Institut Puig Castellar
Santa Coloma de Gramenet

Platform Jumping

CFGM Administració de Sistemes Microinformàtics i Xarxes

Josep Felipe Barrera Callupe
Michael Alejandro Castillo Girón
SX2A
2025 - 2026

Resum del projecte (màxim 250 paraules):

Aquest projecte, anomenat Platform Jumping, consisteix en el disseny, desenvolupament i desplegament web d'un videojoc de plataformes en 2D programat íntegrament amb el motor de codi obert Godot Engine 4. L'objectiu principal de l'equip ha estat crear una experiència interactiva robusta des de zero, la qual integra mecàniques cinemàtiques de moviment, un sistema de combat dinàmic a curta distància gestionat per hitboxes, control de salut del personatge, punts de control (checkpoints) persistents davant de caigudes i diferents menús d'interfície gràfica d'usuari (GUI).

El procés de desenvolupament ha requerit un aprenentatge intensiu del llenguatge GDScript per donar solució a reptes lògics complexos d'aquest software. Un dels pilars del treball ha estat la col·laboració conjunta mitjançant el control de versions amb Git i GitHub, superant conflictes d'integració de codi al repositori i aprenent a sincronitzar els fitxers de manera eficient.

En paral·lel, s'ha programat una pàgina web promocional i adaptativa utilitzant Visual Studio Code com a editor de codi amb les tecnologies HTML i CSS, la qual serveix com a plataforma per allotjar de manera pública el joc exportat en format HTML5 a través de GitHub Pages. El resultat final és un producte totalment funcional que demostra les competències tècniques adquirides al llarg del cicle en l'àmbit de la programació, la infraestructura de xarxes, l'optimització de recursos de hardware i la documentació tècnica de sistemes.

Paraules clau (entre 4 i 8):

- Godot Engine 4
- Videojoc 2D
- GDScript
- Desenvolupament de software
- Control de versions
- Allotjament web
- Multiplataforma

Abstract (in English, 250 words or less):

This project, titled Platform Jumping, focuses on the design, development, and web deployment of a 2D platformer video game built entirely with the open-source software Godot Engine 4. Our main goal was to construct a robust interactive experience from scratch, integrating fluid kinematic movement, short-range combat mechanics based on hitboxes, character health management, persistent checkpoints, and dynamic graphical user interfaces (GUI).

The development phase prioritized active learning of the GDScript language to solve complex logical and physics-based challenges within the game engine. Teamwork was a fundamental element, driven by version control through Git and GitHub, which required us to manage code conflicts and establish efficient workflows to maintain repository integrity.

Additionally, a responsive promotional website was developed using Visual Studio Code with HTML and CSS standards, functioning as a public host for the game exported in HTML5 via GitHub Pages. The final outcome is a fully functional software product that showcases the technical skills acquired during the vocational program in microcomputer systems and networks, particularly in programming, network infrastructure, hardware resource optimization, and technical project documentation.

Keywords (entre 4 i 8):

- Godot Engine 4
- 2D Video game
- GDScript
- Software development
- Version control
- Web hosting
- Cross-platform.

Índex

Llista de figures

1 Introducció

El projecte Platform Jumping consisteix en el disseny, desenvolupament i desplegament d'un videojoc de plataformes i acció en dues dimensions (2D), construït completament des de zero. La proposta neix de la inquietud de l'equip per recuperar i homenatjar l'essència dels jocs arcade clàssics, on la precisió extrema en el salt i la rapidesa en la resposta dels controls eren els pilars fonamentals de la diversió. No obstant això, el nostre objectiu no ha estat fer un clon del passat, sinó fusionar aquest estil retro amb mecàniques modernes de combat basades en hitboxes dinàmiques, sistemes de partícules actius per a l'ambientació i una estructura de control altament optimitzada perquè l'experiència de l'usuari sigui completament interactiva.

El disseny del joc posa a prova l'habilitat del jugador a través d'un entorn de tipus parkour vertical i horitzontal, on la coordinació és la clau per superar els diferents obstacles de l'escenari, esquivar trampes i derrotar enemics distribuïts de forma estratègica. Des de la vessant purament tècnica, el desenvolupament ha posat un èmfasi especial en programar una sincronització perfecta entre els fulls de sprites i el motor de físiques, garantint que el processament dels inputs del teclat reaccioni de manera immediata i sense cap tipus de lag. El resultat final és un prototip de software independent que aconsegueix equilibrar la dificultat clàssica amb un rendiment net i apte per a qualsevol usuari.

1.1 Context

Aquest projecte s'ha dissenyat i executat com a treball de síntesi final per a la superació del Cicle Formatiu de Grau Mitjà en Sistemes Microinformàtics i Xarxes (SMX). En un panorama tecnològic actual on el mercat de l'entreteniment està saturat de títols tridimensionals (3D) de nova generació que exigeixen un hardware d'alt rendiment i targes gràfiques d'últim model, vam veure que era millor fer un joc més lleuger i optimitzat en 2D. Com que és un projecte educatiu per a l'institut, ens ha tocat buscar-nos la vida de forma autodidacta per veure fins on podíem arribar amb l'editor de codi i el motor gràfic de codi obert Godot Engine 4 combinat amb el llenguatge GDScript.

Més enllà de la pròpia creació de la lògica del joc, aquest treball s'enmarca dins d'una infraestructura tecnològica interconnectada que posa a prova de forma directa les competències d'administració que hem après a classe. El desenvolupament exigeix coordinar de manera estricta un flux de treball col·laboratiu en un repositori públic a GitHub, dominant el control de versions per fer pushes, pulls i resoldre conflictes de branques en equip. A més, el context del projecte inclou la necessitat de generar una documentació tècnica detallada d'enginyeria de software i dur a terme el desplegament en producció d'una pàgina web promocional que allotgi l'aplicació en un servidor web real a Internet per donar visibilitat al producte final.

1.2 Justificació

La tria d'aquest projecte de final de cicle es justifica principalment per dos motius essencials que barregen l'interès vocacional amb l'aplicació pràctica d'estudis. En primer lloc, hi ha una forta motivació personal per comprendre la lògica de programació interna i l'arquitectura de sistemes que s'amaga darrere de la creació d'un videojoc, un sector industrial en constant creixement econòmic i tecnològic que cada cop requereix més perfils d'administració de xarxes. En segon lloc, el projecte es justifica com l'eina ideal per posar en pràctica de forma unificada els coneixements del cicle de SMX sobre jerarquia i estructures de fitxers, automatització de tasques, resolució de problemes de sistemes operatius i lògica informàtica bàsica.

El fet d'enfrontar-se a la creació d'un programa d'aquestes característiques des de zero, utilitzant un full de text en blanc, ens ha obligat a resoldre directament una sèrie de reptes de configuració que un software comercial ja fet o un instal·lador automatitzat mai ens podrien ensenyar. Hem hagut d'aprendre a gestionar vectors de força per a la gravetat, programar col·lisions rígides mitjançant càlculs geomètrics del motor, estructurar sistemes de persistència de dades locals per als checkpoints i dissenyar patrons bàsics d'intel·ligència artificial (IA) per al patrullatge dels enemics. Tot aquest esforç de recerca es tradueix en una comprensió molt més profunda de l'optimització de codi i la gestió de memòria, habilitats que considerem vitals per al nostre futur perfil professional com a tècnics en informàtica.

1.3 Objectius

1.3.1 Objectiu general

L'objectiu general d'aquest projecte és desenvolupar un videojoc de plataformes totalment funcional, garantint la correcta implementació de les mecàniques bàsiques del personatge (moviment, salt i atac), la gestió del sistema de salut i el comportament autònom dels enemics.

1.3.2 Objectius específics

- **Tècnics:** Implementar un sistema de moviment fluid amb atacs, moviment, salt… , crear una GUI que mostri la vida en temps real i un botó per pausar el joc.
- **Disseny:** Dissenyar nivells amb una dificultat progressiva que utilitzi correctament els TileSets i els fons de pantalla en paral·laxi. L'art del joc ha de ser de tipus pixelart i no 100% realista.
- **Mecàniques de Joc**: Configurar un sistema de col·lisions precís i una intel·ligència artificial (IA) bàsica per als enemics perquè patrullin zones determinades i reaccionin a la presència del jugador.
- **Gestió:** Utilitzar Git/GitHub per al control de versions, assegurant que el flux de treball sigui col·laboratiu i organitzat.
- **Web:** Crear una pàgina web promocional utilitzant HTML i CSS que inclogui informació del joc i botó per començar a jugar.
- **Aprenentatge Autònom**: Dominar el flux de treball amb GitHub per al control de versions i la resolució de conflictes en un entorn de desenvolupament col·laboratiu.

1.4 Estratègia i planificació del projecte

L'estratègia principal per dur a terme aquest projecte s'ha basat en el desenvolupament d'un producte nou des de zero, prioritzant l'aprenentatge actiu. Hem seguit un procés d'aprendre amb aquests tres punts:

- **Aprenentatge mitjançant tutorials**: Hem utilitzat guies especialitzades per comprendre la lògica de Godot i la seva sintaxi (GDScript).
- **Investigació autònoma**: Davant de reptes específics del disseny de nivells o mecàniques, hem consultat fòrums i documentació oficial a Internet.
- **Suport d'Intel·ligència Artificial**: En situacions on no trobàvem solucions clares o necessitàvem optimitzar fragments de codi concrets, hem fet servir la IA com a eina de consulta per desbloquejar el desenvolupament.

Aquesta estratègia ens ha permès tenir un control total sobre el codi i les mecàniques del joc, garantint que el resultat final s'ajusti exactament als nostres objectius inicials.

1.5 Metodologia de treball

S'ha seguit una metodologia àgil adaptada a l'equip de dues persones. Aquesta elecció es justifica per la necessitat de ser flexibles davant els reptes de programació que han anat sorgint.

- **Sprints**: Hem dividit el treball en cicles de dues setmanes. Cada cicle començava amb una planificació de tasques (per exemple: "Implementar el menú de pausa") i finalitzava amb una revisió del codi funcional.
- **Gestió de tasques**: Hem utilitzat eines digitals i reunions presencials per assignar rols: un membre s'ha centrat més en la lògica de programació i l'altre en el disseny de nivells i la web, tot i que ambdues parts han col·laborat en tot moment.
- **Control de versions (GitHub)**: Ha estat el pilar de la nostra metodologia. L'ús de branques (branches) ens ha permès treballar en funcionalitats noves sense trencar la versió principal del joc, aprenent a gestionar les fusions de codi de manera professional.

1.6 Estudi econòmic i pressupostari

Materials	**Descripció**	**Costos de manteniment**
Recursos de Programari	Godot, GitHub i VS Code	0,00 € (Open Source)
Recursos de Maquinari	Ús d'ordinadors personals	0,00 € (Recursos propis)

2 Descripció del projecte

2.1 Anàlisi de requisits Joc

Els requisits principals són que el jugador es pugui moure, saltar i atacar, també té vida i hi ha enemics amb vida que es mouen i poden fer mal al jugador.

Hi ha un menú d'inici on hi ha diferents botons per a entrar a jugar, anar a un mini tutorial i sortir del joc. Hi ha diversos nivells on es pot passar un a un i amb diferents enemics perquè no sigui senzill.

2.1.1 Requisits funcionals

- **Moviment i control del jugador**: El sistema ha d'executar un control cinemàtic precís en dues dimensions on l'usuari pugui desplaçar el personatge lateralment i realitzar salts fluids mitjançant el teclat, garantint que les físiques responguin de manera instantània i sense errors de col·lisió amb l'entorn.

- **Sistema d'animacions del jugador**: El motor gràfic ha de gestionar de forma reactiva les transicions dels fulls de sprites (spritesheets), canviant automàticament l'estat visual del personatge entre repòs (idle), carrera, salt i atac segons les accions i els inputs rebuts en temps real.

- **Mecànica de combat del jugador**: S'ha d'implementar una àrea de dany ofensiva (hitbox) a curta distància que s'accioni en prémer la tecla d'atac, la qual ha de ser capaç de detectar la presència d'entitats enemigues per intercanviar informació i reduir la seva salut.

- **Gestió de la vida i estat del jugador**: El joc ha de controlar internament els punts de salut del protagonista, emetent senyals cada vegada que rep dany i activant automàticament el protocol de reaparició (respawn) des de l'últim punt de control quan la vida arribi a zero.

- **Moviment i lògica dels enemics**: Les entitats enemigues (com la seta) han de funcionar de manera autònoma mitjançant un sistema de patrulla en línia recta sobre les plataformes, utilitzant sensors de tipus RayCast2D per detectar els límits del sòl i canviar de direcció automàticament.

- **Interacció i atac de l'enemic**: El sistema ha de permetre que els enemics infligeixin dany al jugador per contacte directe, restant un percentatge fix de salut (20 punts) i aplicant una força de retrocés (knockback) vectorial per expulsar el personatge fora del perill immediat.

- **GUI i barra de vida**: La pantalla ha d'incorporar un element visual fix (HUD) que mostri l'estat vital del jugador, el qual ha de ser completament reactiu i redibuixar-se instantàniament només quan rebi un senyal de canvi de salut.

- **Menú d'inici i mode de pausa**: L'aplicació ha d'oferir un menú principal funcional amb botons clars per iniciar la partida, accedir al tutorial de controls o sortir del joc, així com un menú de pausa que congelarà la simulació de les físiques i els enemics en prémer la tecla d'escapament.

2.1.2 Requisits no funcionals

- **Rendiment gràfic optimitzat**: El videojoc s'ha de dissenyar per mantenir de manera estricta una taxa d'execució de 60 fotogrames per segon (FPS) estables, evitant caigudes de rendiment en ordinadors de gamma mitjana com els que es troben habitualment a les aules de l'institut.

- **Elecció de l'estètica Pixel Art**: S'estableix com a requisit l'ús d'un disseny gràfic basat en píxels en dues dimensions, la qual cosa actua com una decisió tècnica d'optimització que redueix el consum de memòria RAM i simplifica la càrrega del processador gràfic.

- **Usabilitat i navegació intuïtiva**: Tota la interfície d'usuari, incloent-hi els menús d'inici, tutorial i pausa, ha de disposar d'un disseny net i accessible que permeti a qualsevol jugador interactuar amb els botons mitjançant el ratolí sense confusions.

- **Compatibilitat i exportació multiplataforma**: El nucli del codi del projecte en Godot 4 ha d'estar completament preparat i estructurat per permetre l'exportació directa i nativa cap a sistemes operatius d'escriptori com Windows i Linux.

- **Accessibilitat a través de l'entorn Web**: El joc ha de ser capaç d'executar-se de manera lleugera en format HTML5 des de qualsevol navegador d'Internet actual (Chrome, Firefox, Edge), integrant-se a la pàgina web promocional del projecte sense requerir instal·lació prèvia.

- **Optimització del pes dels fitxers**: El tamany total de l'aplicació exportada per a la web s'ha de mantenir el més baix possible per evitar llargs temps d'espera durant la descàrrega i garantir una càrrega fluïda fins i tot en xarxes amb connectivitats limitades.

2.1 Previsió de tasques d'investigació

1. Investigar el motor de desenvolupament que s'utilitzarà per a crear el joc
2. Avaluar els diferents tipus de jocs (2D, 3D, 4D…)
3. Buscar diferents estil d'art per al joc.
4. Estudiar la sintaxi del gdscript
5. Estudiar els nodes del godot
6. Investigar com crear escenes i les seves scripts
7. Determinar la millor manera de crear enemics
8. Investigar sistemes de vida del jugador i enemics
9. Provar diferents mecàniques de joc per a veure quins funcionen millor.
10. Avaluar el rendiment del joc i millorar diferents apartats

2.2 Tecnologies

2.2.1 Comparativa de les tecnologies valorades

- **Unity**: Aquesta eina es va valorar inicialment per ser el motor de desenvolupament més utilitzat a la indústria professional i per disposar d'una quantitat immensa de documentació i recursos a la xarxa. No obstant això, es va descartar ràpidament perquè requereix un programari molt pesat i exigent per als ordinadors de l'institut, la corba d'aprenentatge del llenguatge C# és massa lenta per al calendari disponible i la gestió de projectes en dues dimensions és menys nativa i intuïtiva que en altres alternatives.

- **Unreal Engine**: És un dels motors més potents del mercat actual, conegut pel seu alt rendiment gràfic i el seu sistema de programació visual. Es va descartar gairebé immediatament del projecte degut a que està enfocat de manera gairebé exclusiva al disseny tridimensional d'alta gamma, fet que hagués provocat que els nostres equips de treball no poguessin gestionar l'editor amb un mínim de fluïdesa, provocant congelacions i talls en el desenvolupament.

- **Roblox Studio**: Es va tenir en compte com a alternativa gràcies a la seva gran facilitat d'ús, una interfície altament intuïtiva i una compatibilitat nativa excel·lent amb gairebé qualsevol tipus de dispositiu del mercat. Malgrat aquests pros, es va haver de descartar a causa de les limitacions de la pròpia plataforma, ja que no permet exportar el codi de manera lliure i independent per integrar-lo en un servidor web propi ni generar un fitxer HTML5 estàndard, obligant a executar el joc sempre dins del seu propi ecosistema tancat.

- **Godot**: Va ser l'opció guanyadora i escollida per al projecte gràcies al seu caràcter de codi obert i totalment gratuït, així com pel seu pes ultra lleuger de menys de 100 MB que permet executar-lo sense instal·lació a qualsevol ordinador. El seu sistema d'organització basat en "Nodes" i "Escenes" s'adapta perfectament a les necessitats d'un desenvolupador principiant, i el seu rendiment per a jocs en dues dimensions és un dels més òptims i fluids de la informàtica actual.

2.2.2 Tecnologies escollides

- **Motor de Videojocs (Godot Engine)**: S'ha seleccionat la darrera versió estable d'aquest motor per tal d'aprofitar les noves millores en la gestió del node TileMapLayer i l'optimització de les partícules processades per la CPU. L'argument principal de la seva elecció és la seva capacitat nativa per exportar cap a la web (HTML5) de manera neta, generant fitxers poc pesats que faciliten que qualsevol usuari pugui jugar des d'un navegador d'Internet sense necessitat d'instal·lar cap programari addicional.

- **Entorns de Desenvolupament (Godot Editor i Visual Studio Code)**: S'ha optat per treballar amb l'editor integrat de Godot per a la programació dels scripts del joc gràcies a la seva perfecta connexió amb el sistema de nodes. En paral·lel, s'ha utilitzat Visual Studio Code per al disseny de la pàgina web promocional, ja que és un editor de text modular molt potent que inclou connectors que faciliten l'escriptura i la depuració d'estils i etiquetes web.

- **Llenguatges de Programació (GDScript, HTML i CSS)**: S'ha triat GDScript per a tota la lògica interna del videojoc (moviment, dany, físiques i interfície) a causa de la seva gran similitud amb Python, fet que ens ha facilitat la sintaxi i el desenvolupament. Per a la part de presentació i difusió del projecte, s'ha fet servir HTML i CSS, ja que són els estàndards de la indústria web necessaris per incrustar el fitxer del joc i assegurar un disseny adaptatiu i compatible amb ordinadors i dispositius mòbils.

- **Gestió de Projecte i Repositori (Git i GitHub)**: S'ha escollit aquest sistema de control de versions per permetre el treball col·laboratiu i simultani entre nosaltres dins del mateix repositori de codi. Tot i els conflictes inicials que ens van obligar a avançar poc a poc, l'eina s'ha justificat com a indispensable per coordinar el lliurament dels fitxers del joc i mantenir un historial de canvis segur davant de qualsevol pèrdua d'informació, actuant jo mateix com a propietari (owner) per controlar les fusions finals de codi.

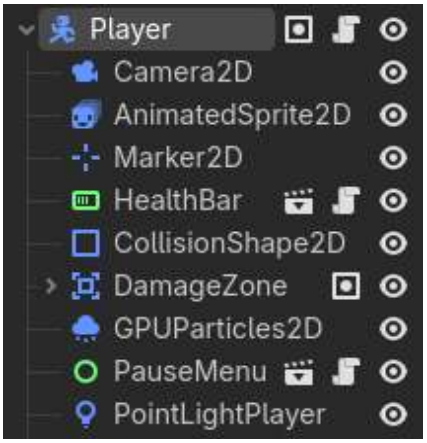
2.3 Estructura del projecte

L'arquitectura de Platform Jumping ha estat dissenyada sota una metodologia modular i jeràrquica, aprofitant el sistema de nodes i escenes propi de Godot Engine 4. Aquesta organització permet que cada element del joc funcioni de manera autònoma però coordinada, facilitant la detecció d'errors i l'escalabilitat del projecte. A continuació es detallen els components principals i la seva funció tècnica:

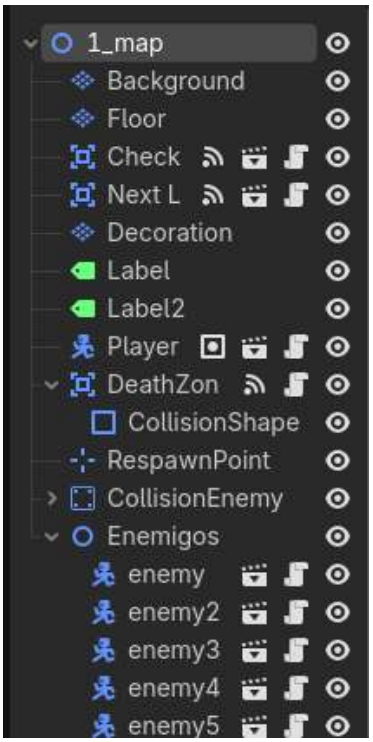
Organització de les Escenes i Nodes principals:

L'escena del jugador és l'entitat més complexa del joc. Està construïda sobre un node de tipus CharacterBody2D, que és el més adequat per a personatges controlats per l'usuari en un entorn 2D. Mitjançant el script de moviment, es gestionen variables constants com la gravetat, la velocitat de marxa i la força de salt. S'utilitza la funció move_and_slide(), que

calcula automàticament les col·lisions i permet que el personatge llisqui per les superfícies sense quedar-se encallat. A més, aquest node conté el control d'animacions amb els diferents llistats de fotogrames o frames. Segons l'estat del jugador (si s'està movent, si és a l'aire o si està atacant), el codi canvia l'animació activa per oferir una resposta visual immediata. Finalment, la gestió de la càmera està integrada dins de la mateixa escena per garantir que el focus sempre estigui sobre ell, configurant el paràmetre Position Smoothing per evitar moviments bruscs i definint els límits de coordenades per evitar que l'usuari vegi el "buit" fora del mapa.



L'escena del món actua com a node mestre o "contenidor" general on s'instancien la resta d'elements, i la seva funció és coordinar l'inici de la partida i gestionar els punts de reaparició. El sistema de mapes d'aquest nivell està dibuixat utilitzant capes de cel·les (TileMapLayers). Això permet optimitzar el rendiment del processador, ja que en lloc de carregar centenars d'imatges individuals, el motor només ha de dibuixar una quadrícula amb referències visuals. Per donar profunditat a l'entorn i al fons del joc, s'ha implementat un fons de parallaxi (ParallaxBackground). Aquest component té una imatge posada una darrera l'altra a tot el mapa, des de l'inici fins al final, que es mou més a poc a poc que el personatge per simular la distància.



El sistema de checkpoints i fogueres utilitza nodes d'àrea (Area2D) distribuïts pel mapa. La seva funció és detectar quan el cos del jugador entra en el seu radi d'acció, moment en què envien una ordre al script global per actualitzar la posició de respawn. Això és vital per a l'experiència d'usuari, ja que evita haver de repetir tot el nivell des de l'inici quan es cau.

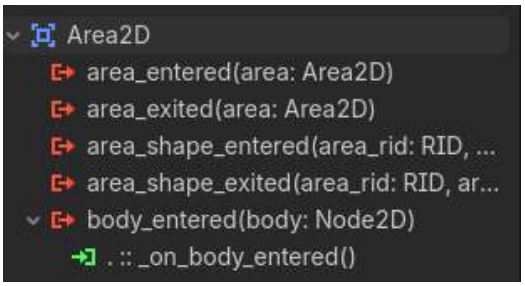
Pel que fa als enemics i la intel·ligència artificial, cada enemic és una instància d'una escena base que comparteix una lògica de patrulla; utilitzen nodes de detecció anomenats RayCast2D, que actuen com a "ulls" invisibles per detectar si hi ha terra davant seu o si hi ha un mur, canviant la direcció del moviment automàticament. Per últim, la interfície d'usuari (HUD) conté la barra de vida i el comptador de punts en una capa independent de la càmera. En estar dins d'un CanvasLayer, aquests elements es mantenen fixos a la pantalla del jugador independentment d'on es mogui el personatge.

Comunicació entre Nodes i Codi:

El sistema de senyals (Signals) és la part més tècnica de tota l'estructura del projecte. En lloc de tenir un codi "massa pesat" que ho controli tot de manera centralitzada, cada node s'encarrega exclusivament de la seva part i avisa els altres components mitjançant senyals automàtics. Per exemple, quan el jugador toca un enemic, el motor emet un senyal de "dany" que la interfície rep de manera asíncrona per restar un percentatge de vida directament a la barra de la pantalla.

Pel que fa als scripts de lògica, cada escena té un fitxer amb extensió .gd associat on es defineix el seu comportament per línies de codi. Aquests fitxers estan organitzats de manera que les variables més importants (com la velocitat de moviment o la vida màxima) estiguin exposades a l'inspector visual de Godot 4. D'aquesta manera, permet als membres de l'equip fer ajustos ràpids des de l'inspector visual a mitja partida.

```
18  func _on_body_entered(body: CharacterBody2D) -> void:
19      # Solo funciona si quien entra pertenece al grupo "Player"
20      if body.is_in_group("Player"):
```



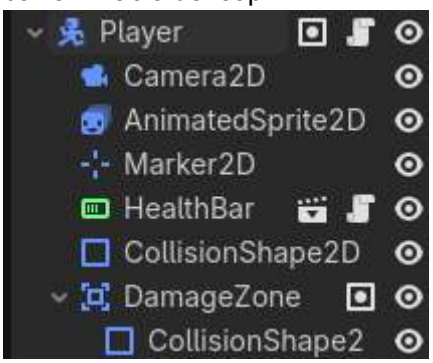
2.4 Descripció dels components

Aquest apartat descriu de manera individual i conceptual cadascuna de les peces fonamentals que configuren l'arquitectura de Platform Jumping, especificant-ne la utilitat, el funcionament intern i les tecnologies integrades del motor de jocs.

2.4.1 Component 1 Jugador

Perquè el nostre personatge es pugui moure pel mapa, el primer que hem fet és crear un objecte físic utilitzant un node de tipus *CharacterBody2D*, que ens permet controlar els seus moviments mitjançant codi. A sobre d'aquest node, hem hagut de posar-li una caixa de col·lisió o hitbox en forma de càpsula (*CollisionShape2D*) configurada exactament a la mateixa capa de físiques que el terra i les parets de l'escenari; si no ho haguéssim fet així, els dos elements no podrien interactuar entre ells i el personatge travessaria el mapa caient al buit infinit a l'instant.

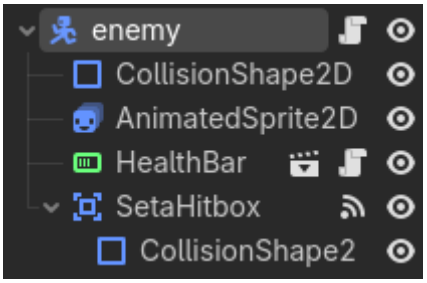
A part d'això, hem afegit una segona caixa de col·lisió al personatge mitjançant un node *Area2D*, que serveix per definir el rang o l'abast on el jugador pot fer mal amb la seva espasa per matar els enemics. Aquesta hitbox d'atac està programada per estar completament desactivada per defecte mentre caminem o saltem, i només s'activa durant una fracció de segon quan el jugador prem el botó d'atacar (la tecla J), tornant-se a apagar de manera automàtica quan acaba l'animació del cop.



2.4.2 Component 2 Enemic

L'enemic tipus "seta" funciona com un obstacle mòbil dins del nivell i, igual que el jugador, necessita la seva pròpia caixa de col·lisió (*CollisionShape2D*) per poder recolzar-se sobre el terra rígid i xocar contra les parets sense traspassar-les. El seu comportament intern està programat per fer una ruta automàtica d'anada i tornada al llarg de la plataforma on el col·loquem, canviant de sentit cada vegada que detecta un obstacle al davant.

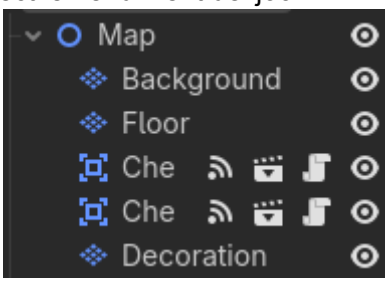
Perquè pugui fer la seva funció ofensiva, l'enemic porta integrada una *Area2D* independent al voltant del seu cos que actua com a zona de perill per al jugador. Quan el personatge s'apropa massa i el seu cos xoca contra aquesta àrea, s'activa un script que calcula l'impacte, reduint automàticament 20% de vida del jugador i aplicant-li un impuls físic de retrocés per allunyar-lo del perill.



2.4.3 Component 3 Mapa

Tot el disseny dels nivells està construït utilitzant la tecnologia de *TileMapLayers* de Godot 4, la qual ens ha permès dividir el disseny del mapa en tres capes independents de rajoles gràfiques per treballar de manera molt més còmoda per treballar de manera còmoda i evitar confusions a l'hora d'afegir detalls. La primera capa es troba al fons del tot i serveix per pintar el fons (background), la segona capa és la intermèdia i s'utilitza exclusivament per col·locar el terra sòlid i les parets, i la tercera capa va per sobre i està reservada per als objectes decoratius que no afecten el moviment.

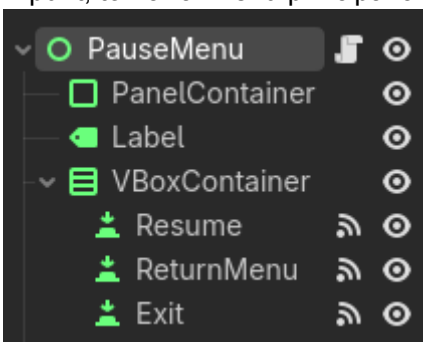
Gairebé tots els blocs de la capa intermèdia tenen activades les seves pròpies colisions geomètriques, ja que són les estructures on volem que el personatge hagi de caminar, saltar o esquivar barrancs per no caure. Per donar-li un toc visual molt més viu i dinàmic a l'escenari, hem completat aquest component afegint nodes de partícules de CPU (*CPUParticles2D*) que simulen ràfegues de vent i petits efectes de pols ambientals que volen per la pantalla sense que afecti el rendiment del joc.



2.4.4 Component 4 GUI

La interfície gràfica és la capa visual que permet a l'usuari interactuar amb el flux del joc i veure el seu estat de salut, utilitzant nodes de tipus *CanvasLayer* perquè els menús i els textos es dibuixin sempre fixes a la pantalla i no es moguin de lloc quan la càmera avança. D'una banda, tenim el Menú d'Inici de l'aplicació, que compta amb diferents botons programats amb senyals per llançar la partida directa, obrir el nivell de tutorial on s'expliquen els controls o tancar el joc de manera segura.

D'altra banda, durant la partida el jugador té visible en tot moment una barra de vida que és reactiva, el que significa que només es redibuixa i canvia el seu tamany quan rep un senyal de text que diu que la vida del personatge ha canviat. A més, si el jugador prem la tecla "Esc" (Escapament), el codi activa el Menú de Pausa, el qual congela per complet les físiques del mapa i el moviment dels enemics per mostrar uns botons que permeten reanudar la partida des del mateix punt, tornar al menú principal o sortir definitivament.



2.4.5 Component 5 Checkpoint

El sistema de punts de control s'ha dissenyat com un element d'ajuda interactiu que es basa en una foguera que inicialment està apagada i mostra una flama de color taronja. Utilitzant un node de tipus Area2D, el component es manté a l'espera que el personatge passi per davant i, en el moment en què es detecta la col·lisió, canvia l'animació mitjançant codi per mostrar un foc de color blau elèctric i augmentar la seva il·luminació, indicant visualment a l'usuari que la seva posició ha estat guardada de manera segura.

El funcionament intern d'aquest component es comunica amb un script global del joc que emmagatzema les coordenades X i Y exactes d'aquesta foguera blava. La lògica que hem programat dicta que, si el jugador calcula malament un salt i cau al buit per un barranc, el motor llegeix aquesta variable i el teletransporta directament a sobre de la foguera activa; en canvi, si l'usuari mor per complet perquè la seva salut arriba a zero lluitant contra els enemics, el checkpoint s'invalida i el sistema el obliga a reiniciar el nivell des de l'inici per mantenir el repte del joc.

2.4.5 Component 5 Càmera i Límit del Mapa

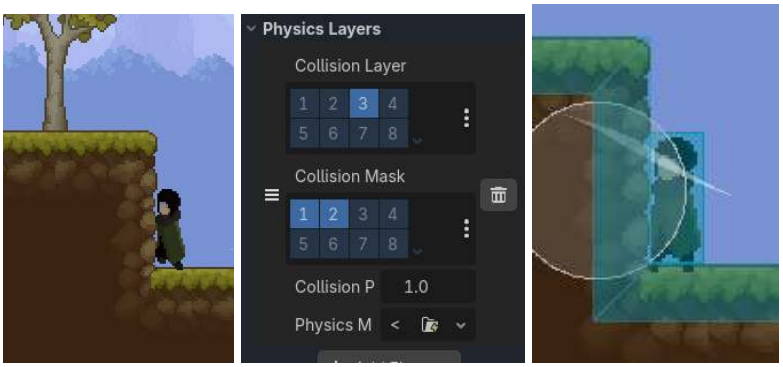
Perquè el jugador pugui explorar els escenaris verticals i horitzontals, hem hagut d'integrar un component de càmera dinàmica utilitzant un node *Camera2D* col·locat com a fill directe dins de l'estructura del jugador, fent que la pantalla segueixi de forma automàtica tots els desplaçaments del personatge a temps real. Per evitar que el moviment de la càmera sigui rígid o provoqui marejos, hem activat la propietat de suavitzat (Position Smoothing), creant una petita inèrcia que fa que el visor acompanyi els salts d'una manera molt més fluida i natural.

Un dels reptes interns d'aquest component ha estat configurar els límits d'enfocament gràfic (Limits) introduint de manera manual les coordenades en píxels del tamany màxim del nostre *TileMapLayer*. Mitjançant aquesta configuració, aconseguim bloquejar la càmera perquè s'aturi en arribar als marges inferiors i laterals de l'escenari, evitant que es vegi el fons buit de l'editor de Godot quan el personatge s'apropa a les vores i assegurant que l'usuari només vegi l'entorn dissenyat per a la partida.

2.5 Definició de les tasques

2.5.1 Detecció de col·lisions del jugador i mapa

Aquesta prova s'ha centrat a validar la integritat física del món virtual, assegurant que el node *CharacterBody2D* del jugador interactuï de manera precisa amb les geometries definides al *TileMapLayer*. Per dur a terme aquesta investigació, s'han configurat les màscares de col·lisió (Collision Masks) de manera que el personatge reconegui exclusivament els elements sòlids del terreny, evitant així que el motor hagi de processar càlculs innecessaris amb elements decoratius o de fons. Durant els tests realitzats, s'ha forçat el desplaçament del jugador contra diversos tipus de superfícies, com parets verticals i sostres, per comprovar que les funcions internes de Godot, com *is_on_floor()* i *is_on_wall()*, retornin els valors correctes en cada frame de l'execució. El resultat satisfactori d'aquestes proves ha permès concloure que la combinació de formes de col·lisió rectangulars per al mapa i una càpsula per al jugador ofereix el millor equilibri entre realisme i fluïdesa, evitant que el personatge quedi bloquejat en les unions de les rajoles o en les cantonades del nivell.



2.5.2 Interacció entre enemic i jugador

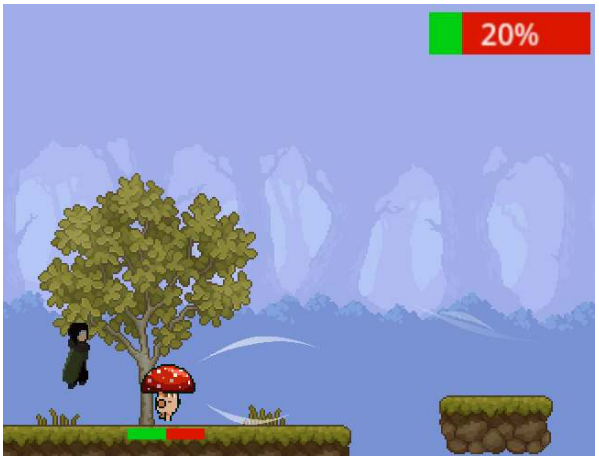
La investigació de la interacció de combat s'ha enfocat principalment en el desenvolupament d'un sistema de dany reactiu que inclogui tant la reducció de salut com una resposta física de retrocés, coneguda tècnicament com a knockback. Mitjançant l'ús de nodes Area2D integrats en l'enemic "seta", s'ha programat un detector que, en entrar en contacte amb el cos del jugador, executa una funció específica de dany definida en el script del personatge. Aquesta prova ha permès determinar les constants de força necessàries per crear una paràbola de desplaçament involuntari, concretament aplicant un impuls vertical de 500 píxels i un desplaçament horitzontal de 250 píxels en direcció

oposada a l'impacte. Aquesta solució tècnica no només serveix per penalitzar el jugador amb la resta de 20 punts de vida, sinó que també proporciona un feedback visual imprescindible que evita que el jugador quedi atrapat dins del radi d'atac de l'enemic, garantint així un combat molt més just i equilibrat.

```gdscript
# Knockback para empujar al jugador
func apply_knockback(source_position: Vector2):
    var knockback_direction = -1 if source_position.x > global_position.x else 1

    # Cambiamos la velocidad directamente.
    velocity.x = knockback_direction * 500
    velocity.y = -250

    move_and_slide()
```



2.5.3 Menu d'inici

El disseny i la implementació del menú d'inici s'han plantejat com una prova de gestió del *SceneTree*, on s'ha volgut comprovar l'eficàcia del canvi d'escenes mitjançant el sistema de senyals de Godot. El procés d'investigació ha consistit a connectar diversos nodes de tipus *Button* a funcions que utilitzen la instrucció *get_tree().change_scene_to_file()*, assegurant que la transició entre el menú principal, el nivell de tutorial i el primer nivell del joc sigui neta i lliure d'errors de càrrega. S'ha verificat que el botó de tutorial permet a l'usuari familiaritzar-se amb els controls de parkour i combat abans d'enfrontar-se al repte real, mentre que el botó de sortida tanca de manera segura tots els processos de l'aplicació. Les conclusions extretes d'aquesta prova confirmen que una interfície ben estructurada amb senyals permet un flux de navegació fluid que millora significativament l'accessibilitat del videojoc des del primer moment.

```gdscript
extends Control

func _on_play_pressed() -> void:
    get_tree().change_scene_to_file("res://Scenes/map1.scn")

func _on_controls_pressed() -> void:
    get_tree().change_scene_to_file("res://Scenes/tutorial.tscn")

func _on_exit_pressed() -> void:
    get_tree().quit()
```

2.5.4 Mode Pausa

El desenvolupament del mode de pausa ha requerit una investigació sobre el comportament dels processos en temps real i la jerarquia de nodes dins del motor de videojocs. S'ha creat un component que conté tres botons interactius ("Volver", "Menú inicio" i "Salir") situat en una capa de llenç independent per evitar que la càmera del joc l'afecti. El repte tècnic principal ha estat configurar la propietat Process Mode per permetre que el menú segueixi responent als inputs de l'usuari mentre la resta de la lògica del joc (enemics, gravetat i temporitzadors) es troba en un estat de congelació total. Aquesta prova ha estat fonamental per garantir que el jugador pugui gestionar les seves sessions de joc amb total control, permetent-li retornar a l'acció exactament en el mateix punt on s'havia quedat o abandonar la partida de manera ordenada si així ho desitja.

2.5.5 Checkpoint

La prova del sistema de punts de control s'ha centrat en la persistència temporal de la ubicació del jugador per evitar la frustració que suposaria haver de reiniciar el nivell des de l'inici després de cada error. El procediment seguit ha consistit a crear un objecte interactiu que actua com a punt de guardat dinàmic: inicialment presenta una flama taronja que simbolitza el seu estat inactiu, però en entrar en contacte amb el jugador, canvia la seva animació a un foc blau i emmagatzema les coordenades globals del personatge en una variable global de tipus Singleton. S'ha comprovat que aquest sistema funciona correctament en casos de caiguda al buit, on el motor teletransporta el personatge a la foguera blava més propera, mentre que en cas de mort total per pèrdua de salut, el sistema reinicia el nivell per mantenir el nivell de repte. Aquesta distinció lògica entre caiguda i mort ha estat clau per polir l'experiència de joc i el balanç de dificultat.

2.5.6 Passar de nivell

Per validar la transició entre escenaris, s'ha investigat un mètode de càrrega dinàmica que no requereixi la creació de scripts individuals per a cada porta o final de fase. S'ha implementat una *Area2D* de meta, senyalitzada visualment amb una foguera, que té la capacitat de llegir el nom del nivell en el qual es troba el jugador actualment. El procés de comprovació ha consistit a programar una petita rutina que suma una unitat al número identificador del nivell i concatena el resultat amb l'extensió de fitxer ".tscn" per buscar la següent escena en el sistema de fitxers del projecte. Aquest enfocament modular ha demostrat ser extremadament eficient, ja que permet l'expansió del joc amb nous nivells sense haver de modificar el nucli del codi, assegurant que el pas de la fase de tutorial al Nivell 1, i d'aquest al Nivell 2, sigui automàtic i lliure de complicacions tècniques.

```gdscript
extends Area2D

const FILE_BEGIN = "res://Scenes/map"

func _on_body_entered(body: CharacterBody2D) -> void:
	if body.is_in_group("Player"):
		var current_scene_file = get_tree().current_scene.scene_file_path
		var next_level_number = current_scene_file.to_int() + 1

		var next_level_path = FILE_BEGIN + str(next_level_number) + ".scn"
		get_tree().change_scene_to_file(next_level_path)

```

2.5.7 Sistema de Partícules i Ambientació

Com a prova final de rendiment i qualitat visual, s'ha dut a terme una investigació sobre la implementació de partícules per simular condicions ambientals com el vent i la pols de l'escenari. S'han utilitzat nodes de tipus *CPUParticles2D* per generar efectes visuals que reaccionen a les accions del jugador, com per exemple una petita explosió de pols que s'activa quan el personatge aterra d'un salt o un flux constant de partícules de vent que creuen la pantalla per donar sensació d'altura. El procés de validació ha consistit a monitorar la càrrega de processament durant situacions on moltes partícules coincideixen amb múltiples enemics en pantalla. Les conclusions d'aquesta prova han confirmat que, gràcies a l'optimització de Godot 4, és possible mantenir una experiència visualment rica i professional sense sacrificar la taxa de fotogrames per segon, aconseguint que el món de Platform Jumping se senti viu i dinàmic.

2.6 Definició de les funcionalitats

2.6.1 Control i moviment del personatge

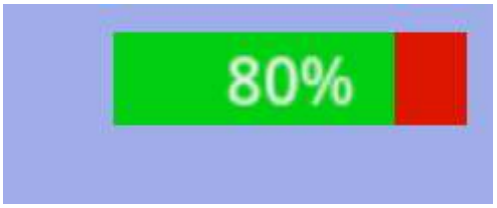
Aquesta funcionalitat permet a l'usuari desplaçar el protagonista pel món virtual de manera fluida i precisa. El procés es realitza mitjançant la captura d'esdeveniments de teclat que modifiquen en temps real el vector de velocitat del node principal. Conceptualment, el sistema calcula la fricció quan el jugador s'atura i l'acceleració quan comença a córrer, assegurant que el moviment no sigui instantani ni rígid, sinó que tingui una certa inèrcia que millori l'experiència de joc. El salt s'ha dissenyat com un impuls instantani cap amunt que contraresta la força de la gravetat fins que el personatge arriba al punt àlgid de la paràbola. **Estat: Implementada totalment.**

2.6.2 Sistema de combat i detecció d'impactes

La funcionalitat de combat ofereix al jugador la capacitat de defensar-se dels enemics mitjançant un atac d'espasa de curta distància. Quan l'usuari prem la tecla d'atac, el sistema activa temporalment una "hitbox" o àrea de dany que es desplaça juntament amb el personatge. El procés intern consisteix a detectar si algun cos amb l'etiqueta "Enemy" es troba dins d'aquesta àrea en el moment de l'impacte; si és així, s'envia una instrucció per reduir la salut de l'enemic i reproduir una animació de mort. Aquesta mecànica inclou també un temps de recuperació (cooldown) per evitar que el jugador pugui prémer el botó de manera infinita sense penalització. **Estat: Implementada totalment**.

2.6.3 Gestió de salut i interfície reactiva (HUD)

Aquesta funcionalitat permet visualitzar l'estat vital del jugador i gestionar les conseqüències del dany rebut. El procés es basa en un sistema de senyals on, cada vegada que la variable de salut canvia, s'emet una alerta que la interfície gràfica rep per actualitzar els cors o la barra visual en pantalla. Si la salut arriba a zero, el sistema bloqueja els controls del jugador i inicia el protocol de derrota. S'ha buscat que la interfície sigui totalment reactiva, la qual cosa significa que la GUI no consumeix recursos constantment, sinó que només es redibuixa quan rep la notificació de canvi d'estat del jugador. **Estat: Implementada totalment.**



2.6.4 Persistència del progrés mitjançant Checkpoints

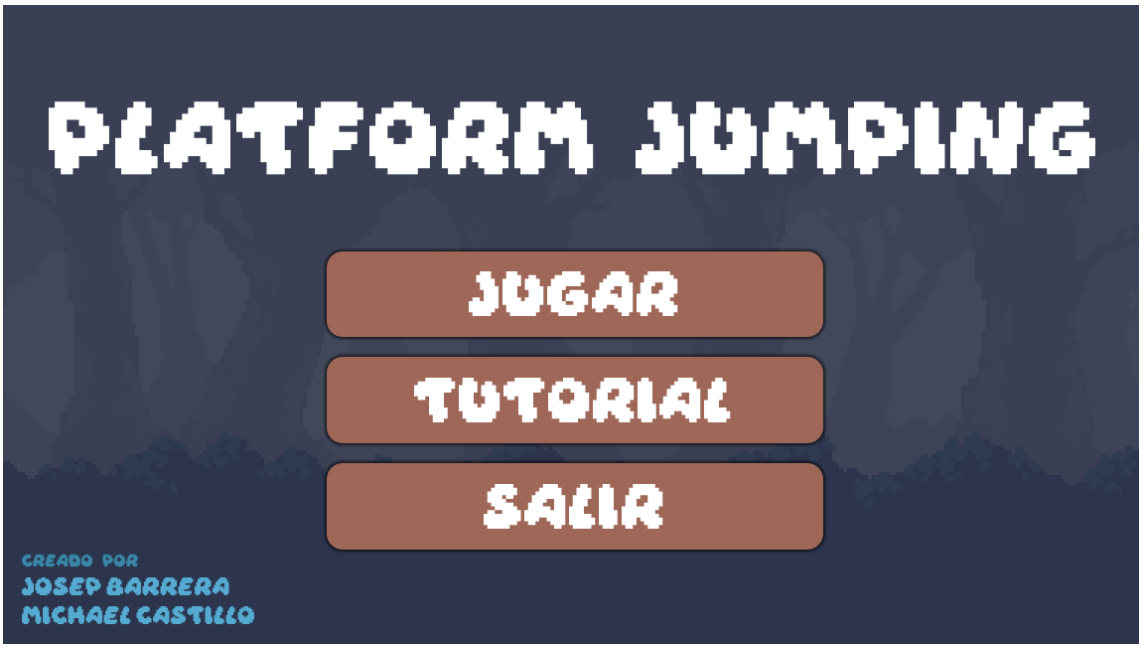
Aquesta funcionalitat permet guardar la posició de l'usuari de manera temporal durant la sessió de joc. Quan el personatge col·lideix amb una foguera o bandera, el sistema activa una rutina que guarda les coordenades X i Y exactes. Conceptualment, el checkpoint actua com un "ancla" de seguretat; si el personatge surt dels límits del mapa o toca un obstacle mortal, el procés de respawn el col·loca en la darrera posició guardada en lloc de reiniciar tot el nivell. Això redueix la frustració i millora el flux de joc. **Estat: Implementada totalment.**

2.6.5 Transició dinàmica de nivells i meta

La funcionalitat de finalització de fase permet al jugador progressar en la història del joc. En arribar a l'objectiu final del nivell, el procés detecta la col·lisió i executa una seqüència de càrrega per a la següent escena. El sistema s'ha programat de manera modular per reconèixer automàticament quin és el següent fitxer que ha de carregar segons l'ordre numèric dels nivells. Aquesta estructura permet afegir contingut nou al projecte de manera extremadament senzilla. **Estat: Implementada totalment.**

2.6.6 Sistema de menús i configuració

Aquesta funcionalitat gestiona el flux d'entrada i sortida del joc, així com les pauses durant la partida. El procés permet a l'usuari navegar per una interfície visual on pot començar una nova partida, aprendre els controls en un tutorial dedicat o tancar l'aplicació. El menú de pausa, per la seva banda, permet aturar la simulació física del joc sense perdre el context de la partida, oferint la possibilitat de reprendre l'acció o abandonar-la en qualsevol moment. **Estat: Implementada totalment.**


PLATFORM JUMPING
JUGAR
TUTORIAL
SALIR
CREADO POR
JOSEP BARRERA
MICHAEL CASTILLO

3 Gestió de riscos i contingències

El desenvolupament d'aquest projecte ha estat marcat per una gran capacitat d'adaptació de l'equip davant d'un dels riscos més habituals en la creació de programari: la necessitat de canviar de rumb a mig camí. Inicialment, durant els primers mesos del curs, vam estar treballant en un prototip i una idea de videojoc completament diferents. No obstant això, a finals de gener ens vam adonar que el projecte no ens acabava de convèncer, no complia les expectatives de jugabilitat que volíem i el calendari se'ns tirava a sobre. Com a mesura de contingència, vam prendre la difícil decisió de descartar tot el que teníem fet i començar Platform Jumping des de zero, una decisió arriscada però completament necessària per salvar el projecte final.

Aquest gir radical ens va obligar a reestructurar la planificació i a treballar a contrarellotge durant la segona meitat del curs per recuperar el temps perdut. Per aconseguir lliurar el joc a temps, vam aplicar una estratègia de reducció de l'abast, focalitzant tots els nostres esforços a polir al màxim el nucli essencial del joc (moviment, col·lisions i combat) i deixant les idees més complexes per a futures actualitzacions. L'èxit d'aquesta maniobra demostra que vam saber gestionar la pressió i el temps de manera madura, aconseguint en la meitat de temps un producte final molt més estable i divertit que el disseny original que havíem plantejat.

4 Desplegament de la plataforma web i servidor

Com a estudiants del cicle de Sistemes Microinformàtics i Xarxes, la implementació d'aquest projecte no es limitava només a programar el videojoc dins de l'editor de Godot, sinó que incloïa el repte tècnic de desplegar-lo a Internet perquè fos completament accessible per a qualsevol usuari sense necessitat d'instal·lar res. Per fer-ho possible, hem configurat el motor per exportar el projecte en format Web (HTML5), un procés que genera un paquet de fitxers combinats que inclou el codi executable comprimit en *.wasm* (WebAssembly), els scripts de llançament en *.js* i la pàgina d'índex *.html* principal que s'encarrega d'incrustar el llenç del joc.

Per a l'allotjament d'aquesta plataforma web promocional, hem pres la decisió de xarxa d'utilitzar el servei de GitHub Pages, vinculat directament al nostre repositori públic (https://github.com/josepb80/platfrom_jumping). El principal avantatge d'aquesta infraestructura és que funciona com un servidor de contingut estàtic altament optimitzat que ens permet actualitzar el joc de manera immediata cada vegada que fem un git push correcte. Hem integrat els fitxers exportats dins de la ruta de la nostra web promocional (https://josepb80.github.io/platfrom_jumping/Web/index.html), configurant el codi HTML i els estils CSS per assegurar que el joc s'escali correctament.

5 Conclusions

5.1 Conclusions generals del projecte

El desenvolupament de Platform Jumping ha esdevingut una experiència de superació i aprenentatge integral que ha permès augmentar els coneixements obtinguts al cicle. El motiu de major orgull per a l'equip és haver aconseguit que tot el nucli essencial del videojoc, les col·lisions, el sistema de moviment, la sincronització de les animacions i les mecàniques de combat estigui completament implementat i funcioni de manera robusta. A més, s'ha aconseguit el repte d'afegir elements i detalls extra perquè l'experiència no resulti monòtona i cada nivell posseeixi la seva pròpia identitat visual i mecànica.

El factor més sorprenent de tot aquest procés ha estat la nostra capacitat de reacció davant les adversitats del calendari, ja que a finals de gener es va prendre la difícil decisió de descartar un projecte anterior que no ens acabava de convèncer. Desenvolupar aquest nou videojoc des de zero ens va obligar a treballar a contrarellotge, aconseguint en la meitat de temps un producte molt més polit, sòlid i professional que l'original, fet que demostra la nostra maduresa tècnica a l'hora de gestionar imprevistos en un projecte informàtic.

5.2 Consecució dels objectius

Pel que fa als objectius plantejats, podem determinar que s'han assolit de manera molt satisfactòria les fites principals de programació, disseny i desplegament de la plataforma web promocional. Tot i la limitació temporal provocada pel canvi de rumb del projecte a mig curs, el nucli jugable actual és completament estable i lliure d'errors crítics. El personatge es mou amb fluïdesa, els enemics patrullen de forma automatitzada i el HUD es comunica perfectament amb el sistema gràcies a una arquitectura dirigida per esdeveniments i senyals.

No obstant això, algunes de les idees més ambicioses que havíem planificat s'han hagut de posposar de manera conscient i reservar com a propostes de millora per no comprometre el lliurament oficial. Entre les funcionalitats que s'han hagut d'acotar es troben la implementació d'un sistema de loot o botí en eliminar els enemics, el disseny de nivells addicionals, l'obertura de cofres interactius que aportin habilitats especials al personatge i la programació de grans bosses finals. Deixar aquestes funcions per a més endavant ha estat una decisió de gestió correcta per prioritzar la qualitat del nucli del joc.

5.3 Valoració de la metodologia i planificació

El treball en equip entre nosaltres ha estat una lliçó magistral de coordinació gràcies a l'ús de GitHub, tot i que el control de versions ens va plantejar situacions de crisi complexes. Durant les primeres fases, en realitzar canvis grans simultanis, ens trobàvem sovint amb conflictes de fusió on, en fer un *git push* o un *git pull*, el codi es corrompia i es perdien línies de treball sencer. La nostra solució inicial d'emergència consistia a clonar el repositori des de zero, la qual cosa ens feia perdre el progrés immediat i alentia el nostre ritme.

Aquesta dinàmica ens va obligar a canviar radicalment l'estratègia i a aprendre a avançar de manera més cautelosa, de poc en poc, establint un protocol de comunicació verbal estricte. A partir d'aquell moment, vam decidir notificar-nos cada canvi abans d'aplicar-lo per no sobreescriure mai el fitxer de l'altre i vam acordar que, en ser jo el propietari del repositori de GitHub, faria sempre el *git push* primer per garantir la integritat de la branca mestra, minimitzant així els errors d'integració.

5.4 Visió de futur

Com que Platform Jumping s'ha dissenyat s'ha sota una arquitectura modular molt flexible en Godot 4, el projecte té una gran viabilitat per seguir creixent de manera autònoma. El nostre objectiu ideal a llarg termini és expandir de manera massiva el nombre d'escenaris disponibles i introduir batalles contra un cap final cada cert nombre de fases superades per potenciar la corba de repte. En l'apartat d'ambientació, es preveu integrar música i efectes d'àudio personalitzats per a cada mapa segons la seva estètica, així com afegir una major densitat de partícules visuals.

Finalment, des de la vertent purament mecànica de l'aplicació, es vol implementar una selecció de diferents personatges jugables, un inventari dedicat, noves varietats d'atacs i la capacitat de recollir claus per obrir els cofres i obtenir objectes únics de cada enemic eliminat. Totes aquestes millores estan pensades per transformar el prototip funcional actual en un videojoc d'acció i plataformes complet, profund i preparat per a una futura distribució o fins i tot per a la seva exportació a dispositius mòbils

6. Glossari

- **Area2D**: Node bidimensional de Godot Engine utilitzat per detectar la superposició, entrada o sortida de cossos físics en un espai determinat sense aplicar forces físiques reals. Al projecte s'utilitza per gestionar l'abast dels atacs del jugador i les zones d'activació dels checkpoints o fogueres.

- **Branch (Branca)**: Línia de desenvolupament independent dins d'un repositori de control de versions (Git). Permet als desenvolupadors treballar en funcionalitats noves paral·leles (com dissenyar menús o provar codi) de manera aïllada i segura sense alterar el codi principal del joc fins que la nova funció estigui completament provada.

- **CanvasLayer**: Node especial de renderitzat que permet dibuixar elements gràfics en una capa independent de la càmera del joc. S'utilitza per crear interfícies d'usuari (menús o barres de salut) que queden fixes i immòbils a la pantalla de l'usuari mentre el personatge es mou pel mapa.

- **CharacterBody2D**: Tipus de node especialitzat de Godot dissenyat específicament per a cossos en dues dimensions controlats directament per línies de codi (scripts). Proporciona mètodes interns com *move_and_slide()* que calculen de manera eficient la gravetat, els vectors de velocitat i els lliscaments suaus sobre superfícies sòlides sense que el personatge travessi les parets o el terra.

- **CPUParticles2D**: Component del motor gràfic dedicat a la generació massiva de partícules visuals bidimensionals processades de manera directa pel processador central (CPU) del maquinari. En el projecte serveix per renderitzar ràfegues de vent i efectes de pols ambiental sense penalitzar el rendiment global de l'ordinador.

- **CSS (Cascading Style Sheets)**: Llenguatge de fulls d'estil utilitzat en el desenvolupament web per definir i aplicar la capa de disseny visual, colors, tipografies i comportament adaptatiu (responsive) a la infraestructura estructurada d'un document HTML.

- **Frame (Fotograma)**: Cadascuna de les imatges individuals consecutives que es renderitzen a la pantalla per segon per simular el moviment continu de l'animació o del propi joc. El joc està optimitzat per mantenir una taxa estable de 60 fotogrames per segon (FPS).

- **GDScript**: Llenguatge de programació d'alt nivell, interpretat i orientat a objectes, dissenyat de forma nativa per a Godot Engine. Té una sintaxi molt neta i similar a Python, la qual cosa redueix la corba d'aprenentatge i facilita la comunicació directa amb l'inspector del motor de videojocs.

- **Git**: Programari lliure i de codi obert dissenyat per al control de versions descentralitzat. Permet registrar l'historial complet de modificacions d'un directori de fitxers de codi, assegurant el flux de treball col·laboratiu d'un equip i evitant pèrdues accidentals de fitxers.

- **GitHub / GitHub Pages**: GitHub és una plataforma d'allotjament al núvol que utilitza el control de versions de Git per emmagatzemar i coordinar repositoris de codi. Per altra banda, GitHub Pages és una funcionalitat específica d'aquest servei que permet allotjar i desplegar de forma gratuïta pàgines web i aplicacions estàtiques a Internet.

- **Godot Engine 4**: Motor de videojocs multiplataforma de codi obert i ultra lleuger que funciona mitjançant una arquitectura jeràrquica basada en nodes i escenes independents. És la tecnologia central que processa tota la lògica, gràfics, àudio i físiques del videojoc.

- **GUI (Graphical User Interface)**: Interfície Gràfica d'Usuari que engloba el conjunt de finestres, textos, formularis i botons visuals interactius que permeten a una persona navegar i comunicar-se amb les funcions de programari d'un ordinador de manera senzilla mitjançant el ratolí o el teclat.

- **Hitbox**: Àrea o caixa de col·lisió invisible associada a una entitat o objecte geometre de l'editor. S'utilitza per calcular en quines coordenades precises de la pantalla interaccionen dos cossos entre si, com ara registrar quan l'arma del jugador infligeix dany dins del radi d'acció d'un enemic.

- **HTML5 (HyperText Markup Language, versió 5)**: Estàndard actual de llenguatge de marques que defineix l'arquitectura i l'estructura bàsica del contingut de les pàgines web. Ofereix capacitat nativa per incrustar codi binari complex (com videojocs exportats) directament al navegador d'Internet sense necessitat d'instal·lar cap plugin extern.

- **HUD (Heads-Up Display)**: Conjunt d'elements visuals, indicadors o icones que es mostren sobreposats a la pantalla del jugador durant la partida. Té l'objectiu de transmetre informació vital en temps real (com els punts de vida restants) sense interrompre l'acció del joc.

- **Input**: Qualsevol tipus de senyal, ordre o dada d'entrada que s'introdueix en un sistema operatiu o programari de simulació a través d'un perifèric de maquinari, principalment a través de les tecles o el ratolí d'un ordinador.

- **Knockback**: Efecte o força vectorial de retrocés físic immediat aplicada sobre un cos o personatge virtual com a reacció en el moment exacte de patir un impacte o atac enemic. S'utilitza com a resposta de disseny per expulsar l'usuari fora de l'àrea de perill immediat.

- **Parallax (Efecte Parallaxi)**: Tècnica d'il·lusió gràfica en 2D que consisteix a desplaçar de manera independent diverses capes de fons de l'escenari a velocitats diferents de la del personatge principal. Això genera una falsa sensació de profunditat espacial en tres dimensions a l'entorn.

- **Pull / Push**: Ordres de transmissió de dades utilitzades habitualment en el programari Git. El pull serveix per descarregar les actualitzacions d'un servidor extern cap al repositori local, i el push serveix per pujar i fusionar les línies de codi noves creades des de l'ordinador local cap a la xarxa del servidor públic.

- **RayCast2D**: Node de Godot que funciona simulant un sensor de raig lineal invisible llançat en una direcció determinada de les coordenades gràfiques. S'utilitza per intercanviar informació ràpida a cada frame, per exemple per detectar si l'enemic té un sòl continu davant dels seus peus o si arribarà a un mur per obligar-lo a girar l'orientació.

- **Respawn (Reaparició)**: Protocol automatitzat pel codi d'un programari que reubica la instància d'un personatge o jugador dins unes coordenades de posició concretes del mapa virtual un cop les seves variables de vitalitat han arribat a zero.

- **Sprite / Spritesheet**: Un sprite és un mapa de bits o imatge individual bidimensional que representa un element dinàmic o fix del joc. Un spritesheet (full de sprites) és un únic fitxer d'imatge gran que agrupa desenes d'aquests fotogrames en una quadrícula organitzada, permetent al motor gràfic crear animacions fluides en reproduir-los ordenadament a gran velocitat.

- **TileMapLayer**: Tecnologia i eina de disseny modular pròpia de Godot Engine 4 que permet construir nivells i entorns de joc pintant i col·locant de forma estructurada petites rajoles o cel·les gràfiques predefinides (Tiles) sobre una graella de coordenades. Minimitza dràsticament l'ús de RAM i de potència gràfica de l'ordinador durant l'execució.

- **Visual Studio Code (VS Code)**: Editor de codi font modular, gratuït i molt potent desenvolupat per Microsoft. Disposa d'una gran varietat d'extensions i connectors informàtics que faciliten la sintaxi correcta a l'hora de desenvolupar pàgines web amb HTML i CSS.

7. Bibliografia

1. **AJ08Coder.** (2023, 23 diciembre). *How to make a landing dust particle in Godot 4 in 3 minutes* [Vídeo]. YouTube. https://www.youtube.com/watch?v=5_3NrtkI478

2. **Bissash.** (2024, 22 junio). *How to make Wind Particles in Godot Engine 4* [Vídeo]. YouTube. https://www.youtube.com/watch?v=G5b4agx75Fc

3. **Bitlytic.** (2023, 4 abril). *How you can easily make your code simpler in Godot 4* [Video]. YouTube. https://www.youtube.com/watch?v=74y6zWZfQKk

4. **Chris' Tutorials.** (2022a, 25 febrero). *Easy swing animations you can swap out (Pickaxe, sword, axe) ~ Godot 4 tutorial* [Video]. YouTube. https://www.youtube.com/watch?v=-BNpTIp7wc8

5. **Chris' Tutorials.** (2022b, 20 abril). *How to fix pixel art blurriness in Godot 4* [Video]. YouTube. https://www.youtube.com/watch?v=whd1oCWSXLI

6. **Chris' Tutorials.** (2022c, 14 junio). *How to add global music and 2D audio in Godot 4 ~ Tutorial for beginners* [Video]. YouTube. https://www.youtube.com/watch?v=7UmyaSIpsh0

7. **Chris' Tutorials.** (2022d, 28 junio). *How to add Tilemap collisions in Godot 4* [Video]. YouTube. https://www.youtube.com/watch?v=1Uk1yhGtnOo

8. **DevWorm.** (2022, 17 julio). *How to use PARTICLE SYSTEMS in Godot* [Video]. YouTube. https://www.youtube.com/watch?v=W5lR6Q31VFQ

9. **DevWorm.** (2023, 21 diciembre). *Everything to know about PARTICLES in Godot 4* [Video]. YouTube. https://www.youtube.com/watch?v=yWIH7hHfWyU

10. **DevWorm.** (2024a, 3 mayo). *Create a Complete Platformer Game in Godot 4 (step by step guide)* [Vídeo]. YouTube. https://www.youtube.com/watch?v=6F4I8efwnUc

11. **DevWorm.** (2024b, 6 mayo). *Create Complete Player ANIMATIONS in Godot 4 (step by step)* [Video]. YouTube. https://www.youtube.com/watch?v=JIRz5PqxlGg

12. **DevWorm.** (2024c, 14 mayo). *Create a Complete COMBAT SYSTEM in Godot 4 (step by step)* [Vídeo]. YouTube. https://www.youtube.com/watch?v=rUE9gA8OltA

13. **DevWorm.** (2024d, 15 junio). *How to Create Customizable Health Bars in Godot 4* [Vídeo]. YouTube. https://www.youtube.com/watch?v=CSYxd94mnOU

14. **findemor.** (2024, 24 abril). *Cómo usar Godot y Aprender desde CERO a hacer juegos* [Video]. YouTube. https://www.youtube.com/watch?v=-_LiMyZGoXw

15. **Firebelley Games.** (2025, 27 septiembre). *How to build Godot Editor tools like a Pro* [Video]. YouTube. https://www.youtube.com/watch?v=VyTys5oCdN8

16. **GameDevKnight.** (2023, 18 junio). *Melee Attack - Godot 4 Platformer Tutorial - Part 6* [Video]. YouTube. https://www.youtube.com/watch?v=2n1kILlyx8M

17. **IcyEngine.** (2024, 2 septiembre). *Godot 2D Sprite Animation - The Basics in 4 Minutes* [Video]. YouTube. https://www.youtube.com/watch?v=-f1bHR0iiEY

18. **Indierama.** (2024a, 4 julio). *Collision LAYERS / MASKS en Godot: ¡YA es hora de que los ENTIENDAS!* [Video]. YouTube. https://www.youtube.com/watch?v=WmpBGQC8LY8

19. **Indierama.** (2024b, 15 noviembre). *Cómo crear TILEMAPS en Godot (Godot 4.3+)* [Video]. YouTube. https://www.youtube.com/watch?v=I_R-3QuLQMA

20. **Jh soft.** (2022, 18 diciembre). *Clonar un repositorio Git de GitHub 2026* [Video]. YouTube. https://www.youtube.com/watch?v=rAnn6vtLm90

21. **Karto.** (2024, 28 agosto). *Godot Control Node (UI) Masterclass* [Video]. YouTube. https://www.youtube.com/watch?v=5Hog6a0EYa0

22. **KiriSoft Games.** (2023, 21 septiembre). *Falling leaves tutorial using Particles 2d | Godot tutorial |* [Video]. YouTube. https://www.youtube.com/watch?v=n1HPoBWTf1s

23. **KiriSoft Games.** (2025, 28 octubre). *Working on my indie game "The Twins" | Pixel Art + Dev Stream* [Video]. YouTube. https://www.youtube.com/watch?v=AZLn400BoVc

24. **KobeDev.** (2024, 17 enero). *Make a Pause Menu in Godot 4+ in 5 Minutes!* [Vídeo]. YouTube. https://www.youtube.com/watch?v=e9-WQg1yMCY

25. **LuisCanary.** (2021a, 16 diciembre). *Juego de Plataformas 2D/Godot Tutorial/Introduccion Godot/1-Capitulo/Programacion Videojuegos* [Video]. YouTube. https://www.youtube.com/watch?v=F3T_ZhllzJs

26. **LuisCanary.** (2021b, 23 diciembre). *Juego de Plataformas 2D/Godot Tutorial/Crear el Mapa/Godot/2-Capitulo/Programacion videojuegos* [Video]. YouTube. https://www.youtube.com/watch?v=SR7mdh0_i6Q

27. **LuisCanary.** (2021c, 28 diciembre). *Juego de Plataformas 2D/Godot Tutorial/Movimiento Personaje/Godot/3-Cap/Programacion videojuegos* [Video]. YouTube. https://www.youtube.com/watch?v=NeUS3Ytjty4

28. **LuisCanary.** (2024, 20 mayo). *Start MENU en Godot/Main Menu/ Facil y Sencillo para 3D y 2D* [Video]. YouTube. https://www.youtube.com/watch?v=hyTu5ZTR5Yk

29. **LuisCanary.** (2025, 5 mayo). *Como Crear Mapas 2D en Godot con Tilemap Layer!* 🎨✨🕹️ [Video]. YouTube. https://www.youtube.com/watch?v=A4zU3p1EXVY

30. **LuisCanary.** (2026, 2 febrero). *Aprende TODOS los NODOS 2D de GODOT 4* 🎮🔥 *| Tutorial DEFINITIVO en 10 MIN* ⏱️ [Video]. YouTube. https://www.youtube.com/watch?v=Vd2rF0fEjVY

31. **Mostly Mad Productions.** (2025, 6 agosto). *Checkpoint System in Godot 4.4* [Vídeo]. YouTube. https://www.youtube.com/watch?v=BJdRBp0IKmQ

32. **Olav3D Tutorials.** (2024, 29 mayo). *How to change the game background color in Godot* [Video]. YouTube. https://www.youtube.com/watch?v=YS0K51kt8R8

33. **Paper Mouse Games.** (2024a, 31 mayo). *How to make 2D player animations in Godot* [Video]. YouTube. https://www.youtube.com/watch?v=H_3HEzOqso0

34. **Paper Mouse Games.** (2024b, 7 junio). *Make tile terrains for platformers in Godot 4* [Video]. YouTube. https://www.youtube.com/watch?v=h3TVHdnHQkE

35. **PlugWorld.** (2023, 26 julio). *Godot 4: Switching levels made easy* [Vídeo]. YouTube. https://www.youtube.com/watch?v=GZrALMvOwY8

36. **Queble.** (2025, 22 octubre). *Advanced Hitbox System in Godot! (hurtbox, hitlogging, etc.)* [Video]. YouTube. https://www.youtube.com/watch?v=cX-vzfmzjnE

37. **Rafa Fiedo.** (2021a, 18 marzo). *How to do Enemy AI 2D - melee attack, turn around in Godot 3? Tutorial* [Video]. YouTube. https://www.youtube.com/watch?v=4WywpSBncFI

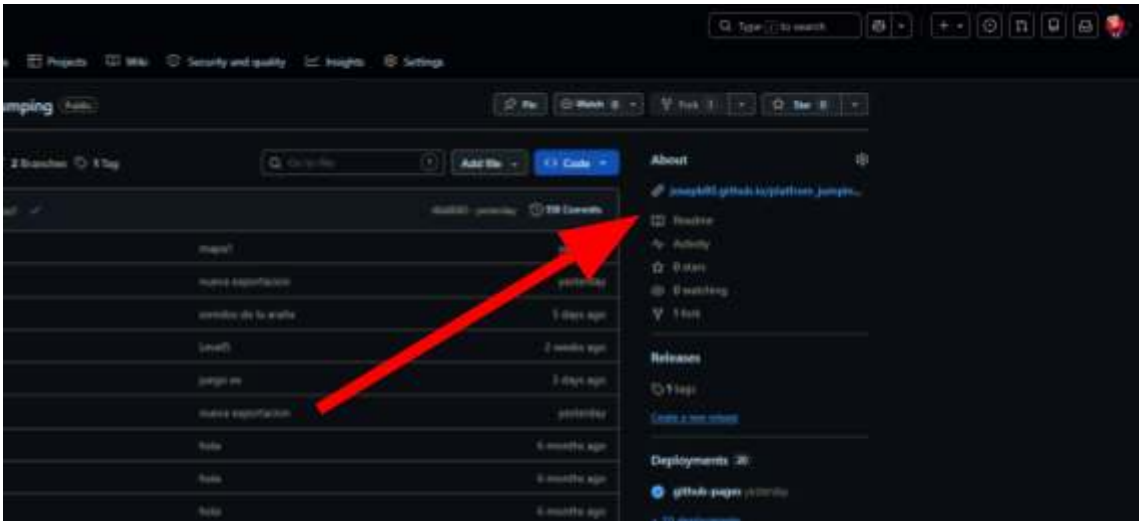
38. **Rafa Fiedo.** (2021b, 1 abril). *How to make CHECKPOINT in Godot 3? Tutorial* [Video]. YouTube. https://www.youtube.com/watch?v=cC6TynnRY7U

39. **Xorunix.** (2024, 9 diciembre). *¡CREA un MENU en GODOT en 10 MINUTOS: Guía Rápida y Fácil!* [Video]. YouTube. https://www.youtube.com/watch?v=8adi_q3eQPc

40. **Xorunix.** (2025, 2 enero). *¡EXPORTA TU JUEGO a WEB en GODOT en 5 MINUTOS! | Tutorial Paso a Paso* [Video]. YouTube. https://www.youtube.com/watch?v=JAe9j-9n0Tw

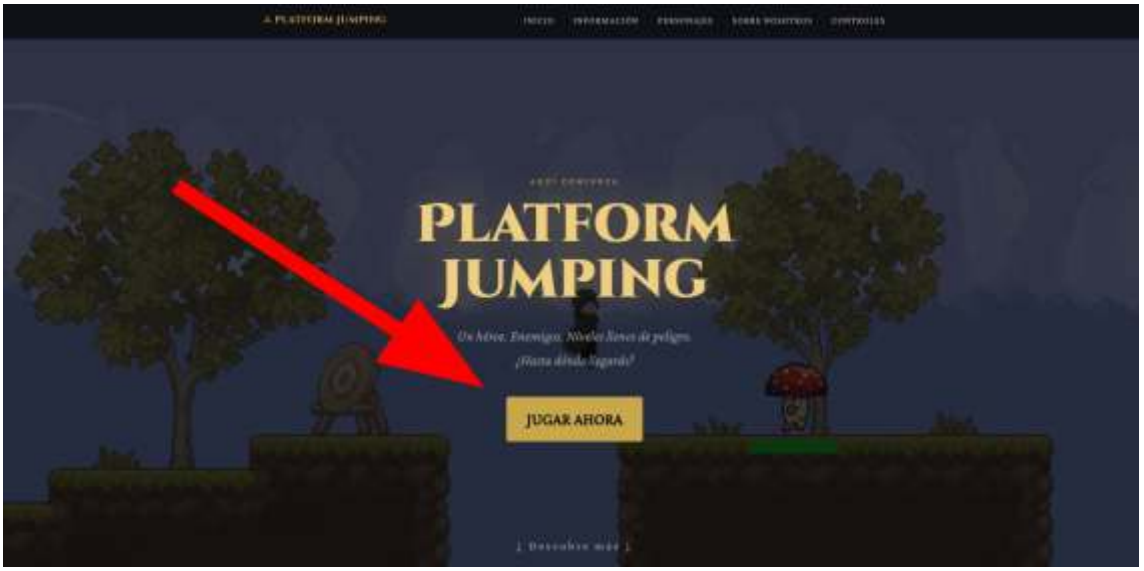
8. Annexos

1. **Com accedir al joc**
   Per poder jugar a Platform Jumping, el primer pas és accedir a l'enllaç del desplegament web que es troba a la descripció del nostre repositori de GitHub.



Quan s'accedeix a l'enllaç, es carregarà la pàgina web prèvia del joc. En aquest espai es pot consultar informació detallada sobre el projecte abans de començar la partida, com ara els controls, els personatges, la informació i altres dades d'interès.

Per començar a jugar, només cal fer clic al botó central “Jugar Ahora” i el joc s'iniciarà directament al navegador de manera automàtica.



2. Controls del personatge

Perquè el jugador pugui conèixer els controls abans o durant la partida, hem implementat dues opcions diferents per consultar-los:

a. Des de l'interior del joc

A la part superior de la pàgina web prèvia, hi ha un menú de navegació on es troba l'apartat de "Controls". En fer-hi clic, la pantalla es desplaçarà fins a la secció on s'explica detalladament la distribució del teclat i les accions bàsiques per interactuar amb l'entorn.

b. Des de l'interior del joc

Si el jugador ja ha iniciat el videojoc, al menú principal d'inici hi trobarà un botó anomenat “Tutorial”. En prémer-lo, s'obrirà una pantalla informativa interactiva on es mostren quins botons s'han de prémer per moure el personatge, com saltar, com utilitzar el sistema de combat i s'expliquen breument les mecàniques bàsiques que es trobarà al mapa.

3. Guia de la interfície i menús

Per facilitar la navegació de l'usuari, el joc compta amb un sistema de menús intuïtiu dividit en dues pantalles principals:

a. Quan el joc s'executa per primera vegada, es mostra la pantalla d'inici amb tres botons principals:

   i. **Jugar**: Inicia la partida i transporta el jugador directament al primer nivell del videojoc.

   ii. **Tutorial**: Obre la pantalla d'ajuda (esmentada a l'apartat anterior) per conèixer els controls i mecàniques.

   iii. **Salir**: Tanca l'aplicació o finalitza l'execució de la sessió de joc.

b. Durant la partida, si el jugador prem la tecla de pausa (per exemple, Esc), el joc es congela completament (atura el moviment del personatge, enemics i animacions) i es desplega una interfície amb tres opcions:

   i. **Volver**: Repèn la partida exactament en el mateix punt on s'havia quedat, desplaçant el menú de pausa i reprenent el temps.

   ii. **Menú inicio**: Abandona la partida actual i torna a carregar la pantalla principal d'inici del videojoc.

   iii. **Salir**: Surt definitivament del joc.