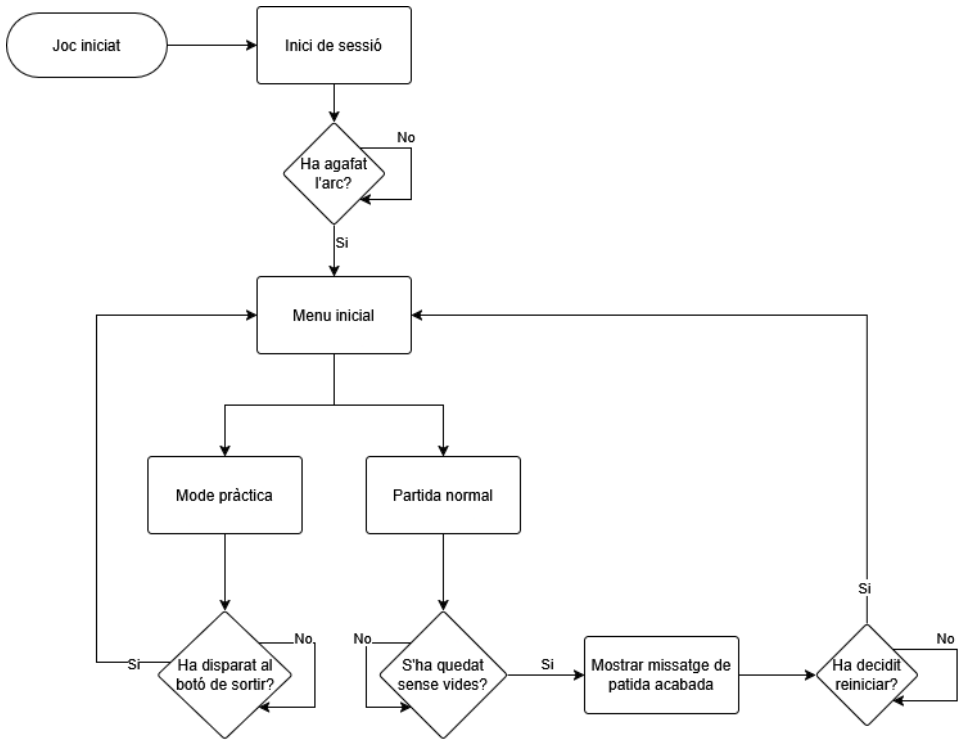

Figura 3.4 Cicle de jugabilitat

3.3.5. Model de dades

Atès que la quantitat de dades a emmagatzemar és limitada, el model de dades es presenta de manera senzilla i eficient. Aquest model es compon d'una única taula que agrupa tota la informació rellevant, la qual cosa facilita la seva gestió i accés. En centralitzar les dades en una sola estructura, s'optimitzen tant el rendiment com la simplicitat en les consultes, cosa que permet un maneig àgil i directe de la informació sense la necessitat de complicades relacions entre múltiples taules.

Nom	Tipus	Atributs
nom	VARCHAR	PK
edat	SMALLINT	PK
experiencia	BOOLEAN	
sexe	CHAR	
tirs	MEDIUMINT	UNSIGNED
encerts	MEDIUMINT	UNSIGNED
puntuacio	INT	UNSIGNED
ronda	SMALLINT	UNSIGNED

Taula 3.1 Model de dades

Com es pot veure a la Taula 3.1, les dades guardades són les següents:

- El nom i l'edat que introdueixi l'usuari servirà per identificar-lo, per això s'han definit com a claus primàries, això és perquè l'usuari pugui iniciar sessió i actualitzar la seva puntuació si aconsegueix una de més alta.
- Experiència serveix per saber si l'usuari tenia experiència prèvia en RV.
- Sexe és purament estadístic.
- Tirs i encerts són el nombre de tirs i encerts efectuats, respectivament, amb l'arc durant la partida.
- Puntuació és la puntuació total obtinguda al final de la partida.
- Ronda és la ronda en la qual ha perdut en aquesta partida.

3.3.6. Disseny arquitectònic

En aquest apartat, es presenta l'arquitectura del sistema. S'ha implementat com un model client-servidor, aquest model divideix el sistema en dues parts principals:

- El client: és la interfície a través de la qual l'usuari accedeix al sistema.

- El servidor: és qui dona servei a les peticions dels clients.

L'esquema de funcionament d'aquest model seria el següent:

1. El client realitza una petició al servidor.
2. El servidor rep la petició, la processa i es comunica amb la base de dades si és necessari.
3. El client rep la resposta del servidor, la processa i mostra el resultat si és necessari.

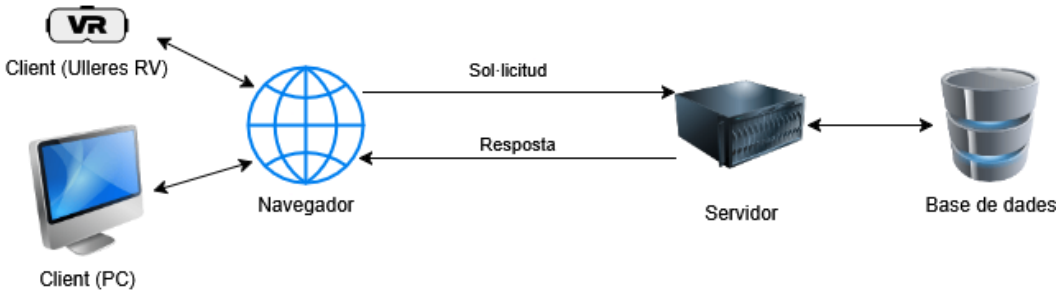

Figura 3.5 Diagrama arquitectònic

La Figura 3.5 mostra com s'ha implementat el model més específicament en aquest projecte. A on un client amb un ordinador o unes ulleres de realitat virtual independents utilitza el navegador per fer les sol·licituds al servidor, i aquest es pot comunicar amb la base de dades si fa falta, però el client no pot accedir a la base de dades directament tant per seguretat com per simplicitat.

3.4. Implementació

En l'apartat d'implementació, s'abordarà l'explicació del disseny arquitectònic client-servidor definit en el punt 3.3.5. Aquest enfocament permet que els clients interactuïn amb el servidor a través de sol·licituds específiques, mentre que la base de dades s'encarrega de gestionar i emmagatzemar la informació de manera eficient. Al llarg d'aquesta secció, es detallaran els components clau de la implementació, incloent-hi la configuració del servidor, la integració de la base de dades i el desenvolupament de les aplicacions client, així com les tecnologies i eines utilitzades per a garantir un rendiment òptim i una experiència d'usuari fluida.

3.4.1. Servidor

Per la creació del servidor s'ha utilitzat Node.js [27], junt amb el *framework* Express.js[28] per facilitar la creació del servidor web. Aquest proporciona el fitxer

[27] https://nodejs.org/en
[28] https://expressjs.com/

HTML del joc i també serveix com a intermediari entre la comunicació del joc i la base de dades.

Per a gestionar la instal·lació i manteniment d'aquest framework i altres mòduls necessaris s'ha utilitzat el gestor npm[29]. Aquest gestor et permet accedir a un gran repositori de paquets, facilitant la incorporació de funcionalitats als projectes sense necessitat de desenvolupar tot des de zero. A més, habilita la gestió de dependències, l'execució de scripts i la configuració de projectes a través d'un arxiu anomenat *package.json*.

3.4.1.1. Codi del servidor

Fent servir el *framework* d'Express.js, s'ha aconseguit gestionar les sol·licituds, respostes sobre el mateix i el *middleware* necessari perquè funcioni tot correctament. Pel que fa a la comunicació del client amb la base de dades s'ha implementat com si fos una API.

3.4.1.2. Variables d'entorn

Les variables d'entorn són parells clau-valor que s'utilitzen per a emmagatzemar informació de configuració en un sistema operatiu o en un entorn d'execució d'aplicacions. Aquestes variables permeten als programes accedir a dades com a rutes d'arxius, configuracions de xarxa, credencials d'accés i altres paràmetres que poden variar entre diferents entorns (per exemple, desenvolupament, proves i producció). D'aquesta manera, es facilita la gestió de configuracions sense necessitat de modificar el codi font.

Per aconseguir aquesta funcionalitat, s'ha importat el mòdul *dotenv*[30] a Node.js perquè agafi aquestes variables des del fitxer *.env*. Aquest arxiu conté els parells clau-valor que es necessiten en el projecte. En el nostre cas només s'ha necessitat incloure la informació de la connexió amb la base de dades.

Un cop tenim l'arxiu simplement necessitem carregar-lo dins l'aplicació Node.js, per tant, s'importa el fitxer a la primera línia de l'arxiu del servidor.

3.4.1.3. Accés a la base de dades

Com s'ha explicat abans, el servidor s'encarrega de la comunicació entre el joc i la base de dades. Com mostra el Codi 3.1, s'han afegit diverses rutes a les quals el client pot accedir mitjançant AJAX[31].

Aquestes rutes executen les funcions que conté l'arxiu *DBFunctions.js* que conté la connexió a la base de dades i les diferents funcions que executen les consultes o modificacions pertinents amb el codi SQL.

[29] https://www.npmjs.com/
[30] https://www.npmjs.com/package/dotenv
[31] https://en.wikipedia.org/wiki/Ajax_(programming)

```
const DBFunctions = require('./DBFunctions.js');

app.post('/getUsuari', async (req, res) => {
  DBFunctions.GetUser(req, res);
});

app.post('/preguntesUsuari', function (req, res) {
  DBFunctions.InsertPreguntes(req, res);
});

app.post('/updatePuntuacio', function (req, res) {
  DBFunctions.UpdatePuntuacio(req, res);
});

app.get('/getTopPuntuacions', function (req, res) {
  DBFunctions.GetTopPuntuacions(req, res);
});
```

Codi 3.1 Rutes AJAX

Per la connexió amb la base de dades s'ha utilitzat el mòdul de MariaDB[32], el qual et permet crear connexions i fer consultes a bases de dades MySQL o MariaDB.

Gràcies a aquestes funcions i la utilització d'AJAX, des del client podem crear l'usuari i saber si fer-li les preguntes o iniciar sessió directament i actualitzar les seves dades sobre la puntuació.

3.4.2. Base de dades

Com ja s'ha explicat, les dades que hem de guardar són molt poques i es pot utilitzar només una taula per emmagatzemar tot el que necessitem.

```
CREATE TABLE `puntuacions` (
  `nom` varchar(25) NOT NULL,
  `edat` tinyint(4) NOT NULL,
  `experiencia` tinyint(1) DEFAULT NULL,
  `sexe` char(1) DEFAULT NULL,
  `dispars` mediumint(11) DEFAULT NULL,
  `encerts` mediumint(11) DEFAULT NULL,
  `puntuacio` int(11) DEFAULT NULL,
  `ronda` smallint(11) DEFAULT NULL
);

ALTER TABLE `puntuacions` ADD PRIMARY KEY (`nom`,`edat`);
```

Codi 3.2 Taula SQL

Com es pot veure en el Codi 3.2, a part de crear la taula, també es genera el parell nom i edat com a claus primàries per identificar els diferents usuaris.

[32] https://www.npmjs.com/package/mariadb

3.4.3. Client

La implementació del joc es fa en el costat del client, per tant, aquest apartat abasta la major part de la implementació del projecte considerant la part feta en A-Frame i JavaScript.

3.4.3.1. A-Frame

El *framework* d'A-Frame es basa en HTML i JavaScript, per tant, s'ha hagut de crear un document HTML afegint el *framework* dins l'element *<head>* de l'arxiu. Un cop fet es pot crear una escena d'A-Frame, aquesta escena és la que conté els components a mostrar del joc. Per crear i modificar la lògica dels components es fa a través de JavaScript.

Un component és una unitat modular que encapsula una funcionalitat específica i es pot reutilitzar en diferents entitats dins d'una escena, per exemple, el component fletxa es reutilitza per totes les fletxes generades. Existeixen components ja creats per A-Frame que ofereixen certes funcionalitats, però també et permet crear els teus propis, d'aquesta manera, els desenvolupadors poden agregar característiques i comportaments als elements 3D de manera senzilla i organitzada.

3.4.3.2. Components

Com s'acaba d'explicar en el punt anterior, els components necessaris i la modificació dels mateixos s'ha fet a través de JavaScript.

```
AFRAME.registerComponent('nomComponent', {
  schema: {
    dada1: {type: 'number', default: 0.5},
    [...]
  },
  init: function () {
    [...]
  },
  update: function () {
    [...]
  },
  tick: function () {
    [...]
  },
  remove: function () {
    [...]
  },
  funcioPersonalitzada: function () {
    [...]
  }
});
```

Codi 3.3 Component d'A-Frame

Al Codi 3.3 es pot veure l'estructura bàsica d'un component, que pot contenir els següents apartats:

- schema: És un objecte JSON[33] que defineix les propietats del component com per exemple el color o la grandària.
- init: És la funció que s'executa només una vegada quan es crea el component a l'escena.
- update: És la funció que s'executa en qualsevol moment que les propietats del component canvien.
- tick: És la funció que s'executa a cada fotograma que es genera.
- tock: Idèntic a la funció tic però s'executa un cop ja s'ha renderitzat l'escena.
- remove: És la funció que s'executa quan s'elimina el component.
- pause: És la funció que s'executa quan l'entitat o l'escena es pausa.
- play: És la funció que s'executa quan l'entitat o l'escena es reprèn.
- Funcions personalitzades: Es poden afegir diferents funcions al component, encara que no s'executen automàticament per A-Frame, s'han de cridar explícitament.

Per cobrir les funcionalitats del joc s'han creat els següents components, els quals s'explicaran a continuació:

- Arc
- Corda
- Fletxa
- Planta
- Vagoneta
- Terra
- Diana

3.4.3.2.1. Arc

Aquest component s'encarrega d'assignar el model 3D de l'arc a l'entitat, el qual es pot veure a la Figura 3.6 i generar la *pool*[34] de fletxes. Una *pool* en A-Frame és un conjunt de components de mida limitada els quals es poden anar agafant i retornant de manera dinàmica, utilitzant sempre les mateixes entitats ja generades. D'aquesta manera evitem haver de destruir l'entitat i generar-ne una de nova. En aquest cas, actua com un carcaix virtual, en el qual les fletxes ja disparades es retornen al carcaix i es reutilitzen.

Quan s'agafa l'arc, es canvia el model de la mà que l'ha agafat pel model de l'arc, també s'afegeix el component de la corda i internament s'assigna l'altra mà com la "mà de la corda", que serà la que podrà agafar les fletxes.

Un cop fet això demana una fletxa de la *pool* i genera el menú principal.

[33] https://en.wikipedia.org/wiki/JSON
[34] https://aframe.io/docs/1.7.0/components/pool.html#main

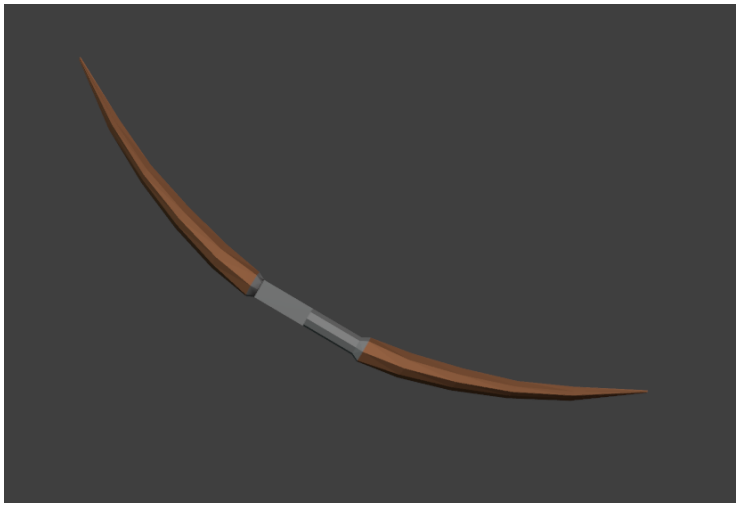

Figura 3.6 Model 3D de l'arc

Per igualar la rotació de la fletxa amb l'arc s'han fet ús dels quaternions[35]. Un quaternió és un nombre complex estès a quatre dimensions, que s'utilitza en matemàtiques i física per a representar rotacions en l'espai tridimensional. Els quaternions són especialment útils en gràfics per ordinador, robòtica i navegació, ja que permeten realitzar interpolacions suaus i evitar problemes com la *gimbal lock*[36] que poden ocórrer amb altres mètodes de representació de rotacions, com els angles d'Euler. Es pot veure un exemple de *gimbal lock* en el moviment del braç de la Figura 3.7, tret d'un vídeo de YouTube en el que ho explica [2], a causa de la utilització d'angles d'Euler (en la primera fila) en vers de quaternions (la segona fila).

Figura 3.7 Exemple de *gimbal lock*

[35] https://ca.wikipedia.org/wiki/Quaterni%C3%B3
[36] https://en.wikipedia.org/wiki/Gimbal_lock

Amb l'arc es volia aconseguir una simulació més real estirant també els extrems de fusta, això es pot aconseguir mitjançant una armature[37] o esquelet, en el qual es defineixen els ossos de l'objecte. A partir d'aquests ossos es pot doblegar el model i es doblega en els punts a on estan les juntes entre els ossos. En un programari d'objectes 3D com pot esser Blender, aquesta aplicació és bastant trivial sobre objectes simples (com un arc) però quan es passa a Three.js, es pot accedir als ossos generats, però la deformació de l'objecte no és senzilla, s'ha de fer de manera manual, per aquest motiu, es va decidir centrar-se en la resta del joc i simular la corda en la manera en la que s'explica en el següent punt.

3.4.3.2.2. Corda

Aquest component simplement genera una corba utilitzant l'algorisme de Three.js de *CatmullRomCurve3*, al Codi 3.4 es pot veure la implementació de la funció junt amb la creació de l'element 3D de la corba que es mostrarà a l'escena, aquesta corba simularà l'extensió i retracció de la corda.

La corba de Catmull–Rom[38] utilitza l'algorisme de Catmull–Rom per generar una corba llisa segons els punts indicats. Al Codi 3.4 es pot veure en les primeres línies que genera la corba a partir de quatre punts, una àncora superior que seria la part de dalt de l'arc, el contrari per l'àncora inferior, i dos punts intermedis que estan molt junts simulant esser només un perquè la funció requereix un mínim de quatre punts. Aquests dos punts intermedis són als que se li aplica la distància entre les mans perquè només s'estiri la corba d'enmig.

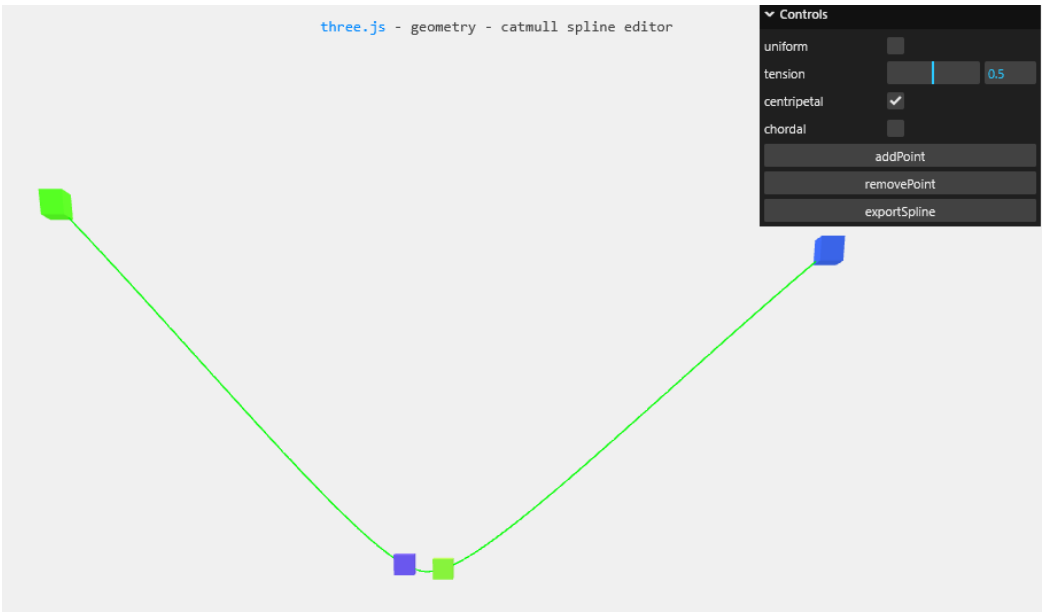

Figura 3.8 Corba Catmull-Rom de three.js

[37] https://en.wikipedia.org/wiki/Armature_(computer_animation)
[38] https://en.wikipedia.org/wiki/Centripetal_Catmull%E2%80%93Rom_spline

```
// Crear la corba a partir dels punts proporcionats i la
distància de les mans
let corda = new THREE.CatmullRomCurve3([
  new THREE.Vector3(0,
      cordaEntity.components.cordamath.data.ancoraSuperior, 0),
  new THREE.Vector3(0,
      cordaEntity.components.cordamath.data.puntIntermigSuperio
r, distanciaMans),
  new THREE.Vector3(0,
      cordaEntity.components.cordamath.data.puntIntermigInferio
r, distanciaMans),
  new THREE.Vector3(0,
      cordaEntity.components.cordamath.data.ancoraInferior, 0)
]);

let pointsCorda = corda.getPoints(50);
// Crear l'objecte 3D a partir dels punts de la corba
let geometry = new
      THREE.BufferGeometry().setFromPoints(pointsCorda);

// Assignar el color
let material = new THREE.LineBasicMaterial({
  color: cordaEntity.components.cordamath.data.color,
  linewidth: 1
});

// Generar la corba junt amb el color
return new THREE.Line(geometry, material);
```

Codi 3.4 Funció CatmullRomCurve3

S'ha fet servir aquest algorisme, perquè la corba que genera és molt similar a l'extensió d'una corda, com es pot veure en la Figura 3.8. A més a més, també dibuixa una línia recta si es posen tots els punts d'un eix a la mateixa altura, per tant ens permet simular tots els possibles estats de la corda.

3.4.3.2.3. Fletxa

El component de fletxa és el que més càlculs conté, defineix la fletxa, el seu moviment i les col·lisions.

Primerament, defineix el model 3D de la fletxa, el qual es pot veure en la Figura 3.9, i li assigna el component *aabb-collider*[39] el qual aprovisiona de detecció de col·lisions.

Un detector de col·lisions AABB (*Axis-Aligned Bounding Box*) és un mètode utilitzat en gràfics per ordinador i simulacions físiques per a determinar si dos objectes en un espai 2D o 3D incideixen. Aquest enfocament es basa a embolicar cada objecte en un rectangle (en 2D) o un paral·lelepípede (en 3D) alineat amb els eixos de coordenades, la qual cosa simplifica el càlcul de col·lisions. En verificar si les AABBs de dos objectes se superposen, es pot determinar ràpidament si hi ha una possible

[39] https://en.wikipedia.org/wiki/Minimum_bounding_box

col·lisió entre ells, la qual cosa permet optimitzar el rendiment en aplicacions on es manegen múltiples objectes, com a videojocs o simulacions físiques.

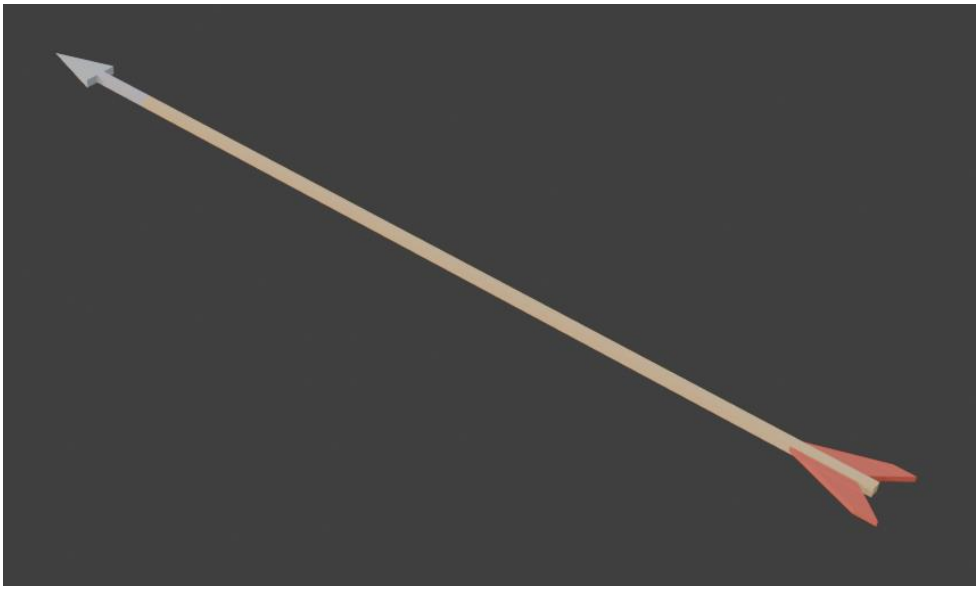

Figura 3.9 Model 3D de la fletxa

També disposa de la funció *tick*, la qual actualitza el seu estat en cada fotograma. Aquesta funció s'encarrega de copiar la posició i rotació de la mà de la corda fins que s'apropa prou a l'arc i es pressiona el botó disparador. Quan ocorr això, en vers de copiar la posició i rotació de la mà de la corda, passa a copiar-ho de la mà de l'arc. D'aquesta manera se simula que s'ha posat la fletxa a l'arc.

Un cop la fletxa està acoblada a l'arc, segons la distància entre la mà de la corda i la mà de l'arc, es mourà la fletxa i la corda de forma proporcional fins a un límit. Amb això aconseguim modular la tensió de la corda.

A partir d'aquí, un cop s'amolla el botó disparador, s'agafa el mínim entre la distància entre les mans i el límit de la corda, establert anteriorment, per calcular la velocitat inicial de la fletxa seguint la fórmula que s'explica a l'article [3] i detallada a l'Equació 1.

$$v_0 = \sqrt{\frac{eFx}{m + kM}}$$

Equació 1 Velocitat inicial de la fletxa

A on:

- e = Eficiència de l'arc, pels arcs medievals és de *0.9*.
- F = Força de l'arc, pels arcs medievals varia entre 200 i 400 N.
- x = Distància entre la posició en repòs i la tensió actual, en el nostre cas la distància entre les mans.

- m = La massa de la fletxa.
- k = Factor el qual representa la suma de les energies cinètiques de les parts mòbils de l'arc. Pels arcs medievals varia entre 0.03 i 0.07.
- M = La massa de l'arc (aproximadament 1 kg).

En el cas del joc, totes aquestes variables són constants durant tota la durada de l'execució, excepte la x (la distància entre les mans). Per tant, es pot separar l'equació en dues parts com es veu a l'Equació 2.

$$v_0 = \sqrt{\frac{eF}{m + kM}} \cdot \sqrt{x}$$

Equació 2 Velocitat inicial de la fletxa simplificada a *x*

D'aquesta manera el que aconseguim és tenir una constant multiplicada per la variable de la distància per estalviar recursos en temps d'execució. Tot això quedaria com es veu en el Codi 3.5.

```
// Forca (N) de tensió de l'arc
const FORCAARC = 330;
const EFICIENCIAARC = 0.9;
const MASSAARC = 0.9; // Massa de l'arc en kg
const MASSAFLETXA = 0.044; // Massa de la fletxa en kg
const FACTORKE = 0.05; // La suma de l'energia cinètica (KE) de
les parts mòbils de l'arc.
// Factor de tensió per multiplicar a la distancia de la corda
i obtenir la força
const TENSIO = Math.sqrt((EFICIENCIAARC * FORCAARC) /
(MASSAFLETXA + FACTORKE * MASSAARC));


function calcularTensio() {
  // Calcular la distancia entre les mans
  const distancia =
Math.min((maArc.object3D.position.distanceTo(maCorda.object3D.p
osition) / 4), CORDAESTIRARMAX);

  // Calcular la velocitat inicial
  return TENSIO * Math.sqrt(distancia);
}
```

Codi 3.5 Càlcul de la velocitat inicial de la fletxa

Com a constants, per tenir una representació el més real possible, s'han definit les següents a partir de dades reals i ajustant alguns valors, per obtenir una sensació de dispar més agradable. A l'apèndix A.1 es poden veure les fonts d'on s'ha obtingut la informació.

- $e = 0.9$
- $F = 330\ N$
- $m = 0.044\ Kg$

- $k = 0.05$
- $M = 0.9\ Kg$

Un cop ja sabem totes aquestes dades només ens falta conèixer la distancia entre les mans (el mínim entre la distància entre les mans i el màxim que es pot "estirar" la corda) i multiplicar-ho per la constant de tensió, d'això s'encarrega la funció *calcularTensio()*. La distància entre les mans es divideix entre 4 perquè s'ha de tenir en compte que la corda està a certa distància de la mà que sosté l'arc. Amb el moviment de la corda es pot visualitzar en temps real el valor que retorna la funció de Three.js *distanceTo()*, i simplement restant la distància entre la fletxa i l'arc no és suficient per obtenir un valor correcte, fent prova i error es va trobar que dividir el valor entre 4 era la millor opció.

Un cop coneguda la velocitat inicial de la fletxa, ara ha de seguir la trajectòria d'un projectil típic amb la força de la gravetat actuant sobre ell. Per aconseguir això s'ha utilitzat la fórmula obtinguda del vídeo [4] i detallada a l'Equació 3.

$$x = v_0 t \cos\theta$$

$$y = v_0 t \sin\theta - \frac{1}{2} g t^2$$

Equació 3 Trajectòria d'un projectil

A on:

- v_0 = És la velocitat inicial obtinguda anteriorment.
- t = És el temps transcorregut.
- θ = És l'angle en el que apunta la fletxa.
- g = És la força de la gravetat.

Igual que s'ha fet abans per reduir el nombre d'operacions en temps d'execució, s'ha dividit la fórmula en les següents passes.

Com a especificació del component es té que:

1. $g = GRAVETAT\ \frac{1}{2}$, a on *GRAVETAT* és una constant amb el valor *9,81*.

A l'hora de disparar es calculen les velocitats inicials sobre x i y, a partir de l'angle obtingut del quaternió de la fletxa, sense tenir en compte el temps.

2. $v_x = v_0 \cos\theta$
3. $v_y = v_0 \sin\theta$

Com el que varia és el temps transcorregut, al bucle de renderització (la funció tick() del component) es calcula la resta d'operacions que depenen d'ell.

4. $x = v_x t$
5. $y = v_y t - g t^2$

Gràcies a aquesta optimització, el bucle *tick()*, que s'executa en cada fotograma, realitzarà un menor nombre d'operacions. Cosa que es tradueix en una major

eficiència en el rendiment del sistema, es millora la fluïdesa i la resposta general de l'aplicació, la qual permet una experiència més fluida per al jugador.

```
element.addEventListener('grab-end', async (event) => {
  [...]
    // Calcular la força de dispar
    data.forca = calcularTensio();
    // Agafa la rotació de l'objecte en radians comparant-la
amb la rotació neutra 0
    let rotacio = this.el.object3D.quaternion.angleTo(
      new THREE.Quaternion(0, 0, 0, 0)
    );
    data.velX = data.forca * Math.cos(rotacio);
    data.velY = data.forca * Math.sin(rotacio);
    // Dispara la fletxa amollada
    data.disparada = true;

[...]

});

element.addEventListener('hitstart', (event) => {
  numEncerts++;
  if (this.data.disparada) {
    escena.components.pool__fletxa.returnEntity(this.el);
    this.data.disparada = false;
  }
});
tick: function (time, timeDelta) {
  let data = this.data;
  let el = this.el;
  if (data.disparada) {
    // Aplica la fórmula de la trajectòria de projectils per
moure la fletxa cap endavant i cap abaix
    let distX = data.velX * data.temps;
    let distY = data.velY * data.temps - (data.gravetat *
data.temps * data.temps);
    el.object3D.translateZ(-(distX - data.distAnteriorX));
    el.object3D.translateY(distY - data.distAnteriorY);
    data.temps += 0.01;
    data.distAnteriorX = distX;
    data.distAnteriorY = distY;
    // Si l'altura de la fletxa és menor a 0 o el temps es
major a 8 retorna la fletxa a la pool
    if (this.el.object3D.position.y < 0 || data.temps > 8)
escena.components.pool__fletxa.returnEntity(this.el);
  }
  [...]
}
```

Codi 3.6 Trajectòria de la fletxa

En el Codi 3.6 es pot veure com s'ha dut a terme la implementació de la fórmula en el moment en què s'amolla la fletxa. Primer es calcula la tensió o velocitat inicial de la fletxa, que es guarda com la força actual. Seguidament, l'angle de tir s'obté a partir del quaternió de la fletxa com ja s'ha explicat anteriorment, el qual ens permet calcular la distància tant en x com en y. A partir d'això, amb la funció *tick()*, a cada fotograma es calcula la distància segons el temps que ha passat. Finalment, quan la fletxa supera el temps establert o té una posició en y (altura) negativa, es retorna a la *pool* per poder tornar a utilitzar-la més endavant sense haver de

generar i renderitzar una fletxa nova cada vegada. A part de retornar-se també en el moment de col·lisionar amb alguna entitat.

A partir de la fórmula de velocitat inicial segons la tensió exercida i la fórmula de la trajectòria d'un projectil, podem obtenir una representació realista del tir d'una fletxa amb un arc en la Terra. Aquestes fórmules ens permeten calcular amb precisió la velocitat i l'angle de llançament, i al mateix temps determinar la trajectòria que seguirà la fletxa. Gràcies a això, podem simular de manera efectiva com es comporta la fletxa en un entorn real, proporcionant així una major immersió.

3.4.3.2.4. Planta

La planta és l'enemic principal, aquest component s'encarrega d'assignar a l'entitat una rotació aleatòria i la mida especificada. Un cop fet, genera dos elements fills que contenen els models 3D, els quals s'han vist anteriorment en conjunt a la Figura 3.3, un és el de la planta amb les animacions de créixer i de morir i l'altre és el de la fruita, que li assigna un color diferent segons la vida que tingui , i només amb l'animació de morir, pel fet que el component de les col·lisions no s'actualitza a la posició real, sinó que assigna el col·lisionador al primer fotograma de l'animació. A la Figura 3.10 es poden veure els diferents colors segons la quantitat de vida de la planta.

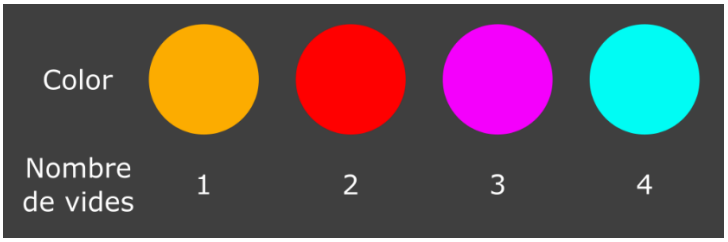

Figura 3.10 Color de la fruita segons la vida

La lògica programada del component és la següent:

- Quan es crea un nou element del tipus planta, aquest comença amb l'animació de la planta de créixer, quan finalitza l'animació, la fruita passa a ser visible.
- Quan el col·lisionador de la fruita detecta una col·lisió, i aquesta té menys de dos punts de vida, comença les animacions de morir, tant de la fruita com de la planta, junt amb el so d'encertar a la fruita. Si té més de dos punts de vida, es resta un punt i s'actualitza al color adient.
- Quan la vagoneta supera la distància assignada, s'elimina el component de l'escena i s'activa el so de perdre vida.

Una complicació que es va tenir amb la implementació dels enemics va ser que, primerament, es varen afegir els enemics com una pool, però quan es demanava una entitat ja utilitzada a l'escena anteriorment i se li assignava una nova posició, el component de detecció de les col·lisions AABB no s'actualitzava a la nova posició,

per tant per que detectés la col·lisió s'havia de disparar a la primera posició assignada a l'entitat, posició en la qual ja no hi havia res visualment a l'escena.

3.4.3.2.5. Vagoneta

Aquest component s'encarrega d'assignar el model 3D de la vagoneta, mostrat a la Figura 3.11, les animacions de moure i reiniciar i l'entitat que mostrarà la vida i els punts. A part d'això també controla quan es completa cada animació i realitza les funcions pertinents.

L'animació de moure mou la vagoneta els metres especificats a la constant *AVANCVAGONETA* cap endavant. Un cop finalitza, posa la variable *jugant* a fals i genera el modal del punt de control.

L'animació de reiniciar, mou la vagoneta al punt inicial $(x: 0, y: 0, z: 0)$ i quan acaba l'animació, elimina els camins generats anteriorment, genera el camí inicial, actualitza la vida i els punts i finalment genera el modal del menú.

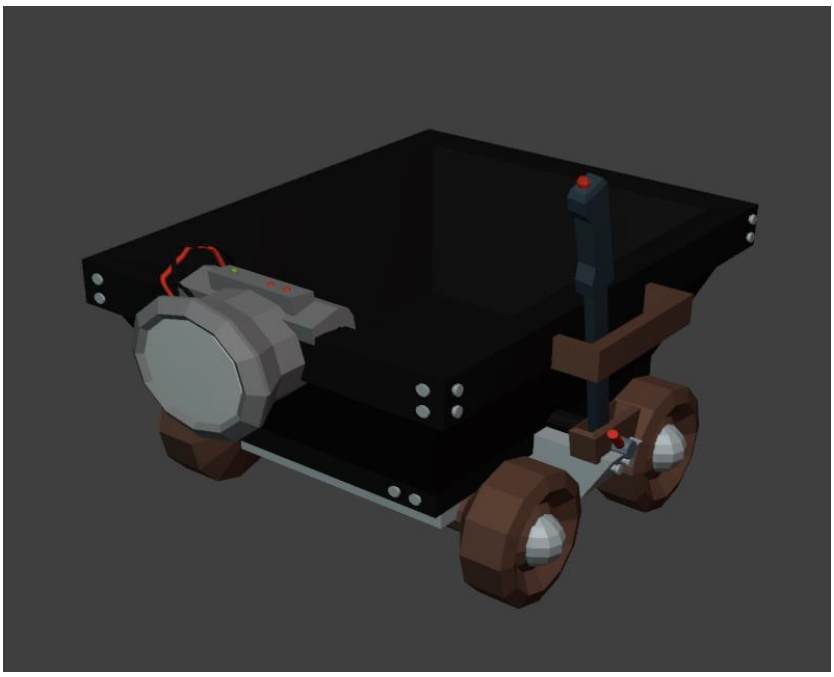

Figura 3.11 Model 3D de la vagoneta

Per limitar la mobilitat del jugador a moure's només dins la vagoneta, es fa ús d'un *navmesh*[40].

Un *navmesh*, o navegació de malla, és una representació en forma de malla d'un entorn 3D que s'utilitza en el desenvolupament de videojocs i simulacions per a facilitar la navegació de personatges o entitats dins d'aquest entorn. Aquesta malla defineix les àrees en les quals els personatges poden moure's, i això permet implementar intel·ligència artificial que permeti als personatges trobar rutes

[40] https://en.wikipedia.org/wiki/Navigation_mesh

òptimes entre punts i evitar obstacles. Els *navmesh* són especialment útils, ja que simplifiquen el procés de càlcul de rutes i milloren l'eficiència del sistema de navegació en comparació amb altres mètodes més tradicionals, com per exemple comprovar la posició del jugador constantment.

3.4.3.2.6. Terra

Aquest component simplement aplica el model 3D especificat a l'entitat i conté la funció especial de *comprovarDistancia()* que comprova la distància de l'entitat a la vagoneta, si és major a la distància màxima especificada s'elimina ell mateix de l'escena.

Com es pot veure a la Figura 3.12 i a la Figura 3.13 es tenen dos models 3D de camins. Un és el camí inicial amb el castell al darrere i l'altre és el camí que es va afegint a mesura que s'avança en el joc el qual conté només el camí amb els arbres als costats. A més a més, es pot comprovar que els arbres tenen certes parts del color lila de la planta alienígena, això és per fer veure que la planta està envaint i infectant la flora local.

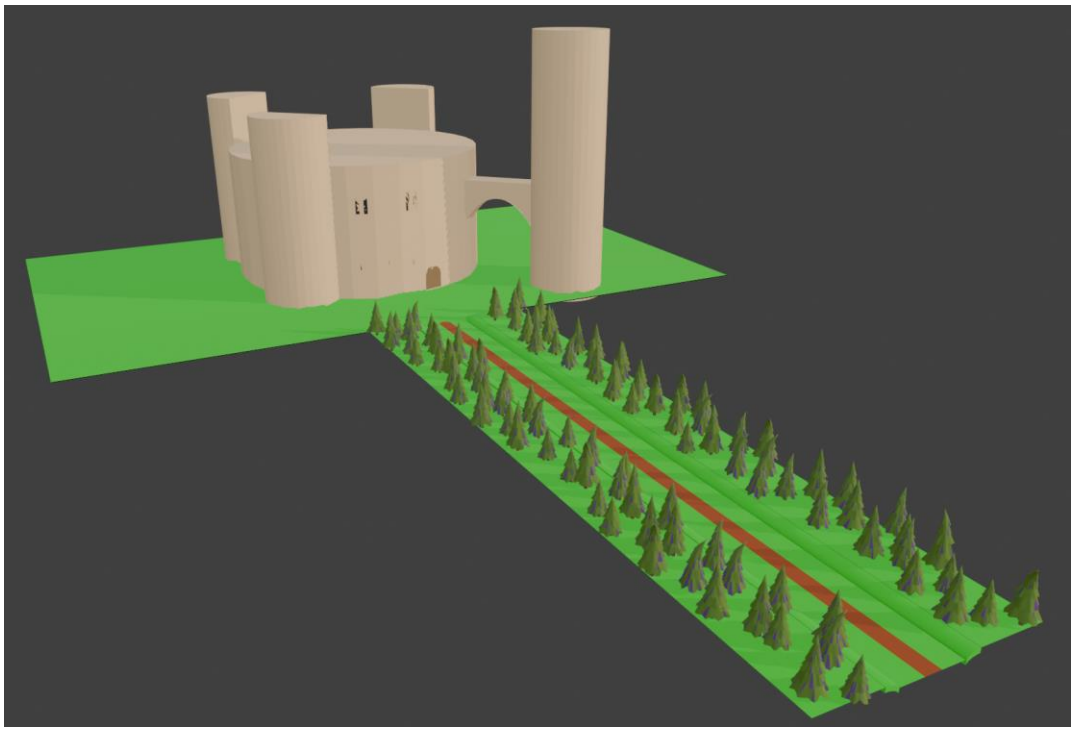

Figura 3.12 Model 3D del camí inicial

Durant la realització del projecte, es va comprovar que és més eficient tenir només un model 3D gran que tenir un tros petit el qual es va repetint constantment encara que s'utilitzés una *pool*. Per tant, es genera un model gran i es fa servir la funció *comprovarDistancia()* quan s'arriba al punt de control, per no estar comprovant constantment la distància. Com la distància màxima és la mida del

camí i la distància entre els punts de control és la meitat, només s'eliminarà l'entitat quan la vagoneta hagi passat al punt de control de la meitat del següent camí. Així mateix, només es genera un nou camí si el nombre de la ronda actual és senar, això provoca que com a màxim es tinguin a la vegada dues entitats de terra en escena. En resumits comptes, quan la vagoneta arriba al punt de control que s'ubica a la meitat d'un camí, s'elimina el camí anterior i es crea el següent.

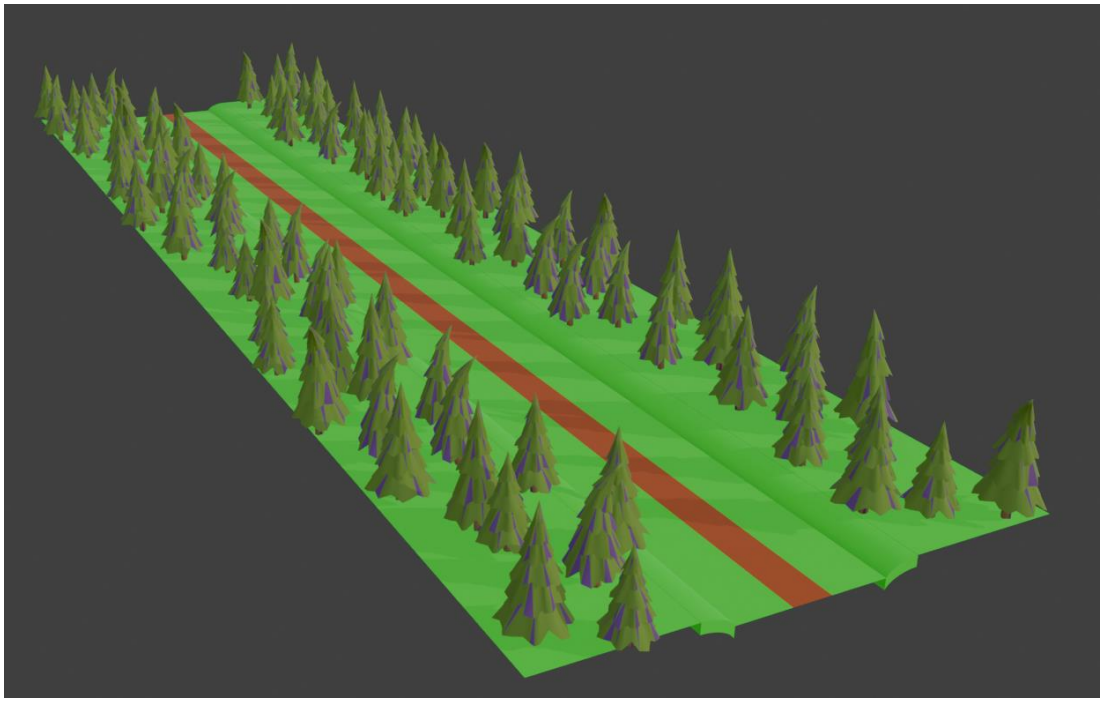

Figura 3.13 Model 3D dels camins subseqüents

3.4.3.2.7. Diana

Aquest element és molt senzill, només aplica el model 3D de la diana a l'entitat, que es pot veure a la Figura 3.14, i li afegeix el detector de col·lisions per eliminar-la de l'escena quan es fer amb la fletxa, a més d'executar el so corresponent.

Figura 3.14 Model 3D de la diana

3.4.3.3. Funcions auxiliars

Ara es detallaran les funcions auxiliars que s'han hagut d'implementar per la lògica del joc i no es podien fer o no provocaria cap avantatja fer-ho a través dels components d'A-Frame.

3.4.3.3.1. controladorEnemics(tipus)

És la funció principal del funcionament dels enemics. És un orquestrador que va generant els enemics especificats fins a un màxim d'enemics simultanis en pantalla.

És un bucle que mentre la variable jugant estigui a *true*, efectua les següents operacions.

Primer comprova que el nombre d'enemics en pantalla actuals sigui igual o menor que el màxim establert. Si és major, se pausa el temps especificat d'espera entre la generació d'enemics i es torna a l'inici del bucle.

Si és menor, es crea una entitat de l'enemic especificat per paràmetre, li assigna una posició aleatòria segons si és planta o diana. Si és de tipus planta, també li assigna una mida aleatòria entre dos valors.

Finalment, l'hi aplica la classe del col·lisionador per detectar les col·lisions amb les fletxes, afegeix l'entitat a l'escena i augmenta el nombre d'enemics en pantalla actuals en 1.

Un cop fet tot això s'atura el temps especificat d'espera entre la generació d'enemics i torna a començar.

El bucle s'acaba quan la vagoneta arriba al punt de control o el jugador es queda sense vides, ja que ambdues funcions posen la variable jugant a *false*.

3.4.3.3.2. generadorModals(tipus)

Figura 3.15 Modal del menú

Els modals agafen el nom dels modals d'HTML, pel fet que, en essència, són molt similars a un quadre de diàleg, perquè no pots interactuar amb l'entorn sinó només amb el quadre. Tots els tipus de modals son visualment molt similars i desapareixen (o en generen un de nou) quan selecciones una de les opcions.

A la Figura 3.15 es pot veure el modal del menú com exemple. Existeixen 6 tipus de modals, i segons el tipus escriu el text adient a cada un.

1. **Menú**

És el menú principal del joc el qual et dona la benvinguda i et deixa seleccionar entre començar la partida o anar al mode de pràctica.

Si decideixes començar la partida, reinicialitza les variables del joc al valor per defecte i comença la ronda. En canvi, si decideixes anar al mode de pràctica, canvia les variables del joc per unes més adequades per la pràctica i activa el controlador d'enemics amb el tipus d'enemic diana.

2. **Continuar**

S'encarrega de generar el camí de davant si la ronda és senar, com s'ha explicat abans, i actualitzar la puntuació a la base de dades si aquesta és major a l'anterior, com a manera de "protecció" en cas que ocorri qualque imprevist que impedeixi enviar la puntuació un cop s'ha quedat sense vides, com per exemple, que s'esgoti la bateria de les ulleres. Quan es dispara al botó de continuar, augmenta el nombre de la ronda al mateix temps que modifica les variables del joc perquè sigui de cada vegada un poc més difícil. A més a més, també té un botó per reiniciar el joc.

3. **Reiniciar**

Activa el so de partida acabada, envia la puntuació a la base de dades, en cas de ser major a l'anterior, i es crea el botó de reiniciar el qual elimina tots els enemics en pantalla, tots els camins generats, reinicialitza les variables necessàries de la lògica del joc a més de les que s'utilitzen per a l'estadística com el nombre de tirs i finalment mou la vagoneta a l'inici del joc i genera el camí inicial junt amb el modal del menú.

4. **Sortir pràctica**

Aquest modal es genera a un costat quan s'està en el mode de pràctica i serveix per sortir d'aquest, és a dir, atura el generador d'enemics, elimina els enemics (dianes) en pantalla i genera el modal del menú.

5. **Experiència**

Només surt quan és un usuari nou i demana si té experiència prèvia en RV. Quan es dispara a algun dels botons genera el modal sexe.

6. **Sexe**

Aquest modal també surt només quan és un usuari nou i demana el sexe de la persona.

Quan es dispara a algun dels botons ja no hi ha més preguntes i genera el modal del menú per poder començar a jugar.

Un problema d'A-Frame és que no utilitza UTF-8, per tant, no suporta els accents ni altres caràcters com la "ç".

3.4.3.3.3. actualitzarVidaPunts()

Aquesta funció és molt senzilla perquè simplement modifica el text que conté la vida i la puntuació cada vegada que hi ha qualque canvi. Però també important, pel fet que comprova si la vida és menor a 1, si ho és executa la funció *partidaAcabada()* que s'encarrega d'interrompre el joc i acabar la partida.

3.4.3.3.4. partidaAcabada()

Quan el jugador es queda sense vides, aquesta funció s'encarrega d'apagar el controlador d'enemics, aturar el moviment de la vagoneta i generar el modal de reiniciar el joc.

3.4.3.3.5. Funcions AJAX

Són les funcions específiques que s'encarreguen de comunicar-se a través d'AJAX amb la base de dades. Entre elles tenim:

1. **enviarPuntuacions()**

Actualitza la puntuació obtinguda actual, junt amb el nombre de tirs, nombre de tirs encertats i ronda actual dins la base de dades.

a. **Generació de la petició:** La direcció és */updatePuntuacio*. Al cos de la petició s'ha d'incloure un JSON amb la informació del Codi 3.7. El nom (com a *string*) i l'edat (com a *int*) són les claus primàries per saber quin registre s'ha de modificar, seguidament, es requereixen les dades de dispars, encerts, puntuació i ronda com a *integers*.

```
{
  nom: "nom",
  edat: edat,
  dispars: numDispars,
  encerts: numEncerts,
  puntuacio: punts,
  ronda: ronda
}
```

Codi 3.7 Cos JSON d'una petició d'enviarPuntuacions()

b. **Resposta de la petició:** Es retorna el codi HTTP *200 – OK* o si hi ha hagut un error, el codi *500 – Internal Server Error*.

2. **enviarPreguntes()**

Envia les respostes de les preguntes d'usuari nou, és a dir, la pregunta sobre experiència prèvia i la del sexe.

a. **Generació de la petició:** La direcció és */preguntesUsuari*. Al cos de la petició s'ha d'incloure un JSON amb la informació del Codi 3.8. A on el nom(com a *string*) i l'edat (com a *int*) són les claus primàries per saber quin registre s'ha de modificar, i les respostes d'experiència com una *string* amb els possibles valors de *true* o *false* i sexe com a *string* amb un únic caràcter.

```
{
  nom: "nom",
  edat: edat,
  experiencia: "experiencia",
  sexe: "sexe"
}
```

Codi 3.8 Cos JSON d'una petició d'enviarPreguntes()

b. **Resposta de la petició:** Es retorna el codi HTTP *200 – OK* o si hi ha hagut un error, el codi *500 – Internal Server Error*.

3. **crearTaulaPuntuacions()**

Fa la petició de les millors puntuacions a la base de dades i genera la taula de puntuacions a partir de les files rebudes.

a. **Generació de la petició:** És una petició a la direcció */getTopPuntuacions* del tipus *GET*, per tant no es posa res al cos.

b. **Resposta de la petició:** Es retorna el codi HTTP *200 – OK* junt amb un JSON amb els valors de nom, puntuació i ronda, com es pot veure en el Codi 3.9, de les 10 primeres files de la taula ordenada per puntuacions (de major a menor) o si hi ha hagut un error, el codi *500 – Internal Server Error*.

```
{data: [{
  nom: "nom",
  puntuacio: 0,
  ronda: 0
},{
  nom: "nom",
  puntuacio: 0,
  ronda: 0
},
...
]}
```

Codi 3.9 Resposta JSON de les 10 millors puntuacions

3.4.3.4. Controls

Per habilitar la interacció amb l'entorn virtual, s'han afegit diferents components a l'escena d'A-Frame que són principalment per la càmera (la perspectiva del jugador) i les mans.

A l'escena s'han especificat els controls, a partir dels components proporcionats pel mateix A-Frame, dins l'entitat de la vagoneta el que provoca que la posició de la càmera i les mans sigui relativa a la posició de la vagoneta. Junt amb això, s'ha afegit un *navmesh* que limita el moviment del jugador a dins la vagoneta.

Al Codi 3.10,es pot veure que els components utilitzats són:

1. ***camera***

Especifica la perspectiva des de la qual l'usuari veurà l'escena, a més de la posició i rotació inicials. Aquesta posició i rotació se sobreescriurà a partir de la informació de les ulleres connectades.

2. ***movement-controls***

Permeten moure la càmera i especificar el *navmesh* per restringir el moviment.

3. ***look-controls***

Provoca que l'entitat pugui girar igualant la rotació les ulleres.

4. ***hand-controls***

Proporciona compatibilitat amb els comandaments de RV i dibuixa les mans rastrejades amb gestos animats.

5. ***laser-controls***

Habilita que un controlador de RV tingui un làser o un cursor de raigs que s'utilitza per a l'entrada i les interaccions.

6. ***raycaster***

Proporciona un mètode d'intersecció basat en línies amb el *raycaster* de *Three.js*. Aquest mètode estén una línia des d'un origen cap a una direcció, i comprova si aquesta línia s'interseca amb altres entitats d'una llista.

```
<!-- Entitat del jugador (posició, rotació...) -->
<a-entity id="controls" movement-controls="constrainToNavMesh:
true; speed: 0.7" rotation="0 180 0" position="0 0.7 -0.5"
keyboard-controls>

  <!-Càmera o ulleres RV -->
      <a-entity id="camera" fixcamera="" look-controls="
      enabled: false" camera position="0 1.6 0"
      rotation=""></a-entity>

  <!-- Mans -->
      <a-entity id="maEsquerra" hand-controls="hand: left"
      mixin="mans"></a-entity>
      <a-entity id="maDreta" hand-controls="hand: right" laser-
      controls="hand: right" raycaster="objects: .collidable"
      cursor="rayOrigin: controller" mixin="mans"></a-entity>
</a-entity>
```

Codi 3.10 Controls

Com a component extra s'usa *super-hands*[41], el qual afegeix una millor interacció amb els controladors. Els gestos que implementa, són *hover, grab, stretch* i *drag-drop*. En el projecte s'ha usat principalment *grab* el qual facilita la funcionalitat d'agafar objectes i deixar-los anar, en el nostre cas l'arc i les fletxes.

3.4.3.5. Inici de sessió

Per habilitar l'inici de sessió s'ha necessitat un teclat virtual per poder introduir el nom i l'edat. Per això s'ha afegit el component *aframe-keyboard*[42] el qual ha facilitat la incorporació d'un teclat virtual.

Un cop es té l'entrada de dades s'ha creat la lògica que controla l'entrada de dades i comprova que el nom introduït és correcte. Després de comprovar-ho, ho envia a la base de dades i segons la resposta de la base de dades s'inicia sessió, o s'especifica que és un usuari nou, posant les dades pertinents al *sessionStorage*. El *sessionStorage* és molt similar al *localStorage,* el qual et permet guardar dades de manera quasi persistent, és a dir, no té data d'expiració i l'usuari ho pot esborrar manualment o s'esborra quan el navegador elimina les galetes, a més a més, les dades d'un document carregat en una "navegació privada" s'esborren quan es tanca el procés. En canvi, les dades guardades com *sessionStorage*, s'esborren sempre quan es tanca la finestra.

Quan s'ha iniciat sessió, gràcies a les dades del *sessionStorgage*, quan s'agafa l'arc sabem si s'ha de generar el modal de les preguntes d'usuari nou o es pot generar directament el modal del menú.

3.4.3.6. Sons

S'ha fet ús de diferents sons per augmentar el nivell d'immersió i entreteniment al joc, els quals es pot veure l'origen i la llicència a l'Apèndix B. Per aconseguir-ho, s'ha fet ús del mètode *play()* disponible a l'API Web d'HTMLMediaElement mitjançant JavaScript. Aquest mètode intenta iniciar la reproducció del contingut audiovisual especificat.

En un principi, es va intentar utilitzar el component d'àudio proporcionat per A-Frame, però quan s'especificava que no es reproduís de manera automàtica, provocava que l'hagués d'iniciar manualment l'usuari i si s'intentava posar en marxa a través de JavaScript després d'alguna acció com la d'una col·lisió entre dos elements, no et permetia la reproducció pel fet que no l'havia engegat expressament l'usuari. Això és així per evitar l'abús de contingut audiovisual no consentit per l'usuari que acaba essent irritant i proporciona una experiència degradable.

S'han implementat tant efectes de so, com música de fons. A la Taula 3.2 es poden veure els diferents sons i la descripció.

[41] https://github.com/c-frame/aframe-super-hands-component
[42] https://github.com/WandererOU/aframe-keyboard

Nom del so	Descripció
BotoPressionat.mp3	Dona una resposta auditiva al ferir un botó.
FletxaDisparada.mp3	Efecte del llançament d'una fletxa o similar
FruitaMorta.mp3	Dona una resposta auditiva quan es fereix a una fruita.
GameOver.mp3	Dona èmfasi al moment de quedar-se sense vides.
PerdreVida.mp3	Dona una resposta auditiva quan es perd una vida.
ColisioDiana.mp3	Dona una resposta auditiva quan la fletxa col·lisiona amb una diana.
MenuFons.mp3	Millora l'ambient del menú.
JocFons.mp3	Genera una experiència de joc més intensa i divertida.

Taula 3.2 Sons usats

En quant a les cançons de fons, MenuFons.mp3 i JocFons.mp3, s'han elegit melodies similars, però amb la diferència de que la del menú és un poc més "apagada" i la del Joc és més "alegre" i amb un ritme més ràpid. El fet de que siguin similars millora la consistència del joc.

3.5. Instal·lació

En aquest apartat, es detallaran les passes necessàries per a instal·lar el servidor web. El joc en si no necessita instal·lació per part de l'usuari, ja que s'executarà en el seu navegador a partir de les dades proporcionades per aquest servidor.

Es requereix tenir instal·lat el següent software, entre parèntesi la versió recomanada:

- Node.js (20.x)
- MariaDB (18.x)

A més a més de tenir descarregat el projecte.

Un cop es tenen instal·lats falta instal·lar les dependències de Node.js, com el framework d'express.js i la funcionalitat amb MariaDB. Per això, des d'una terminal s'ha d'executar la següent comanda des de dins la carpeta del joc.

Quan s'executi la comanda del Codi 3.11, aquesta mirarà a la carpeta actual un fitxer anomenat *package.json*, el qual conté tota la informació sobre el projecte necessària perquè es configuri npm i s'instal·lin les dependències associades.

```
# npm install
```

Codi 3.11 Instal·lació de les dependències de npm

Seguidament, s'ha d'insertar dins el MariaDB una base de dades anomenada "bellver" i dins aquesta crear una taula anomenada puntuacions amb els camps mencionats anteriorment en la Taula 3.1. També es pot optar per importar el fitxer "bellver.sql" (que es troba dins la carpeta del projecte) al MariaDB i ja es tindrà la base de dades configurada.

Ara només queda crear un fitxer anomenat ".env" i a dins afegir-hi els parells clau i valor d'usuari, contrasenya, *host* i nom de la base de dades correctes.

Finalment, per arrancar el servidor s'ha d'executar el Codi 3.12 des de la mateixa carpeta des d'on s'ha executat el Codi 3.11 anteriorment.

```
# npm run start
```

Codi 3.12 Executar el servidor

Capítol 4. Resultats

El principal resultat obtingut d'aquest projecte és una aplicació web, la qual conté un joc destinat a usuaris amb dispositius de RV, a la Figura 4.1 es pot veure una captura d'una partida iniciada. A més a més de comptar també amb una base de dades a on es guarda la puntuació màxima de cada usuari.

Figura 4.1 Resultat final

Aquest joc està disponible a través d'internet mitjançant la web towerdefense.ltim.uib.es. Tot el codi i documentació està disponible al *GitHub de AlexMorroUIB*.

A continuació es detallaran els aspectes proposats inicialment i el seu resultat.

Primerament, l'apartat més essencial era el d'aconseguir un joc de tir amb arc en RV i basat en web. Aquest objectiu s'ha aconseguit completament gràcies a la implementació del canvi en la força de l'arc segons s'estiri la corda i la trajectòria d'un projectil a la Terra. Aconseguir fer-ho en web, ha sigut possible gràcies als *framework* usats.

Un dels objectius principals era el de fer un joc amb el gènere de tower defense a la inversa, és a dir, recórrer un camí amb enemics als costats als quals s'haurà de disparar. Aquest objectiu s'ha assolit degut a que quan s'inicia la partida, el jugador està obligat a recórrer el camí i es generen enemics als costats els quals

ha de ferir amb les fletxes. Si no els fer, el jugador es va quedant sense vides i la puntuació no augmenta.

També es va proposar tenir una versió jugable (o versió *beta*) per a la fira Ciència per a tothom, d'aquesta manera els assistents podien experimentar les possibilitats de la tecnologia de RV i donar la seva retroalimentació, com es pot veure a la Figura 4.2. Encara que el joc no estigués del tot acabat, es va aconseguir tenir una versió a temps que era prou funcional i entretinguda per a poder presentar-se a la fira. En general, a la gent li va agradar el joc.

Figura 4.2 Presentació a Ciència per a tothom

Es va preparar una base de dades per poder iniciar sessió, registrar la puntuació màxima i altres dades que servien per treure una informació estadística. El problema és que els que provaren el joc a la fira no es molestaren en omplir les dades, a més a més, molts de pics s'usava el mateix usuari i només en el mode de pràctica, el qual no registra cap puntuació, ja que se suposa que s'està practicant. Per tant, es té la funcionalitat, però no es tenen prou dades per a treure qualque tipus d'estudi o estadística.

Poder presentar el joc a la fira, també va servir per descobrir errades de programació, les quals es varen arreglar posteriorment.

- Dos botons diferents per agafar la fletxa provoca una errada visual a on surt més d'una fletxa quan es pressionen els dos botons en segons quines combinacions. La solució va ser usar només un botó per a agafar la fletxa. Al mini joc Longbow de l'Steam Lab, el botó per agafar la fletxa és el

disparador, quan el vaig provar per primera vegada em va parèixer més intuïtiu i per aquesta raó només s'ha deixat aquest.

- Es permetia la rotació de la càmera amb la palanca del controlador, si la utilitzaves l'arc també rotava, però les fletxes no, per tant, només afectava visualment. Com a solució es va llevar completament el moviment amb controlador, ja que no és del tot necessari i pel que fa a la rotació, és més intuïtiu que roti el jugador físicament.
- Es podia agafar l'arc un cop haguessin iniciat sessió, però com la connexió no era molt bona a vegades trigava fins a uns 5 segons per poder agafar l'arc per mor d'esperar la resposta del servidor. Per solucionar-ho, un cop verificat que el nom és correcte, es va permetre agafar l'arc abans d'enviar-ho al servidor.
- La partida normal es feia un poc massa llarga i arribava a ser un poc avorrida. Per fer-ho més dinàmic es va augmentar lleugerament la velocitat de la vagoneta, el nombre d'enemics a la vegada en pantalla i es va reduir el temps de generació d'enemics.
- Les fletxes estaven constantment adherides a l'arc, el que feia que li llevés immersió. La solució adoptada va ser, fer que les fletxes estiguin a la mà contrària fins que es pressiona el botó disparador a prop de l'altra mà, en aquest moment passa a estar adherida a l'arc igual que anteriorment.
- A vegades, si es tiraven moltes fletxes seguides i quan es demanava una fletxa a la pool, aquesta estava buida i havies de reiniciar el joc complet. Per solucionar-ho, es va aplicar un temps de penalització per haver llançat moltes fletxes en poc temps.

A l'apartat 2.2.1 es varen proposar una sèrie de característiques que feien bé altres jocs similars i es podien implementar en el projecte. S'han aconseguit aplicar totes a excepció d'una:

- Poder usar diferents tipus de fletxes, com fletxes explosives o de foc.

Finalment, no s'ha pogut pel *framework* usat. A causa de la impossibilitat de crear una interacció a on es pogués canviar de fletxa, de manera ràpida i fluida. Es va pensar a utilitzar algun dels botons A, B, X o Y, però A-Frame no disposa d'aquesta interacció, i amb el temps disponible no s'ha pogut implementar des de zero.

A part d'aquesta, la característica d'"Elements distribuïts pel recorregut que donen punts extra." Es volia implementar com uns globus que s'enlairaven, però, a causa del problema ja esmenat anteriorment, que el col·lisionador no s'actualitza si l'element canvia de posició, no es va poder fer d'aquesta manera i es va decidir reutilitzar les dianes del mode de pràctica.

Capítol 5. Conclusions

Per concloure el projecte, es revisaran els objectius del mateix proposats a l'inici. De la mateixa manera, es compartirà la opinió personal sobre el treball realitzat, les possibles millores i l'estat de les tecnologies emprades.

5.1. Objectius

En primer lloc, la base del projecte era el de crear un joc de tir amb arc, en RV i basat en tecnologies web, perquè de cada vegada més aplicacions es fan pensant en ser usades des d'un entorn web i es volia conèixer de primera mà l'estat de l'art en aquest àmbit. Aquesta proposta s'ha complert amb èxit, ja que s'ha aconseguit realitzar el joc, és completament funcional en RV a través d'un navegador web i pot servir de base per a altres jocs de tir amb arc, o modificacions d'aquest. Altrament, la utilització de les fórmules de trajectòria del projectil i la velocitat inicial de la fletxa segons s'estiri la corda, han estat crucials per a la realització correcta del sistema de tir amb arc.

També es varen definir diferents característiques que s'havien d'implementar, vistes al Capítol 4, i la majoria s'han dut a terme de manera exitosa, encara que s'hagin trobat diferents complicacions que han alentit el desenvolupament o s'han hagut d'implementar de manera diferent, sobretot amb el problema del detector de col·lisions que no s'actualitza a la posició real de l'objecte, cosa que impedeix tenir animacions que modifiquin la mida o posició quan es genera l'objecte o la reutilització d'objectes si aquests canvien de posició.

Per millorar l'experiència de l'usuari, es pretenia tenir un bon rendiment, el qual ha resultat ser bo, però si s'inclouen més elements o un escenari més detallat, es nota un empitjorament en la fluïdesa. Més endavant, acabant la realització del projecte, es va descobrir WebAssembly[43] que compila llenguatges com C, C++ o Rust, entre d'altres, a codi binari portable, el qual s'executa des del navegador del client i ofereix un major rendiment en aplicacions d'ús de CPU o GPU intensiu. A partir d'aquesta informació i inspeccionant diferents jocs web, ja siguin en RV o simplement en 3D, es pot veure, que la immensa majoria de jocs en RV i la majoria de jocs 3D pel navegador usen aquesta tecnologia, cosa que permet fer jocs més complexos, pel fet que aprofiten millor els recursos hardware. Si hagués de fer un altre joc en RV per executar-se en web, segurament usaria WebAssembly.

[43] https://webassembly.org/

5.2. Possibles millores

Unes de les millores principals, podrien ser les complicacions que no s'han pogut dur a terme, com per exemple, poder tenir animacions inicials que no interfereixin amb el col·lisionador o usar un altre col·lisionador que s'actualitzi a la posició i mida reals de l'objecte.

Una millora d'immersió podria ser la deformació visual de l'arc segons s'estiri, com ja s'ha esmenat anteriorment en el punt 3.4.3.2.1, en vers de només deformar la corda. Així com millorar les textures del castell i del camí que són molt simples.

Un afegit extra podria ser el d'afegir girs al moviment de la vagoneta, acordes amb un escenari o camí que també els implementi, a més a més, es podrien incorporar canvis d'alçada. Tot això pot ajudar a donar més dinamisme i entreteniment al joc.

5.3. Opinió personal

Per acabar, en l'àmbit personal, ha estat un projecte amb el qual he après molt sobre l'estat de la tecnologia web actual, sobretot en un entorn de RV, i al mateix temps, m'he divertit mentre el realitzava.

Una cosa que em va ajudar bastant va ser dedicar un temps a usar Three.js sense A-Frame ni cap mena d'abstracció. Vaig aprendre molt llegint la documentació oficial de Three.js, el seu propi fòrum i material fet per la comunitat com vídeos a YouTube. Gràcies a aquesta inversió, vaig entendre millor com funciona A-Frame i em va llevar la por d'accedir a les propietats de Three.js des de l'abstracció que crea A-Frame, a més a més, em va ajudar a saber quines propietats havia de modificar o com ho havia de fer, sobretot pel fet de saber com havia de llegir la documentació oficial. Llevat d'això, també em vaig topar amb certes limitacions de Three.js, com per exemple, que no es pot teletransportar un objecte modificant directament el seu vector de posició, això em va permetre saber si certs problemes venien d'A-Frame o directament era Three.js que no ho permetia.

Referències

[1] J. Hale, «Top 5 WebXR Frameworks - Comparison,» [En línia]. Available: https://wonderlandengine.com/news/top-5-webxr-frameworks-comparison/.

[2] A. Pandemic, «YouTube,» [En línia]. Available: https://www.youtube.com/watch?v=aZ3Ut61RPto.

[3] A. Gonzalez, «The Physics Behind Archery,» [En línia]. Available: https://ffden-2.phys.uaf.edu/webproj/212_spring_2015/Addis_Gonzalez/Addis_Gonzalez/bow.html.

[4] J. Gibson, «Math and Science - YouTube,» [En línia]. Available: https://www.youtube.com/watch?v=z7UogSqABVc.

Apèndix A.

A.1. Fonts de l'especificació de l'arc

- Massa de la fletxa (estimació): https://exarc.net/issue-2019-3/mm/shooting-experiments-early-medieval-arrowheads
- Força de l'arc: https://en.wikipedia.org/wiki/English_longbow#Draw_weights
- Constants d'eficiència i força cinètica: https://ffden-2.phys.uaf.edu/webproj/212_spring_2015/Addis_Gonzalez/Addis_Gonzalez/bow.html

Apèndix B.

Apèndix de les obres creades per altres autors.

B.1. Arc

Llicència: CC BY 3.0

Autor: Zsky

Modificacions: Sí

Ref.: https://poly.pizza/m/XkJVO6cACA

B.2. Fletxa

Llicència: CC BY 3.0

Autor: Poly by Google

Modificacions: No

Ref.: https://poly.pizza/m/6pMBOmBFxGt

B.3. Planta

Llicència: CC0 1.0

Modificacions: No

Ref.: https://poly.pizza/m/0bVHWZZnNg

B.4. Vagoneta

Llicència: CC BY 3.0

Autor: Hunter Paramore

Modificacions: Sí

Ref.: https://poly.pizza/m/fjvk6xVJ3u3

B.5. Arbres

Llicència: CC0 1.0

Modificacions: Sí

Ref.: https://poly.pizza/m/oYtDty0fR6

B.6. Diana

Llicència: CC0 1.0

Modificacions: Sí

Ref.: https://poly.pizza/m/gKYbYR3z0M

B.7. Cel

Llicència: CC0 1.0

Modificacions: No

Ref.: https://polyhaven.com/a/kloofendal_48d_partly_cloudy_puresky

B.8. So partida perduda

Llicència: CC0 1.0

Modificacions: No

Ref.: https://freesound.org/people/Fupicat/sounds/538151/

B.9. So arrancar fruita

Llicència: CC0 1.0

Modificacions: No

Ref.: https://openverse.org/audio/42d2a0db-e82c-4732-aacf-986d7acfbec1?q=squashing+fruit&p=9

B.10. So perdre vida

Llicència: pixabay

Modificacions: Sí

Ref.: https://pixabay.com/sound-effects/search/effect/

B.11. So botó pressionat

Llicència: CC BY-NC 4.0

Autor: Mellau

Modificacions: No

Ref.: https://freesound.org/people/Mellau/sounds/506054/

B.12. So fletxa disparada

Llicència: CC0 1.0

Modificacions: Sí

Ref.: https://freesound.org/people/bruno.auzet/sounds/527435/

B.13. So col·lisió amb diana

Llicència: CC0 1.0

Modificacions: No

Ref.: https://openverse.org/audio/8e30b688-e42b-4e26-9202-96541ed81972?q=ding&p=17

B.14. Música de fons del menú

Llicència: CC BY 4.0
Autor: Sunsai

Modificacions: No

Ref.: https://openverse.org/audio/8d1de271-7de7-47c4-80f9-b8370f97a597?q=background+music&p=9

B.15. Música de fons de la partida

Llicència: CC BY 4.0
Autor: ShadyDave

Modificacions: No

Ref.: https://openverse.org/audio/b526fce0-0c51-4b5e-ac06-afd5af3faa02?q=background+music+forest&p=7