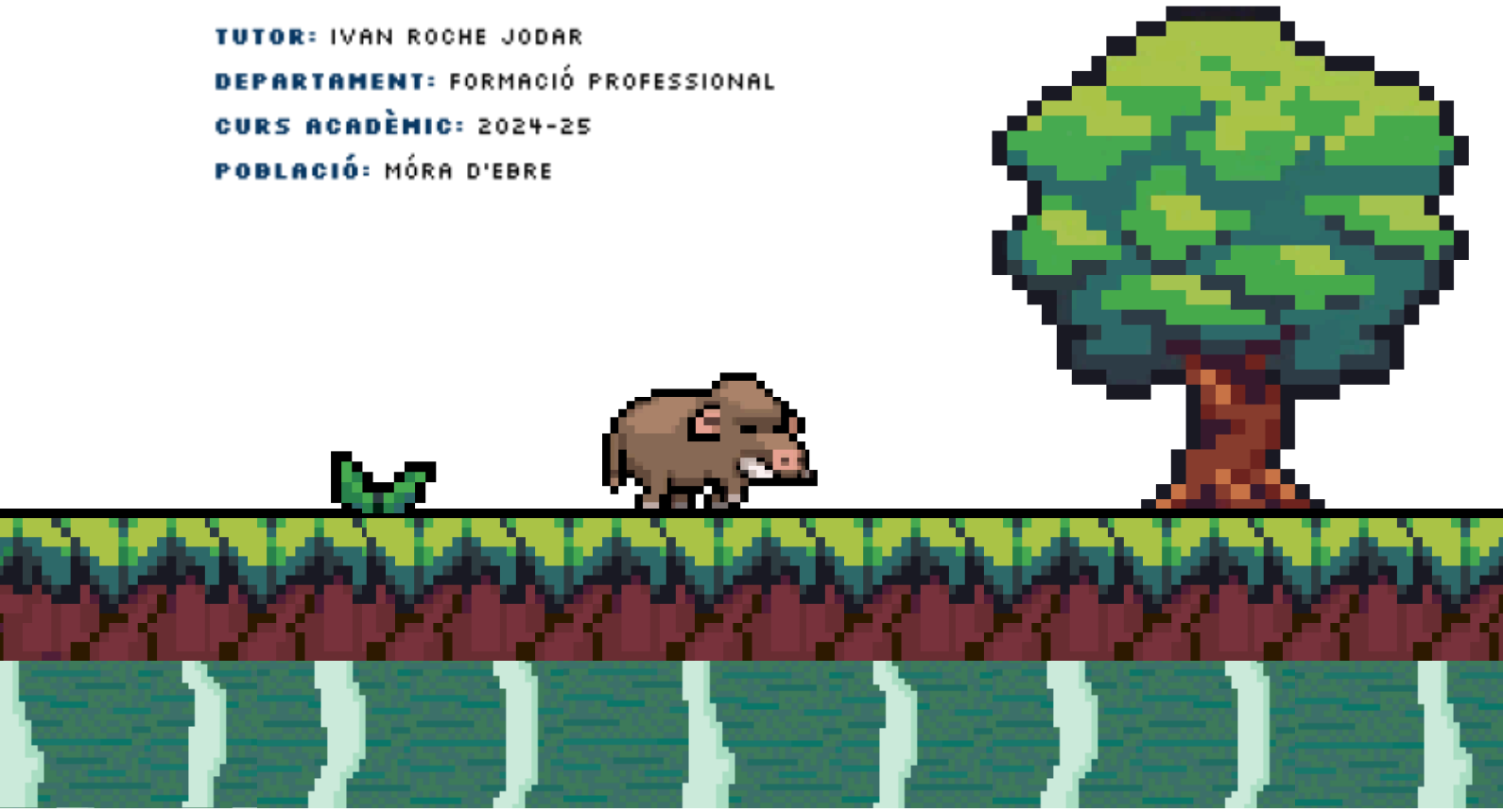
Generalitat de Catalunya
Departament d'Educació
Institut Julio Antonio

LO PESCADOR

CREACIÓ D'UN VIDEOJOC AMB UNITY

TUTOR: IVAN ROCHE JODAR
DEPARTAMENT: FORMACIÓ PROFESSIONAL
CURS ACADÈMIC: 2024-25
POBLACIÓ: MÓRA D'EBRE



INSTITUT JULIO ANTONIO
MÓRA D'EBRE

Una bona idea es pot convertir en un gran joc si s'executa de manera adequada.

- Shigeru Miyamoto -

RESUM

Resumen

Nuestro trabajo de investigación consta de un marco teórico en el que se explora la creación de un videojuego utilizando el motor gráfico multiplataforma 2D y 3D "Unity[1]". Nuestro videojuego está desarrollado en dos dimensiones y programado en el lenguaje de programación "C#[2]", diseñado para ejecutarse en sistemas operativos[3] Windows. La temática del videojuego se centra en un personaje principal, un pescador, que inicia un recorrido por el río Ebro desde la presa de Riba-roja d'Ebre hasta la desembocadura del río en el Mar Mediterráneo, en la localidad de Deltebre, provincia de Tarragona.

Durante la travesía, el jugador se enfrentará a distintos enemigos temáticos de la zona, como siluros, moscas negras o caracoles manzana. Estos enemigos pueden herir al personaje y reducir su vida hasta cero. Cuando el personaje es derrotado, vuelve al inicio, ofreciendo así una experiencia más desafiante y dinámica. Para combatir a los enemigos, el personaje cuenta con un arma principal: una piedra.

El videojuego pretende ser una fuente de entretenimiento a la vez que también busca educar a los jugadores sobre la flora y fauna del río Ebro en Cataluña. A través de esta experiencia interactiva, esperamos que los jugadores desarrollen una mayor apreciación por el patrimonio natural y cultural de la región.

[1] Motor gráfico especializado en el diseño de videojuegos que permite crear aplicaciones en dos y tres tamaños.
[2] Lenguaje de programación diseñado para la creación de videojuegos y desarrollado por Microsoft.
[3] Conjunto de los diferentes programas que controlan el funcionamiento de un ordenador.

Abstract

Our research project focuses on creating a video game using the 2D and 3D game engine "Unity[4]". Our game is developed in two dimensions and programmed in "C#[5]", designed to run on Windows operating systems[6]. It tells the story of a fisherman traveling along the Ebre River, starting at the Riba-roja d'Ebre dam and ending at the river's mouth in the Mediterranean Sea, near Deltebre in Tarragona.

During the journey, the player faces enemies from the region, such as bullheads and apple snails, which can harm the character and reduce their health. If the character loses all health, they return to the start, making the game more challenging. The player can fight back using a basic weapon: a stone.

The game is not only intended to be a source of entertainment, but also seeks to educate about the plants and animals of the Ebre river. Through this experience we hope this interactive experience helps players appreciate the region's natural and cultural heritage.

[4] Graphic engine specialized in the design of video games that allows you to create applications in two and three dimensions.
[5] Programming language designed for the creation of video games and developed by Microsoft.
[6] Different programs that control the operation of a computer.

ÍNDEX

1 INTRODUCCIÓ

Quan vam començar a pensar en el tema del nostre Treball de Recerca, ens vam adonar que una gran part de la societat desconeix l'enorme esforç que hi ha darrere del desenvolupament d'un videojoc. Això ens va motivar a explorar sobre aquest tema i buscar una manera de fer visible aquest treball, considerant que la millor forma de fer-ho era desenvolupar-ne un nosaltres mateixos.

La temàtica del videojoc es basa en un escenari proper a la nostra zona: les Terres de l'Ebre. El jugador explora un món inspirat en la vegetació que envolta el riu Ebre, amb un pescador que lluita contra les espècies invasores que l'habiten. A través de diversos pobles colindants, el joc ofereix batalles amb animals alhora que introdueix conceptes culturals.

A més, aquest treball ens ha permès integrar coneixements tant de programació com de disseny. Per exemple, hem dissenyat els personatges i les estructures que apareixen al videojoc, i també hem aplicat conceptes de programació per a la coordinació del jugador amb l'ambient. Finalment, hem adquirit habilitats en gestió de projectes, treball en equip i resolució de problemes per assolir un producte final.

Aquest treball representa un repte acadèmic al mateix temps que ens proporciona experiència pràctica en el desenvolupament de videojocs, un camp en constant creixement i amb un alt potencial professional.

1.1 Motivacions personals i justificació del tema

La motivació principal per emprendre aquest projecte ha estat la nostra passió pels videojocs i la curiositat per entendre el procés complet de la seva creació. Des del principi, ens vam plantejar fer un videojoc com a part del nostre Treball de Recerca per aprendre i visibilitzar l'esforç que implica aquest tipus de desenvolupament. Moltes persones desconeixen la complexitat que hi ha darrere d'un videojoc, fins i tot els més senzills, i considerem que aquest projecte seria una manera ideal de demostrar-ho.

Inicialment, vam adoptar la idea de crear un joc protagonitzat per un ninja que controlava elements naturals com el foc, l'aigua, la terra i l'aire. Tot i això, després d'analitzar la viabilitat del projecte, vam descartar aquesta opció per la seva excessiva complexitat i per la presència de molts jocs amb una temàtica similar. Això ens va portar a buscar una idea més original i propera.

Finalment, ens vam decantar per ambientar el videojoc al riu Ebre, un element emblemàtic de les nostres terres. Amb aquesta decisió, no només podíem crear un joc innovador i educatiu, sinó també homenatjar el nostre territori i donar a conèixer el problema de les espècies invasores. Així, la temàtica del joc no només tenia un propòsit lúdic, sinó també un component de sensibilització mediambiental.

Aquest projecte ens ha permès unir les nostres passions amb l'objectiu de crear un videojoc que no només diverteixi, sinó que també eduqui els jugadors sobre la fauna del riu Ebre, al mateix temps que promogui la valoració del patrimoni de les Terres de l'Ebre.

1.2 Objectius del treball

Els objectius del nostre treball tenen la intenció de guiar tot el procés de recerca i desenvolupament del videojoc:

1. Aprendre a utilitzar el motor gràfic "Unity" per crear videojocs en dues dimensions.
2. Dominar els conceptes bàsics del llenguatge de programació "C#", com ara els controls del personatge, la interacció amb els enemics, la gestió dels sistemes de vida i els efectes sonors dinàmics.
3. Dissenyar personatges i escenaris com el protagonista, els enemics i els entorns del joc relacionats amb el riu Ebre.
4. Integrar educació i entreteniment amb la sensibilització sobre les espècies invasores.

Amb aquests objectius, hem buscat no només completar el projecte, sinó també adquirir coneixements que puguin ser aplicables en futurs estudis.

1.3 Estat de la qüestió

Actualment, el programa "Unity" és una de les eines més populars en el desenvolupament de videojocs, especialment per a projectes en dues dimensions. Amb la seva facilitat d'ús, "Unity" ha estat utilitzat tant per grans empreses com per desenvolupadors independents. A més, el llenguatge de programació "C#", integrat a "Unity", proporciona una gran base per crear mecàniques complexes i personalitzades de manera intuitiva.

Pel que fa a videojocs amb una temàtica ambiental, cada vegada hi ha més projectes que busquen sensibilitzar els jugadors sobre problemes ecològics. Alguns exemples són videojocs que tracten sobre la conservació de recursos naturals o la protecció del medi ambient. Tot i això, la majoria d'aquests jocs tenen abast global i no estan centrats en regions concretes.

En el cas del riu Ebre, no hem trobat videojocs que representin específicament aquesta zona i, encara menys, que estiguin orientats a tractar problemes com les espècies invasores. Aquest buit en els videojocs educatius ens obre una oportunitat única per desenvolupar un projecte que combini els següents aspectes:

- L'educació ambiental, amb una sensibilització sobre les espècies invasores com el silur o el cargol poma.
- La difusió del patrimoni cultural i natural del riu Ebre i les Terres de l'Ebre.
- L'entreteniment, mitjançant una jugabilitat inspirada en els jocs clàssics de plataformes.

Aquesta manca de videojocs tematitzats en el riu Ebre, especialment dissenyats amb un enfocament educatiu, justifica la nostra aposta per desenvolupar un projecte que ompli aquest buit i que alhora utilitzi eines accessibles com "Unity" i "C#".

1.4 Hipòtesi inicial

És possible crear un videojoc en dues dimensions on es representi un pescador combatent espècies invasores del riu Ebre amb el motor gràfic "Unity" utilitzant el llenguatge de programació "C#".

1.5 Metodologia emprada

Per a la realització d'aquest treball de recerca, hem seguit una metodologia estructurada en diverses fases, cadascuna amb objectius específics. En la primera fase, hem realitzat una investigació per estudiar les característiques tècniques de Unity, un motor gràfic especialitzat

en el desenvolupament de videojocs en dues dimensions. També hem investigat les particularitats del llenguatge de programació C#, ja que és l'eina principal utilitzada per programar el joc.

En la fase de disseny inicial, hem elaborat un esquema del joc, definint el protagonista i els enemics amb què es trobarà durant la seva aventura. A continuació, hem passat al disseny gràfic, on hem dissenyat els personatges, escenaris i enemics utilitzant el programa "Piskel", especialitzat en disseny gràfic.

Finalment, en la fase de programació i desenvolupament, hem implementat les mecàniques de joc utilitzant "C#" dins de "Unity". En aquesta fase hi havia inclosa la creació d'interaccions entre el personatge i els enemics, la gestió de la vida del protagonista i la configuració de les condicions de victòria i derrota, així com la música i els efectes de so.

Aquest enfocament ens ha permès elaborar el projecte de manera organitzada, garantint que cada objectiu inicial es compleixi i que el resultat final concordi amb les nostres expectatives inicials.

2 MARC TEÒRIC

2.1 Definició i característiques principals dels videojocs

Un videojoc és un joc electrònic que es desenvolupa en un entorn virtual, en el qual el jugador interactua mitjançant un dispositiu electrònic amb pantalla i diversos perifèrics. Encara que hi ha qui els considera una forma d'art, aquest punt és debatut, tot i que cada vegada són més reconeguts com a part de la cultura moderna.

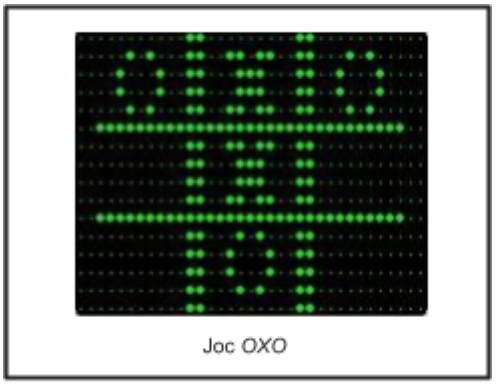
Els dispositius electrònics utilitzats per jugar a videojocs, coneguts com a plataformes, inclouen des dels ordinadors fins a aparells dissenyats específicament per a aquest propòsit, com les videoconsoles o les màquines recreatives, així com telèfons mòbils i altres gadgets[7] electrònics. Pel que fa al programari, els videojocs poden estar emmagatzemats en mitjans físics com cartutxos, targetes, disquets o CD, o bé descarregats de manera digital a través d'internet. Els videojocs poden ser dissenyats per a un únic jugador o per a múltiples jugadors i poden ser jugats localment o en línia.

[7] Són dispositius que han estat creats amb un propòsit i una funció. Solen ser de petites proporcions, molt pràctics i gairebé sempre presenten una novetat.

2.2 Evolució històrica dels videojocs

2.2.1 1952: aparició dels primers ordinadors elèctrics

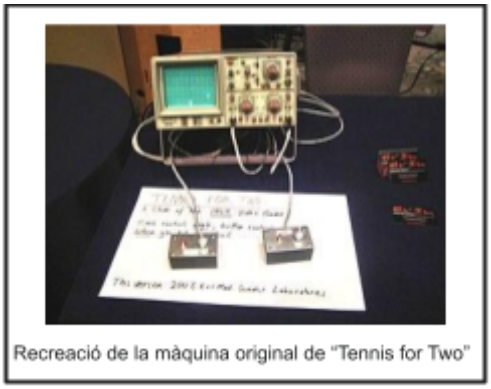
Poc després de l'aparició dels primers ordinadors electrònics, van començar els primers intents de crear programes que permetessin jugar amb aquestes màquines. Tot i això, els primers jocs estaven pensats exclusivament per a experiments científics i no estaven a l'abast del públic general; de fet, cap nen no els va poder provar. Entre aquests primers experiments destaca l'"OXO", una versió molt senzilla del tradicional joc del tres en ratlla, que va ser creada l'any 1952 per Alexander S. Douglas, un estudiant d'informàtica de la Universitat de Cambridge. Aquest joc va ser tota una fita perquè per primera vegada un ésser humà es podia enfrontar a una màquina en una partida. Per funcionar, necessitava un ordinador enorme, l'"EDSAC", que ocupava tota una sala i era molt limitat en comparació amb els ordinadors actuals. A més, cal destacar que aquests primers jocs no tenien cap finalitat comercial, sinó que formaven part d'investigacions acadèmiques.



Joc *OXO*

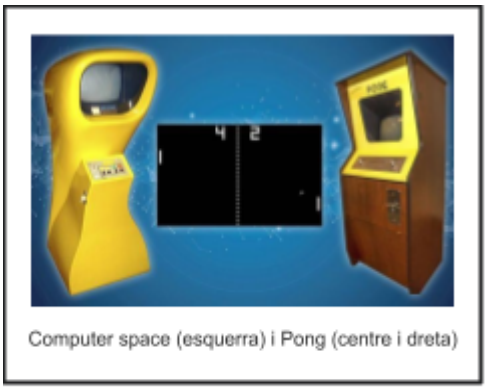
2.2.2 1958: creació del videojoc "Tennis Two"

Un científic dels Estats Units que treballava en un laboratori militar d'armes atòmiques va tenir una idea una mica peculiar. Va adquirir un ordinador capacitat per determinar les rutes de míssils i va crear un joc de tennis. "Tennis for two" mostrava una visió lateral de la pista, amb una línia vertical al seu centre que representava la xarxa. Els dos jugadors havien de colpejar la pilota amb un polsador i assignar-li una direcció amb un comandament força bàsic. Es va presentar al públic en dues ocasions, i centenars d'individus van passar hores esperant per poder jugar una partida.


Recreació de la màquina original de "Tennis for Two"

2.2.3 1971: primeres màquines electròniques comercials

Tots aquests videojocs funcionaven en superordinadors altament costosos i de gran mida. Va ser precisament un enginyer captivat per Spacewar qui va aconseguir desenvolupar la primera màquina electrònica d'ús comercial. S'anomenava "Computer space", funcionava amb monedes i permetia jugar en bars i altres llocs. No va triomfar gaire, però l'any següent va crear "Pong", que va ser tot un èxit. En aquesta simulació de tennis de taula, el participant es topa amb la màquina o amb un altre ésser humà, provocant que la pilota reboti amb la seva pala fins que un dels dos falla el cop. Molt senzill d'aprendre, però extremadament complicat de dominar. Es considera que aquest joc va ser el començament de la indústria dels videojocs.


Computer space (esquerra) i Pong (centre i dreta)

2.2.4 Els 70: aparició les primeres consoles domèstiques

Captura del joc *Space Invaders*

Les primeres consoles domèstiques van aparèixer poc després. Es podien connectar amb la televisió i oferien una gran varietat de jocs per triar. La "Odyssey" de 1972 va ser un èxit en vendes, i encara més la "Atari 2600", que va aparèixer el 1977. Se'n van vendre 25 milions d'unitats al llarg de catorze anys. Als salons recreatius, el joc més popular va ser "Space Invaders", del 1978, en què fileres i més fileres de naus extraterrestres baixen poc a poc cap a la nau del jugador, que les ha d'anar liquidant amb els trets del seu canó làser.

2.2.5 1980: Pac-Man

"Pac-Man" va ser creat pel dissenyador de videojocs japonès Toru Iwatani el 1980, mentre treballava per a la companyia japonesa "Namco". Iwatani estava buscant crear un joc que atrauria a tots els públics, i que fos fàcil de jugar i comprendre.

Els quatre fantasmes que persegueixen "Pac-Man" tenien noms i personalitats úniques. Blinky era el fantasma vermell i agressiu; "Pinky" era el fantasma rosa i tímid; "Inky" era el fantasma blau i tou; i "Clyde" era el fantasma taronja i lent.

El videojoc és considerat com un dels videojocs més influents de tots els temps, perquè va demostrar el potencial dels personatges de videojocs i a més, va ser el primer videojoc per oferir power-ups[6].

[8] És un objecte quan es utilitzat proporciona instantàniament al personatge del jugador característiques especials

2.2.6 Els 80: Nintendo i Sega ofereixen noves consoles

"Nintendo" i "Sega" van canviar el món dels videojocs amb les seves consoles domèstiques, que van aconseguir fer ombra al domini que fins aleshores tenia Atari. Al mateix temps, van aparèixer ordinadors personals com el "Spectrum", l'"Amstrad CPC", el "Commodore 64" i l'"MSX", que eren més econòmics i potents. Això va permetre que moltes famílies poguessin tenir accés a aquesta tecnologia, fet que va donar lloc a una etapa que es coneix com l'edat d'or dels videojocs. Durant aquesta època van sorgir molts dels gèneres que encara avui són molt populars, com els jocs de conducció, lluita, plataformes, estratègia o aventura.

Nintendo (representat per Mario) i Sega (representat per Sonic)

També es van fer molt conegudes les primeres consoles de butxaca. Al principi només tenien un sol joc, però van captivar els jugadors per la seva simplicitat i perquè es podien portar a tot arreu. Al 1981 va aparèixer "Donkey Kong", un joc que ràpidament va triomfar arreu del món. Va ser el primer gran èxit de "Nintendo" fora del Japó i va presentar personatges que es convertirien en llegendaris, com el mateix "Mario", que avui dia és una icona del sector dels videojocs.

2.2.7 1985: "Super Mario Bros"

L'enemic de "Donkey Kong" era un fontaner anomenat "Mario", que havia de rescatar la seva estimada del goril·la. Aquell any va aparèixer en un videojoc propi, amb el seu germà "Luigi", i va esdevenir un dels personatges més estimats pels jugadors de totes les edats. "Super Mario Bros" és un joc de plataformes en què el jugador avança d'esquerra a dreta pels diferents espais del Regne del Bolet. Ha de recollir les monedes que va trobant, mentre supera tota mena d'obstacles i elimina els enemics que l'ataquen,

Pantalla d'inici de *Super Mario Bros*

saltant sobre els seus caps. D'aquest joc se'n van vendre 40 milions de còpies a tot el món. L'èxit va ser tan gran que "Super Mario" s'ha mantingut en totes les plataformes de Nintendo fins avui.

2.2.8 1989: Game Boy

Videoconsola portàtil Game Boy

La "Game Boy" va sortir a la venda per primera vegada al Japó el 21 d'abril de 1989. Va ser dissenyada pel mateix equip responsable de la sèrie "Game & Watch", els populars jocs electrònics portàtils de "Nintendo", així com de diversos títols per a la "Nintendo Entertainment System". Aquesta consola portàtil destacava per les seves característiques innovadores: una pantalla de matriu de punts amb un dial[9] per ajustar el contrast, cinc botons per controlar el joc, un únic altaveu amb control de volum, i l'ús de cartutxos intercanviables per carregar diferents jocs, una característica nova en el món de les consoles portàtils.

Un dels factors clau del seu èxit va ser el joc "Tetris", un títol senzill d'aprendre però difícil de dominar, que atrapava els jugadors amb la seva mecànica de col·locar peces i completar fileres per fer-les desaparèixer. Aquest joc es va convertir en un fenomen mundial i va ajudar a impulsar les vendes de la "Game Boy" fins a xifres rècord, arribant als 118 milions d'unitats venudes arreu del món. Amb aquesta combinació de portabilitat, innovació i un catàleg de jocs emblemàtics, la "Game Boy" va marcar un abans i un després en la història de les consoles de videojocs.

[9] Superfície graduada, de forma variable, sobre la qual es mou un indicador, ja sigui una agulla, un punt lluminós, etcètera, que mesura o assenyala una determinada magnitud, com pes, voltatge, longitud d'ona, velocitat, etcètera.

2.2.9 1994: PlayStation


Videoconsola PlayStation

Les consoles domèstiques van experimentar un important salt tecnològic a finals dels anys noranta i principis dels 2000, amb gràfics que milloraven de forma contínua i oferien una experiència de joc cada vegada més immersiva. Un punt clau en aquesta evolució va ser el llançament de la primera "PlayStation", desenvolupada per "Sony", que va arribar al mercat el 1994 al Japó i el 1995 a la resta del món. Aquesta consola va marcar un abans i un després gràcies a l'ús de "CD-ROM" en lloc dels tradicionals cartutxos, la qual cosa permetia una major capacitat de memòria i jocs amb més contingut i millors gràfics. La "PlayStation" va assolir un gran èxit comercial, amb més de 100 milions d'unitats venudes arreu del món.

L'any 2000, "Sony" va llançar la seva successora, la "PlayStation 2" (PS2), que es va convertir en la consola més venuda de la història amb més de 155 milions d'unitats venudes globalment. La "PS2" destacava no només pels seus avançats gràfics, sinó també per la compatibilitat amb els jocs de la primera "PlayStation" i per la capacitat de reproduir "DVD". No obstant això, la "PS2" va haver de competir amb rivals poderosos, com la "Xbox" de "Microsoft", que va introduir funcions innovadores com l'ús de disc dur per guardar partides, i la "GameCube" de Nintendo, coneguda pels seus jocs exclusius i el seu format de discs més compacte.

2.3 Importància dels videojocs en la societat actual

La indústria dels videojocs ha evolucionat fins convertir-se en un admirat tant pel seu poder tecnològic com per la seva visió empresarial. Amb ingressos globals que van superar els 180.000 milions de dòlars el 2022 i amb expectatives d'arribar als 200.000 milions el 2025, els videojocs dominen l'oci

audiovisual, superant la música i el cinema. Aquest sector ha estat capaç d'atreure més de 3.200 milions de jugadors a tot el món i ha aconseguit èxits massius com "Fortnite", que compta amb més de 230 milions d'usuaris actius mensuals. Amb aquests números, els videojocs han plantejat un desafiament fins i tot a gegants de l'streaming com "Netflix", que ja es veu més amenaçat per aquesta indústria que pels seus competidors directes com "Disney" o "HBO".

Ademés de la seva importància educativa, els videojocs proporcionen una sèrie d'avantatges demostrats per la ciència. Els estudis evidencien, que l'ús de videojocs pot potenciar l'habilitat per respondre i l'agilitat en la presa de decisions. Una investigació realitzada per la Universitat de Rochester va demostrar que els videojocs potencien l'habilitat dels jugadors per manejar situacions inesperades, mentre que l'Institut per al Futur de Califòrnia subratlla que els jocs de "multiplayer[10]" potencien les capacitats de col·laboració. A més, fomenten la creativitat, la concentració i la memòria visual, i potencien l' habilitat de lideratge i estratègia en situar els jugadors en contextos de control.

En termes d'aprenentatge d'idiomes, els videojocs son una excel·lent eina, ja que permeten interactuar amb altres jugadors de diferents parts del món a través de xats i narratives. Fins i tot fomenten el pensament crític, en proposar dilemes ètics i socials que fan reflexionar els jugadors.

No obstant, l'efecte dels videojocs no és completament beneficiós. El 2019, l'Organització Mundial de la Salut va afegir a la Classificació Internacional de Malalties l'addicció als videojocs, cosa que ha provocat preocupació en nombrosos pares. És crucial que aquests estiguin alerta als suggeriments dels sistemes de categorització de videojocs com "PEGI" o "ESRB", que contribueixen a guiar sobre l'apropiació dels jocs d'acord amb l'edat. A més, se suggereix establir hores específiques amb límit de joc.

En resum, els videojocs, més enllà de ser una forma d'entreteniment, tenen un impacte positiu en diverses àrees de la nostra vida, des de l'educació fins al desenvolupament cognitiu i social, sempre que s'utilitzin de manera equilibrada i responsable.

[10] Joc jugat per diversos jugadors.

2.4 Espècies invasores i l'impacte ambiental al riu Ebre

Al nostre videojoc hem incorporat fins a tres tipus diferents d'espècies invasores que es poden trobar al riu Ebre, cadascuna amb les seves pròpies característiques i impactes en l'ecosistema.

El silur, un peix que no és originari de les nostres terres, va ser introduït al riu Ebre l'any 1974 per un pescador alemany. Aquest pescador va decidir alliberar milers d'alevins de silur al riu perquè estava entusiasmat amb la idea de fer captures en un futur. Tot i que al principi no s'hi va donar molta importància, aquesta acció va tenir un gran impacte en l'ecosistema, ja que el silur és un depredador molt potent que ha alterat l'equilibri natural de la fauna autòctona. Amb el pas del temps, la seva presència s'ha convertit en un problema ambiental significatiu per al riu, provocant també l'aparició de la mosca negra.

La mosca negra, que podria haver sorgit de manera indirecta pel fet que el silur depreda espècies locals i altera l'equilibri ecològic. Això hauria pogut influir en les poblacions d'insectes i modificar les cadenes alimentàries. La femella d'aquest insecte busca sang amb la qual alimentar els seus ous i per a això, les seves mandíbules tallen la pell, provocant que la seva picada sigui en realitat una mossegada. Aquesta mossegada produeix fortes reaccions al·lèrgiques i símptomes com altes febres que poden arribar a durar setmanes.

Per últim, el cargol poma és considerada una de les 100 espècies invasores més perjudicials del món. Aquest ha causat danys importants als ecosistemes naturals de les zones que ha colonitzat, especialment en els arrossars i altres cultius de zones humides.

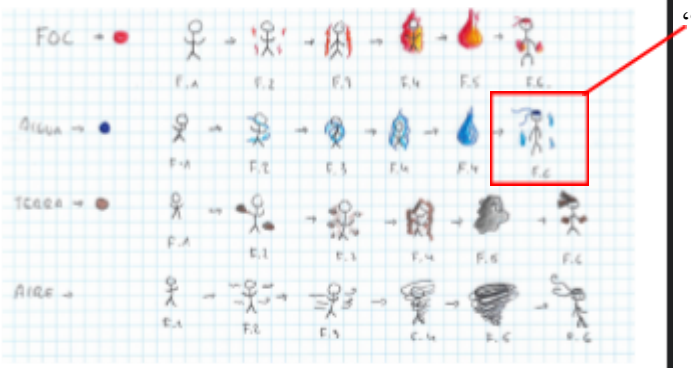
3 DESENVOLUPAMENT DEL PROJECTE

3.1 Elecció del tema

3.1.1 Argument i temàtica

La temàtica per la qual vam optar al principi del treball va ser un personatge principal, un ninja, que pogués transformar-se en els quatre elements principals de la natura (terra, foc, aigua, planta) mitjançant un sistema de bonus[11]. La idea era que el color del personatge fos neutre per fer notable el canvi de colors entre transformacions, llavors el personatge passaria de gris a: roig, blau, marró o blanc depenent de l'element. També, un color no molt vistós del personatge ens permet destacar els colors verds de l'escenari ambientat en una jungla.

Vam començar a fer esbossos a mà de com es transformaria el personatge:



"F" de frames[12]

Esbossos inicials fets a mà.

[11] També anomenat potenciador, és un element o objecte que proporciona un efecte positiu al personatge.
[12] De l'anglès "frame", traducció del qual és: quadre. Cadascuna de les imatges estàtiques que formen part de la successió d'imatges que componen una animació, i que a la vista produeixen sensació de moviment.

Al dibuix es veuen les diferents fases per les quals passa el personatge, en aquestes es veu com poc a poc va incrementant el nivell de partícules, donant així una sensació de força al personatge.

Vam decidir desenvolupar el disseny del personatge i l'entorn utilitzant l'aplicació Piskel, que ens permetia treballar amb píxels per a gràfics en dues dimensions.

Tot i la idea inicial del ninja i els quatre elements, vam veure que no era viable per diversos motius com la dificultat de contingut, ja que requeria un volum de treball massa gran per a un equip tant reduït com el nostre. Un altre motiu va ser l'originalitat, degut a que després d'investigar més, vam trobar que ja existien jocs similars, fet que reduïa la originalitat del nostre projecte.

Davant d'aquestes conseqüències, vam decidir buscar una temàtica més propera per a nosaltres. Va ser llavors quan ens vam plantejar fer un joc ambientat en el riu Ebre, un símbol molt representatiu del territori. Aquesta temàtica ens va semblar més atractiva per la proximitat que tenim al riu Ebre i perquè no hi havia videojocs que valoressin aquesta zona. A més, volíem aprofitar el joc per conscienciar sobre el problema de les espècies invasores que estan afectant l'ecosistema. Finalment, vam concloure que un joc basat amb una mecànica més senzilla era més factible per al nostre nivell i recursos disponibles.

Fotografia del riu Ebre

Aquesta decisió ens va permetre desenvolupar un projecte més relacionat amb el nostre territori amb un missatge que considerem important.

3.1.2 Objectius del joc

El videojoc s'ha desenvolupat amb l'objectiu de combinar diversió i aprenentatge, proporcionant una experiència que dóna l'oportunitat de gaudir mentre involuntàriament s'aprèn més sobre el riu Ebre i la fauna que l'envolta. A continuació, farem una explicació més extensa d'aquests propòsits o objectius.

El primer propòsit és entretenir. El joc s'ha dissenyat per ser molt visual, senzill d'utilitzar i divertit, amb l'objectiu de captar l'atenció dels jugadors des del principi. Això és fonamental perquè les persones vulguin jugar-hi i, alhora, estiguin disposades a aprendre nous coneixements mentre s'ho passen bé.

El segon element fonamental és l'ensenyament. A través del joc, els jugadors poden adquirir coneixements sobre el riu Ebre, la seva flora i fauna, així com els desafiaments que l'afecten, com l'existència d'espècies invasores. En lloc de proporcionar informació de manera monòtona, el joc ho fa de manera interactiva i dinàmica, amb l'objectiu que l'aprenentatge esdevingui una part natural de l'experiència.

En conclusió, aquest videojoc té com a objectiu anar més enllà dels límits d'un simple joc. És una eina per connectar amb el patrimoni natural del riu Ebre, conèixer-ne la riquesa i aprendre a valorar-lo.

3.2 Programes i eines utilitzades

3.2.1 Unity

"Unity" és un motor gràfic especialitzat en el disseny de videojocs que permet crear aplicacions en dues i tres dimensions. És compatible amb més de 25 plataformes, incloent dispositius mòbils, ordinadors de sobretaula, consoles i televisors.

El programa és gratuït i està disponible per a "Windows" i "macOS", cosa que el fa accessible tant per a principiants com per a professionals. Actualment, més del 50 % dels videojocs publicats es desenvolupen amb "Unity", fet que reflecteix la seva popularitat en la indústria. Entre ells, destaquen els videojocs: "Pokémon Go", "Among Us" o "Cuphead".

Un dels seus punts forts és la comunitat activa que hi ha al voltant del programa. La seva "Asset Store[13]" ofereix una manera de comprar i vendre recursos que permeten accelerar el procés de desenvolupament. Això facilita el treball, especialment per als equips petits o per als desenvolupadors que comencen, però nosaltres no l'hem utilitzat.

A més, "Visual Studio" proporciona eines avançades d'edició i depuració que es poden integrar perfectament amb "Unity", oferint un entorn de treball eficient per a programadors.

3.2.2 Visual Studio Code

"Visual Studio Code", també conegut com a "VS Code", és un editor de codi gratuït desenvolupat per "Microsoft". Aquest programa s'enfoca en oferir agilitat en la escriptura i l'edició de codi, combinat amb característiques avançades com:

- Autocompletat intel·ligent.
- Suport per a diversos llenguatges de programació, incloent "C#".

[13] És una plataforma integrada dins de "Unity" que permet als desenvolupadors comprar, vendre i compartir recursos per als seus projectes. Aquests recursos, coneguts com assets, inclouen una àmplia gamma d'elements, com ara: models en 2D i 3D, àudios, animacions o scripts.

- "Debugger[14]" integrat per detectar i solucionar errors de codi.

Gràcies a la seva integració amb "Unity", "VS Code" es converteix en una opció ideal per gestionar i escriure el codi del nostre projecte.

3.2.3 Piskel

"Piskel" és una aplicació gratuïta i de codi obert per crear i editar gràfics basats en píxels. És una eina molt útil per dissenyar elements visuals senzills, com personatges, escenaris o objectes en dues dimensions.

Entre les seves característiques principals destaquen:

- Interfície intuïtiva, ideal per a principiants.
- Opcions per animar sprites, permetent visualitzar les transicions en temps real.

Hem utilitzat "Piskel" per dissenyar personatges i escenaris del nostre videojoc, creant gràfics únics que reflecteixen l'estètica del projecte.

3.2.4 Audacity

"Audacity" és un editor d'àudio gratuït i de codi obert que permet gravar, editar i processar fitxers d'àudio. És molt utilitzat en el desenvolupament de videojocs per a tasques com:

- Creació i edició d'efectes de so.
- Gravació de veu o música.
- Millora de la qualitat d'àudio amb filtres i eines de reducció de soroll.

En el nostre projecte, hem utilitzat "Audacity" per produir els efectes sonors i editar les pistes musicals que acompanyen l'experiència de joc.

[14] És una eina de programació que permet identificar, analitzar i corregir errors (bugs) en el codi d'un programa.

3.3 Procés de creació

3.3.1 Disseny dels personatges i escenaris

El procés de disseny dels personatges i escenaris va començar amb la representació de les idees principals. Inicialment, es van realitzar esbossos manuals dels personatges, els enemics i els elements de l'entorn, tenint en compte la temàtica del riu Ebre i les espècies invasores. Un cop definides les bases, es va utilitzar l'eina "Piskel" per digitalitzar aquests dissenys.

Aquesta aplicació ens va permetre treballar amb gràfics en píxels, que encaixaven a la perfecció amb la idea inicial que teníem. Els personatges i escenaris es van crear amb un estil senzill però funcional, de manera que fossin fàcilment reconeixibles i representatius del riu.
Els escenaris es van dissenyar de manera que reflectissin els diferents trams del riu Ebre, des de la presa de Riba-roja fins a la desembocadura al Delta, amb un seguit dels símbols més representatius de cada poble situat al llarg del riu.

Vam iniciar amb el disseny del terreny, ja que ens proporcionava una base ferma per determinar la varietat de colors que utilitzaríem en el nostre videojoc. Aquest es segmenta en quatre tonalitats fonamentals: el verd, que simbolitza la gespa; el marró, que simbolitza el terra; el negre, emprat per definir les vores, i el lila, que fem servir per generar ombres i proporcionar profunditat al terra.

Després vam procedir a donar-li forma establint els dissenys dels escenaris. Vam utilitzar mesures molt més grans i per tant vam haver de dibuixar molts més pixels. Primerament, vam treballar en les muntanyes, seguint amb el cel i, finalment, representant els elements més

emblemàtics de cada poble. Per estar situats en tot moment, vam dissenyar uns cartells de fusta que indiquen la l'ubicació del pescador.

Per dibuixar el personatge principal, el pescador, vam optar per un model minimalista on els colors no fossin protagonistes per donar més èmfasi a l'entorn del videojoc. Les animacions es van crear redibuixant una a una les parts del pescador en diferents posicions per aconseguir l'efecte de moviment. El cavaller és una versió redissenyada del pescador, però el vam modificar per aconseguir l'aparença d'un templer[15], inspirat en els templers que van ocupar el castell de Miravet durant els segles XII i XIII. Aquest, porta una espasa sobre l'espatlla que tindrà com a funcionalitat atacar al pescador.

Per les espècies invasores, vam optar per dissenys fàcils d'animar ja que havíem invertit molt temps dissenyant els personatges. Totes les espècies invasores estan dibuixades amb realisme, menys el cargol poma que conté una poma sobre l'esquena en homenatge al seu nom.

Porc senglar del joc comparat amb un de real

Els "sprites" utilitzats en el videojoc, incloent personatges, enemics i objectes decoratius, es troben detallats a la segona part dels annexos, anomenada *Disseny dels sprites del joc*, on es poden consultar tant els dissenys finals com algunes de les fases intermèdies del procés de creació.

[15] De l'ordre de cavalleria del Temple, institució religiosa i militar fundada al segle XII.

3.3.2 Programació de les mecàniques del joc

La programació de les mecàniques del joc la vam dur a terme mitjançant "Unity" i el llenguatge "C#". Aquest procés va ser essencial per donar vida al projecte, ja que vam treballar en donar moviment al personatge principal i la interacció d'aquest amb els enemics i objectes.

El moviment del personatge el vam basar en la física en dues dimensions de "Unity", la qual ofereix un comportament realista. Això inclou diverses funcions específiques per a accions com saltar o llençar pedres.

Els enemics són majoritàriament espècies invasores, i els vam programar amb diferents comportaments depenent de si eren terrestres, que es desplacen en una direcció, o aeris, que es mouen en dues dimensions perseguint al pescador..

A més, vam implementar un sistema de vida per al jugador i els enemics. El pescador comença amb cinc vides, i cada cop que col·lisiona amb un enemic, perd una vida. Per la seva banda, els enemics també tenen punts de vida assignats i perden vida cada vegada que són impactats per una pedra llençada pel personatge principal.

Els efectes de so i la música van ser primordials en la creació de l'ambient. Els passos del personatge els vam sincronitzar amb un efecte de so. Els enemics compten amb efectes sonors propis, com els crits de les cigonyes, el vol de la mosca negra o els sons dels porcs senglars.

També vam afegir música de fons en bucle, que acompanya el jugador durant tota la partida acompanyats de sons ambientals com l'aigua del riu i el cant dels moixons. El volum de cada element sonor el vam ajustar per garantir que tots els efectes es poguessin escoltar perfectament sense sobreposar-se a la música.

Finalment, tots els fitxers d'àudio els vam integrar a "Unity" i es gestionen de manera dinàmica segons les accions del jugador i el context del joc.

Els "scripts" que donen vida a aquestes mecàniques es troben a la secció d'annexos, anomenada *Scripts de codi comentats*, on s'expliquen els fragments del codi utilitzats. Aquesta informació proporciona una visió tècnica del desenvolupament del videojoc.

3.3.3 Descripció de l'experiència de joc

El joc comença a Riba-Roja d'Ebre, el primer poble de Catalunya per on passa el riu Ebre. El pescador apareix a dalt de la presa que es troba en aquesta localitat. Si avances cap a la dreta i caus per la presa et trobes amb Flix, on pots observar el parc natural de sebes. Però per desgràcia topes amb un enemic, que és la cigonya. Aquesta et persegueix per matar-te però pots contraatacar llençant-li pedres. A continuació, arribes a Ascó amb l'emblemàtica Central Nuclear, la font d'energia elèctrica més gran de les Terres de l'Ebre. A més, és el lloc de treball de més de mil persones. Si segueixes avançant, passes per Móra la Nova amb un tren, que fa referència al que va arribar a ser l'antiga estació de trens que donava treball a molts habitants. Però abans d'arribar a aquest tren hi ha un enemic molt comú a la zona, el porc senglar. Aquest també t'intenta atacar però pots contrarestar aquest tirant-li pedres. Seguidament, creues el pont d'arcades que uneix Móra la Nova amb Móra d'Ebre. Però ves amb compte a l'hora de creuar-lo, doncs hi ha tres silurs saltant que intenten atacar-te. Si pots sortir amb vida, la següent població a la que arribes és Miravet amb el seu mític castell amb vistes al riu. Al castell t'espera un templer molt ben armat amb la intenció de no deixar que ningú passi a prop seu. Si el cavaller rep deu pedrades, pots avançar fins a Benifallet, on pots entrar a les Coves Meravelles i lluitar contra una mosca negra. Quan surts de la cova, pots

observar els tarongers que son tant típics a Xerta. Seguidament, arribes a la capital de les Terres de l'Ebre, Tortosa, i de fons apareix el mural amb el text: «LO RIU ÉS VIDA». Finalment arribes a Amposta, on pots menjar una carxofa típica de la ciutat i curar la vida al màxim. Però després has de lluitar amb el cargol poma, l'enemic amb més vida del joc, tot i que té una velocitat molt reduïda. El joc s'acaba al mar, si toques l'aigua apareix la pantalla de victòria i comença a sonar una sardana molt coneguda.

En la segona part dels annexos, anomenada *Disseny dels sprites del joc*, es poden trobar captures de pantalla on s'observen diferents zones per on passa el pescador, la lluita amb el cavaller templer i les pantalles de menú inicial, de pausa, de victòria i de derrota.

3.3.4 Manual d'instruccions per a l'usuari

En aquest apartat presentem una guia amb els controls bàsics per a jugar al videojoc:

Avançar:

Tecla D: El personatge es mourà cap endavant.

Alternativa: → (fletxa dreta).

Retrocedir:

Tecla A: El personatge es mourà cap enrere.

Alternativa: ← (fletxa esquerra).

Saltar:

Tecla Espai: El personatge saltarà.

Nota: El salt només és possible si el personatge està a terra.

Disparar:

Botó esquerre del ratolí: El pescador llança una pedra per atacar els enemics.

Nota: El personatge tarda una estona en disparar des que premeu el botó.

Pausa:

Tecla Esc (Escape): El joc es pausa, permetent-te tornar a començar o sortir del joc.

Consells:

Eviteu els enemics: Els enemics poden causar-vos dany si colisionen amb el vostre personatge. Intenteu eliminar-los amb les vostres pedres.

Recupereu la vida: Les carxofes que estan distribuïdes pel joc us recuperen la vida al màxim.

4 RESULTATS

El videojoc desenvolupat representa una aventura en dues dimensions que combina la diversió amb l'educació del medi ambient. El protagonista és un pescador que es desplaça per diverses àrees del riu Ebre a Catalunya, des de la presa de Riba-roja fins al Delta, per eliminar espècies invasores com la mosca negra, els cargols poma i els silurs.

Tots els llocs són simbòlics del riu Ebre, amb components i construccions del paisatge que l'envolten, com la central nuclear d'Ascó o el pont que connecta Móra la Nova amb Móra d'Ebre. Les tècniques que hem posat en pràctica són bàsicament el moviment del pescador, el salt, el llançament de pedres i la interacció amb adversaris, amb un sistema de vides.

A més, el joc inclou música de fons, tret del moment en què s'enfronta al cavaller. A més, inclou sons ambientals com l'aigua i els ocells, juntament amb efectes sonors com el de caminar del pescador i els sorolls dels enemics, tot això amb la finalitat de transportar el jugador a un entorn que recordi el riu.

4.1 Revisió del compliment dels objectius inicials

Els objectius específics del projecte tenien com a finalitat donar-nos coneixements i habilitats relacionades amb el disseny i la programació del videojoc. Aquests objectius eren essencials per al desenvolupament del joc i per aconseguir que el projecte fos complet i funcional. A continuació, analitzem si hem aconseguit resoldre cada un dels objectius específics establerts al principi del treball:

- Aprendre a utilitzar el motor gràfic "Unity" per crear videojocs en dues dimensions: Hem aconseguit dominar les eines bàsiques de "Unity", incloent la creació de personatges, escenaris i la implementació de funcions del joc en dues dimensions.

- Dominar els conceptes bàsics del llenguatge de programació "C#": Hem assolit un coneixement suficient del llenguatge "C#" per implementar les funcions necessàries al joc, com els controls del personatge, la interacció amb els enemics i la gestió del sistema de vides.
- Dissenyar personatges i escenaris com el protagonista, els enemics i els entorns del joc relacionats amb el riu Ebre: El disseny dels personatges i escenaris es va resoldre amb èxit amb l'aplicació "Piskel", creant entorns que representen diferents trams del riu Ebre i integrant els personatges que reflecteixen les espècies autòctones i invasores del riu.
- Integrar educació i entreteniment amb l'exposició sobre problemes mediambientals, com les espècies invasores: El projecte ha aconseguit integrar un component educatiu, ja que el joc inclou informació sobre les espècies invasores del riu Ebre, conscienciant el jugador mentre es diverteix.

D'aquesta manera, el nostre treball ha complert els objectius que ens havíem proposat, integrant els aspectes tècnics, creatius i educatius en un videojoc funcional i amb una temàtica significativa.

5 CONCLUSIONS

5.1 Valoració global del projecte

El projecte ha complert els objectius marcats inicialment i ha validat la hipòtesi plantejada: És possible crear un videojoc en dues dimensions on es representi un pescador combatent espècies invasores del riu Ebre amb el motor gràfic "Unity" utilitzant el llenguatge de programació "C#".

Els resultats que hem obtingut han estat satisfactoris, ja que el joc combina els sprites perfectament amb els scripts i, alhora, amb efectes sonors. El disseny dels escenaris reflecteix la realitat del riu Ebre, i les mecàniques de joc, tot i que puguin semblar senzilles, permeten una combinació de competitivitat amb diversió.

A més, el joc integra continguts educatius, contribuint a la sensibilització que pateix l'ambient d'una manera lúdica. Això valida l'enfocament inicial del projecte, on vam afirmar que podiem combinar un videojoc amb l'ensenyament.

5.2 Aprenentatges adquirits

Al llarg del projecte, hem après a utilitzar eines com "Unity" i "C#" per programar les mecàniques del joc, així com "Piskel" per dissenyar els gràfics en píxels. També hem après a integrar música i efectes de so d'una manera que millora l'experiència de joc i reforça la temàtica del projecte.

D'altra banda, hem après a treballar de manera organitzada i a repartir les tasques dins de l'equip, així com a resoldre problemes tècnics que han sorgit durant el desenvolupament. A més, hem aprofundit en el coneixement dels problemes ambientals del riu Ebre, cosa que ha estat una experiència enriquidora tant a nivell personal com acadèmic.

5.3 Validació de la hipòtesi

La hipòtesi plantejada al començament del projecte afirmava que era possible crear un videojoc en dues dimensions que representés un pescador combatent espècies invasores del riu Ebre utilitzant el motor gràfic "Unity" i el llenguatge de programació "C#". Després de completar el procés de creació del videojoc, podem concloure que la hipòtesi és totalment vàlida i ha estat resolta de manera satisfactòria.

Durant el desenvolupament, vam aconseguir construir un videojoc utilitzant les eines proposades. El programa "Unity" ens va ser molt útil per la creació d'aquest videojoc en dues dimensions, gràcies a les seves eines de física i animació, que van permetre una programació molt intuitiva. A més, el llenguatge de programació "C#" va ser fonamental per implementar les mecàniques del joc, com el moviment del personatge, la lluita amb els enemics i la gestió del sistema de vida, que eren elements fonamentals per al funcionament del videojoc.

Pel que fa a la temàtica del joc, vam aconseguir representar un pescador en el riu Ebre, amb enemics que representen espècies invasores. Les mecàniques de joc, com el combat amb els enemics, van ser dissenyades per educar mentre es manté un constant desafiament. Els escenaris i personatges creats també reflecteixen la fauna típica d'aquest entorn.

En resum, la hipòtesi s'ha confirmat correctament, i el resultat final del videojoc compleix les expectatives plantejades inicialment, creant un joc entretingut que no només representa un

pescador lluitant contra espècies invasores, sinó que també apropa els jugadors al coneixement dels problemes mediambientals del riu Ebre.

6 FONTS CONSULTADES

PÀGINES WEB CONSULTADES

GENCAT: El cargol poma (03 de desembre de 2024)
https://agricultura.gencat.cat/ca/ambits/agricultura/dar_sanitat_vegetal_nou/dar_plagues_males_herbes/dar_plagues/moluscs/cargol-poma/

FANDOM: *Pac-Man* (22 de novembre de 2024).
https://pacman.fandom.com/es/wiki/Pac-Man_(videojuego)#Impacto_y_legado

IMAGINA RÀDIO: Les "espècies invasores" amenacen l'ecosistema de les Terres de l'Ebre. (13 de novembre de 2024).
https://imaginaradio.cat/les-especies-invasores-amenacen-lecosistema-de-les-terres-de-lebre/

MIARCADE: *Space Invaders* (20 de novembre de 2024).
https://miarcade.com/space-invaders-arcade/

NATIONAL GEOGRAPHIC: Siluro, el imparable pez de 200 kg que pone en riesgo los ecosistemas de los ríos de España (2 de desembre de 2024).
https://www.nationalgeographic.com.es/mundo-animal/

NATIONAL GEOGRAPHIC: La plaga de la mosca negra invade los ríos españoles
https://www.nationalgeographic.es/medio-ambiente/2019/08/la-plaga-de-la-mosca-negra-invade-los-rios-espanoles

RAE: Diccionario de la lengua española. (01/12/2024).

https://www.rae.es/diccionario-estudiante/templario

PETIT SÀPIENS: Història dels videojocs. (13 de novembre de 2024).
https://www.petitsapiens.cat/un-viatge-al-passat/historia-videojocs_203149_102.html

VIQUIPÈDIA: Videojoc. (13 de novembre de 2024).
https://ca.wikipedia.org/wiki/Videojoc#

PROGRAMES UTILITZATS

AUDACITY TEAM: Audacity - Free, Open Source, Cross-Platform Audio Software. (23 de novembre de 2024).
https://www.audacityteam.org

MICROSOFT: Visual Studio Code. (23 de novembre de 2024).
https://code.visualstudio.com

PISKEL: Piskel - Free Online Sprite Editor (23 de novembre de 2024).
https://www.piskelapp.com

UNITY TECHNOLOGIES: Real-Time Development Platform. (23 de novembre de 2024).
https://unity.com

VÍDEOS OBSERVATS

Alva Majo. (2024, juliol 13). *Unity para retrasados* [Video]. YouTube. https://www.youtube.com/watch?v=IT7vDqm4xiY&t=101s

BravePixelG. (2024, setembre 12). *Cómo crear un fondo con animación en Unity (Efecto Parallax en Unity)* [Video]. YouTube. https://www.youtube.com/watch?v=7bJT6rf-Jvk

BravePixelG. (2024, setembre 8). *Cómo crear un menú inicial en Unity* [Video]. YouTube. https://www.youtube.com/watch?v=sJUBoFgO7Ng

Chris' Tutorials. (2024, setembre 16). *Learn How to Make a 2D Platformer in Unity 2022 - FULL GAMEDEV COURSE!*[Video]. YouTube. https://www.youtube.com/watch?v=oxiPWg8cdRM

JoexScript. (2024, setembre 29). *Unity 2D - Enemigo Básico 2D (Plataformas)* [Video]. YouTube. https://www.youtube.com/watch?v=aYbcHlpyBuo

JLPM. (2024, abril 21). *Tutorial COMPLETO Unity 2D desde Cero* [Video]. YouTube. https://www.youtube.com/watch?v=GbmRt0wydQU

KanarianDev. (2024, juliol 13). *¡ANIMA a TU PERSONAJE en UNITY! ✅* [Video]. YouTube. https://www.youtube.com/watch?v=kM-jcFYVY3s

FOTOGRAFIES UTILITZADES

ABC [Fotografia]. (n.d.). Jabalí, una especie oportunista https://www.abc.es/deportes/caza/abci-jabali-especie-oportunista-201907010115_noticia.html?ref=https%3A%2F%2Fwww.abc.es%2Fdeportes%2Fcaza%2Fabci-jabali-especie-oportunista-201907010115_noticia.html

Digital Trends en Español [Fotografia]. (n.d.). La guerra de consolas de los 90: Nintendo vs. Sega https://es.digitaltrends.com/videojuego/guerra-de-consolas-90-nintendo-vs-sega/

Evan-Amos [Fotografia]. (n.d.). A Sony PlayStation (SCPH-5001), shown with a DualShock controller and Memory Card. This is the PNG version. https://es.wikipedia.org/wiki/PlayStation_%28consola%29#/media/Archivo:PSX-Console-wController.png

Evan-Amos [Fotografia]. (n.d.). The Nintendo Game Boy, a handheld gaming console released in 1989. The relatively simple and monochrome Game Boy faced off against multiple handhelds and dominated the portable gaming market for almost a decade. This system has a 4.19 MHz CPU, a 4-shade LCD and ran off of 4 AA batteries. It was a very long-running system, only seeing a minor upgrade with the Game Boy Pocket in 1996, then later the Game Boy Color in 1998. https://es.wikipedia.org/wiki/Game_Boy#/media/Archivo:Game-Boy-FL.jpg

Foto del riu Ebre [Fotografia]. (n.d.). Consell Comarcal del Baix Ebre. https://www.baixebre.cat/arees-dactuacio/dinamitzacio-economica/turisme/el-riu-ebre

Marlith [Fotografia]. (n.d.). Picture of a Tennis for Two Machine at CAX 2010. https://es.wikipedia.org/wiki/Tennis_for_Two#/media/Archivo:Tennis_for_Two_Machine_at_CAX_2010.jpg

Miarcade [Fotografia]. (n.d.). Space Invaders. La historia del mítico videojuego arcade https://miarcade.com/space-invaders-arcade/

Nintendo [Fotografia]. (n.d.). Super Mario Bros. | NES | Juegos | Nintendo ES https://www.nintendo.com/es-es/Juegos/NES/Super-Mario-Bros-803853.html

Petit Sàpiens [Fotografia]. (n.d.). La història dels videojocs. https://www.petitsapiens.cat/un-viatge-al-passat/historia-videojocs_203149_102.html

7 ANNEXOS

7.1 Scripts de codi comentats

En aquest apartat dels annexos es presenta el codi font del videojoc, amb comentaris explicatius per a cada línia. Els comentaris es troben indicats després de dues barres "//" i tenen la finalitat d'explicar la funció de cada instrucció. Aquesta secció permet entendre com s'han implementat les mecàniques principals del joc mitjançant el llenguatge "C#" a "Unity".

PESCADOR

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PescadorMoviment : MonoBehaviour
{
    public float Speed; // Velocitat horitzontal del pescador.
    public float JumpForce; // Força de salt del pescador.
    public GameObject Pedra; // Referència al prefab de la pedra.

    private Rigidbody2D Rigidbody2D; // Component per gestionar la física del pescador.
    private Animator Animator; // Referència per controlar les animacions.
    private float Horizontal; // Moviment horitzontal.
    private bool Grounded; // Indica si el pescador està a terra.
    private bool Disparant; // Indica si el pescador està disparant.
    private float LastShoot; // Registra el temps de l'últim dispar.
```

```
    public float delayTime = 0.01f; // Temps d'espera.
    private float timer = 0f; // Temporitzador per controlar el temps d'espera.
    private bool isShooting = false; // Verifica si el pescador està disparant i comença en fals.
    private float Temps; // Temporitzador del temps de joc.
    private AudioSource caminar; // So de les passes del pescador.

    void Start()
    {
        Rigidbody2D = GetComponent<Rigidbody2D>(); // Assigna el component de cos rígid.
        Animator = GetComponent<Animator>(); // Assigna el component Animator.
        caminar = GetComponent<AudioSource>(); // Assigna el component d'àudio al caminar.
        Temps = 0; // Inicialitza el temporitzador en valor zero.
    }

    void Update()
    {
        if (Time.timeScale == 0f) return; // Si el joc està pausat, surt de la funció.

        Horizontal = Input.GetAxisRaw("Horizontal") * Speed; // Verifica cap a quin costat es mou i ajusta la velocitat.

        if (Horizontal < 0.0f) transform.localScale = new Vector3(-1.0f, 1.0f, 1.0f); // Gira el personatge a l'esquerra.
        else if (Horizontal > 0.0f) transform.localScale = new Vector3(1.0f, 1.0f, 1.0f); // Gira el personatge a la dreta.
```

```
        if (Horizontal != 0.0f && Grounded == true) // Si es mou i està a terra, reprodueix el so de caminar.
        {
            if (!caminar.isPlaying)
            {
                caminar.Play(); // Inicia el so de caminar.
            }
        }
        else
        {
            caminar.Stop(); // Para el so de caminar si no es mou.
        }

         Animator.SetBool("PescadorCaminant", Horizontal != 0.0f); // Actualitza l'animació de caminar.
         Animator.SetBool("Grounded", Grounded); // Actualitza l'animació segons si està a terra.
        Animator.SetBool("Disparant", Disparant); // Actualitza l'animació de disparar.

         Debug.DrawRay(transform.position, Vector3.down * 1f, Color.red); // Dibuixa una línia vermella per verificar si està tocant el terra.
         if (Physics2D.Raycast(transform.position, Vector3.down, 1f)) // Detecta si està a prop del terra.
        {
            Grounded = true;
        }
```

```
    else
    {
        Grounded = false;
    }

        if (Input.GetKeyDown(KeyCode.Space) && Grounded) // Si es prem la barra espaiadora i està a terra, salta.
    {
        Jump();
    }

    Temps += 1 * Time.deltaTime; // Incrementa el temporitzador amb el pas del temps.

     if (Input.GetMouseButtonDown(0) && !isShooting && Temps >= 0.3f) // Si es prem el botó esquerre del ratolí, dispara.
    {
        isShooting = true; // Indica que està disparant.
        timer = 0f; // Reseteja el temporitzador del dispar.
        Disparant = true; // Activa l'animació de disparar.
    }

    if (isShooting) // Si està disparant...
    {
        timer += Time.deltaTime; // Incrementa el temporitzador del retard.

        if (timer >= delayTime) // Quan el temps supera el retard definit, dispara.
        {
```

```
            Shoot();
            isShooting = false; // Indica que ja no està disparant.
            Disparant = false; // Desactiva l'animació de disparar.
            Temps = 0; // Reseteja el temporitzador del temps d'espera.
        }
    }
    else { Disparant = false; } // Si no està disparant, desactiva l'animació.
}

private void Shoot() // Funció per gestionar el dispar.
{
    Vector3 direction; // Direcció del dispar.
        if (transform.localScale.x == 1.0f) direction = Vector3.right; // Si mira a la dreta, dispara a la dreta.
    else direction = Vector3.left; // Si mira a l'esquerra, dispara a l'esquerra.

        Vector3 spawnPosition = transform.position + direction * 0.5f + Vector3.up * 0.3f; // Defineix la posició d'aparició de la pedra.

        GameObject pedra = Instantiate(Pedra, spawnPosition, Quaternion.identity); // Crea una pedra a la posició definida.
        pedra.GetComponent<PedraScript>().SetDirection(direction); // Estableix la direcció de la pedra.
}

private void FixedUpdate()
{
    Rigidbody2D.velocity = new Vector2(Horizontal, Rigidbody2D.velocity.y); // Ajusta la velocitat horitzontal.
```

```
    }

    private void Jump() // Funció per fer saltar el pescador.
    {
        Rigidbody2D.AddForce(Vector2.up * JumpForce); // Aplica una força cap amunt.
    }
}
```

CABALLER

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using TMPro; // Importa TextMeshPro per mostrar el text al videojoc.

public class CaballerScript : MonoBehaviour
{
    public Transform player; // Emmagatzemar la referència al jugador.
    public float detectionRadius = 8.0f; // Defineix el radi dins del qual el cavaller detecta el jugador.
    public float atacRadius = 2.0f; // Defineix el radi dins del qual el cavaller pot atacar el jugador.
    public float speed = 3.0f; // Estableix la velocitat de desplaçament del cavaller.
    public float cronometre; // Declara un cronòmetre per controlar el temps de moviment.
    private bool atac; // Declara una variable booleana per determinar si el cavaller està atacant.

   private float Horizontal; // Declara una variable per emmagatzemar la distància horitzontal entre el cavaller i el jugador.
```

```
    private Rigidbody2D rb; // Declara una variable per al component Rigidbody2D del cavaller, que controla la física.
    private Vector2 movement; // Declara una variable per emmagatzemar el moviment del cavaller com a vector 2D.
     public Animator Animator; // Declara una variable per a l'animador del cavaller, que controla les animacions.
     public CastellScript CastellScript; // Declara una variable per la referència al script del castell.
    private AudioSource caballer; // Declara una variable per al component AudioSource del cavaller, que controla el so.

   void Start() // Mètode que s'executa quan comença el joc o l'objecte es crea.
   {
         rb = GetComponent<Rigidbody2D>(); // Assigna el component Rigidbody2D del cavaller a la variable rb.
         caballer = GetComponent<AudioSource>(); // Assigna el component AudioSource del cavaller a la variable caballer.
   }

   void Update() // Mètode que s'executa cada fotograma del joc.
   {
         float distanceToPlayer = Vector2.Distance(transform.position, player.position); // Calcula la distància entre el cavaller i el jugador.
         cronometre += 1 * Time.deltaTime; // Incrementa el cronòmetre pel temps que ha passat des del darrer fotograma.
       if (distanceToPlayer < detectionRadius) // Si la distància entre el cavaller i el jugador és menor que el radi de detecció.
      {
         Animator.SetBool("atac", false); // Desactiva l'animació d'atac si el cavaller es mou cap al jugador.
```

```
        Vector2 direction = (player.position - transform.position).normalized; // Calcula la direcció normalitzada cap al jugador.

        movement = new Vector2(direction.x, 0); // Defineix el moviment només horitzontal (el cavaller no es mou verticalment).

        rb.MovePosition(rb.position + movement * speed * Time.deltaTime); // Mou el cavaller cap al jugador segons la direcció calculada.

        Horizontal = transform.position.x - player.transform.position.x; // Calcula la distància horitzontal entre el cavaller i el jugador.

        if (Horizontal > 0.0f) transform.localScale = new Vector3(-1.0f, 1.0f, 1.0f); // Si el cavaller està a la dreta del jugador, es gira per l'esquerra.
        else if (Horizontal < 0.0f) transform.localScale = new Vector3(1.0f, 1.0f, 1.0f); // Si el cavaller està a l'esquerra del jugador, es gira per la dreta.

        if (atacRadius > distanceToPlayer) // Si la distància entre el cavaller i el jugador és menor que el radi d'atac.
        {
            Animator.SetBool("atac", true); // Activa l'animació d'atac.
        }
    }
    else // Si el jugador no es troba dins del radi de detecció.
    {
        Animator.SetBool("atac", false); // Desactiva l'animació d'atac.

        if (cronometre >= 4) // Si el cronòmetre ha arribat als 4 segons.
        {
```

```
                transform.localScale = new Vector3(1.0f, 1.0f, 1.0f); // Gira el cavaller per la dreta.
                transform.Translate(Vector3.right * speed * Time.deltaTime); // Desplaça el cavaller a la dreta.
            }
            else // Si el cronòmetre és menor que 4 segons.
            {
                transform.localScale = new Vector3(-1.0f, 1.0f, 1.0f); // Gira el cavaller per l'esquerra.
                transform.Translate(Vector3.left * speed * Time.deltaTime); // Desplaça el cavaller a l'esquerra.
            }
            if (cronometre >= 8) // Si el cronòmetre arriba als 8 segons.
            {
                cronometre = 0; // Reinicia el cronòmetre a 0.
            }
        }

        if (vidatext != null) // Si el text de vida no és nul.
        {
            if (transform.localScale.x < 0) // Si el cavaller està mirant a l'esquerra.
            {
                vidatext.transform.localScale = new Vector3(1.0f, 1.0f, 1.0f); // Ajusta l'escala del text per que estigui orientat cap a la dreta.
            }
            else // Si el cavaller està mirant a la dreta.
            {
                vidatext.transform.localScale = new Vector3(-1.0f, 1.0f, 1.0f); // Ajusta l'escala del text per que estigui orientat cap a l'esquerra.
            }
```

```
    }
  }

  public int health = 10; // Declara una variable pública per la salut del cavaller.
   public int damageAmount = 1; // Declara una variable per la quantitat de dany que el cavaller pot rebre.
   public float damageAnimationDuration = 0.3f; // Estableix la durada de l'animació de dany.
   [SerializeField] TextMeshProUGUI vidatext; // Declara una variable per mostrar la salut en un TextMeshPro.

  private void ActualitzaVida() // Mètode per actualitzar el text de la salut.
  {
    vidatext.text = health + " ♥"; // Actualitza el text mostrant la salut actual del cavaller.
  }

   void OnTriggerEnter2D(Collider2D other) // Mètode que es crida quan el cavaller entra en contacte amb un altre col·lisionador.
  {
        if (other.gameObject.CompareTag("Pedra")) // Si l'objecte amb el qual entra en contacte té l'etiqueta "Pedra".
    {
         TakeDamage(damageAmount); // Crida al mètode TakeDamage per restar vida al cavaller.
        StartCoroutine(PlayDamageAnimation()); // Inicia l'animació de dany.
        caballer.Play(); // Reprodueix el so de dany.
    }
  }

   private IEnumerator PlayDamageAnimation() // Mètode per jugar l'animació de dany
```

```
durant un temps determinat.
    {
        Animator.SetTrigger("mal"); // Activa l'animació de dany.
         yield return new WaitForSeconds(damageAnimationDuration); // Espera el temps de durada de l'animació.
        Animator.ResetTrigger("mal"); // Desactiva l'animació de dany.
    }

    void TakeDamage(int damage) // Mètode per restar vida al cavaller quan rep dany.
    {
        health -= damage; // Resta la quantitat de dany a la salut.
        ActualitzaVida(); // Actualitza el text de la vida.
         Debug.Log("L'enemic ha rebut mal! Vida restant: " + health); // Mostra un missatge a la consola amb la vida restant.

        if (health <= 0) // Si la salut del cavaller arriba a 0 o menys.
        {
            Die(); // Crida al mètode Die per destruir el cavaller.
        }
    }

    void Die() // Mètode per gestionar la mort del cavaller.
    {
         Debug.Log("L'enemic ha mort!"); // Mostra un missatge a la consola indicant que el cavaller ha mort.
        Destroy(gameObject); // Destrueix l'objecte del cavaller.
        CastellScript.StopMusic(); // Para la música del castell quan el cavaller mor.
    }
}
```

CARGOL POMA

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using TMPro;

public class CaragolPomaScript : MonoBehaviour // Classe que controla el comportament del cargol.
{
    public Transform player; // Referència al jugador.
    public float detectionRadius = 12.0f; // Radi de detecció del jugador.
    public float speed = 1.5f; // Velocitat de moviment del cargol.
    public float cronometre; // Cronòmetre per al moviment aleatori.
    public float fadeDistance = 15.0f; // Distància a la qual el volum disminueix.

    private float Horizontal; // Variable per controlar la direcció en l'eix X.
    private Rigidbody2D rb; // Referència al Rigidbody2D del cargol.
    private Vector2 movement; // Vector de moviment per al cargol.
    public Animator Animator; // Referència a l'animador.
    private AudioSource caragolpoma; // Referència a l'AudioSource per reproduir so.

    void Start() // Mètode que s'executa al començar el joc.
    {
        rb = GetComponent<Rigidbody2D>(); // Obtén el Rigidbody2D del cargol.
        caragolpoma = GetComponent<AudioSource>(); // Obtén el AudioSource del cargol.
    }

    void Update() // Mètode que s'executa cada fotograma.
    {
            float distanceToPlayer = Vector2.Distance(transform.position, player.position); //
```

```
Calcula la distància al jugador.
      cronometre += 1 * Time.deltaTime; // Incrementa el cronòmetre.

      if (distanceToPlayer < detectionRadius) // Si el jugador està dins del radi de detecció.
      {
            Vector2 direction = (player.position - transform.position).normalized; // Direcció
cap al jugador.
          movement = new Vector2(direction.x, 0); // Moviment en l'eix X.
              rb.MovePosition(rb.position + movement * speed * Time.deltaTime); // Mou el
cargol cap al jugador.

                Horizontal = transform.position.x - player.transform.position.x; // Calcula la
direcció en l'eix X.

            if (Horizontal > 0.0f) transform.localScale = new Vector3(-1.0f, 1.0f, 1.0f); // Mira
cap a l'esquerra.
            else if (Horizontal < 0.0f) transform.localScale = new Vector3(1.0f, 1.0f, 1.0f); //
Mira cap a la dreta.
      }
      else // Si el jugador està fora de radi de detecció.
      {
          if (cronometre >= 4) // Si el cronòmetre és superior a 4.
          {
              transform.localScale = new Vector3(1.0f, 1.0f, 1.0f); // Mira cap a la dreta.
                  transform.Translate(Vector3.right * speed * Time.deltaTime); // Mou el cargol
cap a la dreta.
          }
          else
          {
              transform.localScale = new Vector3(-1.0f, 1.0f, 1.0f); // Mira cap a l'esquerra.
```

```
            transform.Translate(Vector3.left * speed * Time.deltaTime); // Mou el cargol cap
a l'esquerra.
        }

        if (cronometre >= 8) // Si el cronòmetre arriba a 8, es reinicia.
        {
            cronometre = 0;
        }
    }

    if (vidatext != null) // Actualitza la direcció del text de vida en funció de la direcció del
cargol.
    {
        if (transform.localScale.x < 0)
        {
            vidatext.transform.localScale = new Vector3(1.0f, 1.0f, 1.0f); // Mira cap a la
dreta.
        }
        else
        {
            vidatext.transform.localScale = new Vector3(-1.0f, 1.0f, 1.0f); // Mira cap a
l'esquerra.
        }
    }

    float volume = Mathf.Clamp01(1 - (distanceToPlayer / fadeDistance)); // Calcula el
volum segons la distància.
    caragolpoma.volume = volume; // Aplica el volum al so.

    if (volume > 0 && !caragolpoma.isPlaying) // Si el volum és major que 0 i no es
```

```
reprodueix, comença el so.
        {
            caragolpoma.Play();
        }
         else if (volume <= 0 && caragolpoma.isPlaying) // Si el volum és 0 o menys, para el
so.
        {
            caragolpoma.Stop();
        }
    }

    public int health = 20; // Vida del cargol.
    public int damageAmount = 3; // Dany que rep el cargol.

    [SerializeField] TextMeshProUGUI vidatext; // Text per mostrar la vida a la UI.

    private void ActualitzaVida() // Actualitza el text de la vida.
    {
        vidatext.text = health + " ♥";
    }

    void OnTriggerEnter2D(Collider2D other) // Mètode que detecta col·lisions.
    {
        if (other.gameObject.CompareTag("Pedra")) // Si col·lideix amb una pedra, rep dany.
        {
            TakeDamage(damageAmount);
        }
    }

    void TakeDamage(int damage) // Quan rep dany.
```

```
    {
        health -= damage; // Redueix la vida.
        ActualitzaVida(); // Actualitza el text de la vida.
         Debug.Log("L'enemic ha rebut mal! Vida restant: " + health); // Mostra per consola la
vida restant.

        if (health <= 0) // Si la vida arriba a 0, mor.
        {
            Die();
        }
    }

    void Die() // Quan mor el cargol.
    {
        Debug.Log("L'enemic ha mort!"); // Mostra per consola que el cargol ha mort.
        Destroy(gameObject); // Destrueix l'objecte del cargol.
    }
}
```

CARXOFA

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class CarxofaScript : MonoBehaviour // Classe que controla el comportament de la
carxofa.
{
    public Transform player; // Referència al jugador.
    public float speed = 0.3f; // Velocitat de moviment de la carxofa.
```

```
    public float cronometre; // Cronòmetre per alternar el moviment vertical.

    private Rigidbody2D rb; // Referència al Rigidbody2D de la carxofa.
    private Vector2 movement; // Vector de moviment (no s'utilitza actualment).

    void Start() // Mètode que s'executa al començar el joc.
    {
        rb = GetComponent<Rigidbody2D>(); // Obtén el Rigidbody2D de la carxofa.
    }

    void Update() // Mètode que s'executa cada fotograma.
    {
        cronometre += 1 * Time.deltaTime; // Incrementa el cronòmetre.

        if (cronometre >= 1) // Si el cronòmetre és més gran o igual a 1, mou cap amunt.
        {
            transform.position += Vector3.up * speed * Time.deltaTime; // Mou la carxofa cap
amunt.
        }
        else // Si el cronòmetre és menor a 1, mou cap avall.
        {
            transform.position += Vector3.down * speed * Time.deltaTime; // Mou la carxofa
cap avall.
        }

        if (cronometre >= 2) // Si el cronòmetre arriba a 2, es reinicia.
        {
            cronometre = 0;
        }
    }
```

```
    public void OnTriggerEnter2D(Collider2D collision) // Mètode que detecta col·lisions.
    {
        if (collision.CompareTag("Player")) // Si col·lideix amb el jugador, restaurar vida.
        {
            VidaScript VidaScript = collision.GetComponent<VidaScript>(); // Obté el script de vida del jugador.
            if (VidaScript != null) // Si es troba el script de vida, es restaura la vida.
            {
                VidaScript.RestoreHealth(); // Restaura la vida del jugador.
            }

            Destroy(gameObject); // Destrueix la carxofa després de col·lisionar amb el jugador.
        }
    }
}
```

CASTELL

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class CastellScript : MonoBehaviour // Classe que controla el comportament del castell.
{
    public Transform player; // Referència al jugador.
        public float fadeDistance = 20.0f; // Distància a la qual la música comença a desaparèixer.
    private bool caballermort = false; // Variable per saber si el caballer ha mort.
```

```
public AudioSource castell; // Font de so per al so del castell.
public AudioSource musica; // Font de so per a la música de fons.

void Update() // Mètode que s'executa cada fotograma.
{
    if (caballermort) return; // Si el caballer està mort, s'atura l'execució del mètode.

        float distance = Vector2.Distance(transform.position, player.position); // Calcula la
distància entre el castell i el jugador.
    float volume = Mathf.Clamp01(1 - (distance / fadeDistance)); // Calcula el volum de la
música segons la distància.
    castell.volume = volume; // Aplica el volum al so del castell.

    if (distance <= fadeDistance) // Si el jugador està dins de la distància definida.
    {
        if (!castell.isPlaying) // Si el so del castell no està sonant.
        {
            if (musica.isPlaying) // Si la música està sonant, es para.
            {
                musica.Stop();
            }
            castell.Play(); // Comença a sonar el so del castell.
        }
    }
    else // Si el jugador està fora de la distància definida.
    {
        if (castell.isPlaying) // Si el so del castell està sonant, es para.
        {
            castell.Stop();
            if (!musica.isPlaying) // Si la música no està sonant, es torna a començar.
```

```
            {
                musica.Play();
            }
        }
    }

}

public void StopMusic() // Mètode per aturar la música quan el caballer mor.
{
    caballermort = true; // Marca el caballer com mort.
    castell.Stop(); // Para el so del castell.
    if (!musica.isPlaying) // Si la música no està sonant, es reprodueix.
    {
        musica.Play();
    }
}
}
```

CIGONYA

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using TMPro; // Importa TextMeshPro per gestionar el text.

public class CigonyaScript : MonoBehaviour // Classe que controla el comportament de la cigonya.
{
    public Transform player; // Referència al jugador.
```

```
    public float detectionRadius = 6.0f; // Radi de detecció del jugador.
    public float speed = 3.0f; // Velocitat de la cigonya.
    public float cronometre; // Temporitzador per controlar els moviments de la cigonya.

     private float Horizontal; // Variable per emmagatzemar la direcció horitzontal entre la cigonya i el jugador.
    private Rigidbody2D rb; // Referència al Rigidbody2D per controlar la física.
    private Vector2 movement; // Vector per emmagatzemar la direcció del moviment.
    private AudioSource cigonya; // Referència a la font de so de la cigonya.

    void Start() // Mètode que s'executa quan comença el joc.
    {
        rb = GetComponent<Rigidbody2D>(); // Obté el component Rigidbody2D.
        cigonya = GetComponent<AudioSource>(); // Obté la font de so de la cigonya.
    }

    void Update() // Mètode que s'executa cada fotograma.
    {
            float distanceToPlayer = Vector2.Distance(transform.position, player.position); // Calcula la distància al jugador.
        cronometre += 1 * Time.deltaTime; // Incrementa el cronòmetre.

        if (distanceToPlayer < detectionRadius) // Si el jugador està dins del radi de detecció.
        {
            Vector2 direction = (player.position - transform.position).normalized; // Calcula la direcció cap al jugador.
            movement = new Vector2(direction.x, direction.y); // Defineix el moviment.

                rb.MovePosition(rb.position + movement * speed * Time.deltaTime); // Mou la cigonya cap a la direcció del jugador.
```

```
            Horizontal = transform.position.x - player.transform.position.x; // Calcula la diferència en l'eix X.

            if (Horizontal > 0.0f) transform.localScale = new Vector3(-1.0f, 1.0f, 1.0f); // Inverteix l'orientació si el jugador està a l'esquerra.
            else if (Horizontal < 0.0f) transform.localScale = new Vector3(1.0f, 1.0f, 1.0f); // Inverteix l'orientació si el jugador està a la dreta.
        }
        else // Si el jugador no està en el radi de detecció.
        {
            if (cronometre >= 4) // Si el cronòmetre és més gran que 4 segons.
            {
                transform.localScale = new Vector3(1.0f, 1.0f, 1.0f); // Fa que la cigonya miri a la dreta.
                transform.Translate(Vector3.right * speed * Time.deltaTime); // Mou la cigonya a la dreta.
            }
            else
            {
                transform.localScale = new Vector3(-1.0f, 1.0f, 1.0f); // Fa que la cigonya miri a l'esquerra.
                transform.Translate(Vector3.left * speed * Time.deltaTime); // Mou la cigonya a l'esquerra.
            }

            if (cronometre >= 8) // Si el cronòmetre arriba a 8, es reinicia.
            {
                cronometre = 0;
            }
```

```
    }

    if (vidatext != null) // Si el text per la vida existeix.
    {
        if (transform.localScale.x < 0) // Si la cigonya està orientada a l'esquerra.
        {
            vidatext.transform.localScale = new Vector3(-1.0f, 1.0f, 1.0f); // Inverteix el text per que es llegeixi bé.
        }
        else // Si la cigonya està orientada a la dreta.
        {
            vidatext.transform.localScale = new Vector3(1.0f, 1.0f, 1.0f); // Manté el text en la seva orientació normal.
        }
    }
}

public int health = 4; // Vida inicial de la cigonya.
public int damageAmount = 1; // Quantitat de dany que rep.
[SerializeField] TextMeshProUGUI vidatext; // Referència al text que mostra la vida.

private void ActualitzaVida() // Mètode per actualitzar la vida en el text.
{
    vidatext.text = health + " ♥"; // Actualitza el text amb la vida actual.
}

void OnTriggerEnter2D(Collider2D other) // Mètode que es crida quan la cigonya entra en contacte amb un altre collider.
{
    if (other.gameObject.CompareTag("Pedra")) // Si l'objecte amb què col·lideix té el tag
```

```
"Pedra".
        {
            TakeDamage(damageAmount); // Aplica dany.
            cigonya.Play(); // Reprodueix el so de la cigonya.
        }
    }

    void TakeDamage(int damage) // Mètode per aplicar dany a la cigonya.
    {
        health -= damage; // Redueix la vida.
        ActualitzaVida(); // Actualitza el text de la vida.
         Debug.Log("L'enemic ha rebut mal! Vida restant: " + health); // Mostra la vida restant
al log.

        if (health <= 0) // Si la vida és 0 o menor.
        {
            Die(); // Crida al mètode de mort.
        }
    }

    void Die() // Mètode per quan la cigonya mor.
    {
        Debug.Log("L'enemic ha mort!"); // Mostra un missatge a la consola.
        Destroy(gameObject); // Destrueix l'objecte (la cigonya).
    }
}
```

CÀMERA

```
using System.Collections;
```

```
using System.Collections.Generic;
using UnityEngine;

public class CameraScript : MonoBehaviour // Classe que controla la càmera.
{
   public GameObject Pescador; // Referència al GameObject del pescador.

   void Update() // Mètode que s'executa cada fotograma.
   {
      Vector3 position = transform.position; // Guarda la posició actual de la càmera.
         position.x = Pescador.transform.position.x; // Ajusta la posició en X per seguir al pescador.
        position.y = Pescador.gameObject.transform.position.y + 3; // Ajusta la posició en Y per tal d'estar una mica per sobre del pescador.
      transform.position = position; // Actualitza la posició de la càmera.
   }
}
```

PANTALLA DE DERROTA

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement; // Importa SceneManager per gestionar escenes en Unity.

public class GameOver : MonoBehaviour // Classe per gestionar l'escena de "Game Over".
{
   public void Tornar() // Mètode per tornar a l'escena inicial.
   {
```

```
        SceneManager.LoadScene("MenuInicial"); // Carrega l'escena "MenuInicial".
    }

    public void Sortir() // Mètode per sortir del joc.
    {
        Debug.Log("Sortir..."); // Mostra un missatge al log indicant que es sortirà del joc.
        Application.Quit(); // Tanca l'aplicació o el joc.
    }
}
```

PANTALLA DE VICTÒRIA

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class WinScript : MonoBehaviour // Script per gestionar la lògica de guanyar.
{
    public string winSceneName = "WinScene"; // Nom de l'escena que es carrega quan es
guanya.

    // Funció que s'executa quan un altre objecte entra en el collider.
    private void OnTriggerEnter2D(Collider2D other)
    {
        // Comprova si l'objecte que ha entrat al trigger té el tag "Player".
        if (other.CompareTag("Player"))
        {
            WinGame(); // Si és el jugador, executa la funció per guanyar.
        }
```

```
    }

    // Funció per carregar l'escena de guanyar.
    void WinGame()
    {
        SceneManager.LoadScene(winSceneName); // Carrega l'escena especificada en la variable winSceneName.
    }
}
```

PORC SENGLAR

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine; // Importa Unity per a components com Rigidbody2D, Animator i AudioSource.
using TMPro; // Importa TextMeshPro per mostrar text a la pantalla.

public class JabaliScript : MonoBehaviour // Classe per controlar el comportament del porc senglar.
{
    public Transform player; // Referència al jugador.
    public float detectionRadius = 6.0f; // Radi de detecció del jugador.
    public float speed = 3.0f; // Velocitat de moviment del porc senglar.
    public float cronometre; // Cronòmetre per controlar el moviment.
    private bool atac; // Variable per saber si està atacant.

    private float Horizontal; // Diferència de posició horitzontal entre el porc i el jugador.
    private Rigidbody2D rb; // Component Rigidbody2D per controlar el moviment del porc.
    private Vector2 movement; // Direcció del moviment.
```

```
    public Animator Animator; // Component Animator per controlar les animacions.
    private AudioSource jabali; // Component AudioSource per reproduir sons.

    void Start()
    {
        rb = GetComponent<Rigidbody2D>(); // Inicialitza Rigidbody2D per al moviment.
        jabali = GetComponent<AudioSource>(); // Inicialitza el component de so del porc.
    }

    void Update()
    {
            float distanceToPlayer = Vector2.Distance(transform.position, player.position); // Calcula la distància entre el porc i el jugador.
        cronometre += 1 * Time.deltaTime; // Incrementa el cronòmetre.

        if (distanceToPlayer < detectionRadius) // Si el jugador està dins del radi de detecció.
        {
            Animator.SetBool("atac", true); // Activa l'animació d'atac.

            Vector2 direction = (player.position - transform.position).normalized; // Calcula la direcció cap al jugador.

            movement = new Vector2(direction.x, 0); // Moviment horitzontal cap al jugador.

            rb.MovePosition(rb.position + movement * speed * Time.deltaTime); // Mou el porc cap al jugador.

                Horizontal = transform.position.x - player.transform.position.x; // Calcula la direcció horitzontal.
```

```
            if (Horizontal > 0.0f) transform.localScale = new Vector3(-1.0f, 1.0f, 1.0f); // Inverteix la direcció del porc si està a l'esquerra del jugador.
            else if (Horizontal < 0.0f) transform.localScale = new Vector3(1.0f, 1.0f, 1.0f); // Inverteix la direcció si està a la dreta del jugador.
        }
        else // Si el jugador està fora del radi de detecció.
        {
            Animator.SetBool("atac", false); // Desactiva l'animació d'atac.

            if (cronometre >= 4) // Si el cronòmetre és superior a 4, mou el porc cap a la dreta.
            {
                transform.localScale = new Vector3(1.0f, 1.0f, 1.0f);
                transform.Translate(Vector3.right * speed * Time.deltaTime);
            }
            else // Si no, mou el porc cap a l'esquerra.
            {
                transform.localScale = new Vector3(-1.0f, 1.0f, 1.0f);
                transform.Translate(Vector3.left * speed * Time.deltaTime);
            }

            if (cronometre >= 8) // Reseteja el cronòmetre després de 8 segons.
            {
                cronometre = 0;
            }
        }

        if (vidatext != null) // Actualitza la direcció del text segons la posició del porc.
        {
            if (transform.localScale.x < 0)
            {
```

```
            vidatext.transform.localScale = new Vector3(-1.0f, 1.0f, 1.0f);
        }
        else
        {
            vidatext.transform.localScale = new Vector3(1.0f, 1.0f, 1.0f);
        }
    }
}

public int health = 4; // Vida inicial del porc senglar.
public int damageAmount = 1; // Dany que el porc fa al jugador.

[SerializeField] TextMeshProUGUI vidatext; // Text que mostra la vida del porc.

private void ActualitzaVida() // Actualitza la vida del porc en el text.
{
    vidatext.text = health + " ♥";
}

    void OnTriggerEnter2D(Collider2D other) // Quan el porc entra en contacte amb un
objecte.
{
    if (other.gameObject.CompareTag("Pedra")) // Si l'objecte és una pedra.
    {
        TakeDamage(damageAmount); // Fa dany al porc.
        jabali.Play(); // Reprodueix un so quan el porc rep dany.
    }
}

void TakeDamage(int damage) // Redueix la vida del porc.
```

```
    {
        health -= damage; // Redueix la vida segons el dany rebut.
        ActualitzaVida(); // Actualitza la visualització de la vida.
         Debug.Log("L'enemic ha rebut mal! Vida restant: " + health); // Mostra el missatge de dany a la consola.

        if (health <= 0) // Si la vida arriba a 0, mata el porc.
        {
            Die();
        }
    }

    void Die() // Mata el porc.
    {
        Debug.Log("L'enemic ha mort!"); // Mostra el missatge de mort.
        Destroy(gameObject); // Destrueix el gameObject, eliminant el porc.
    }
}
```

MENÚ INICIAL

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement; // Importa SceneManager per carregar o gestionar scenes.

public class MenuInicial : MonoBehaviour // Classe per controlar el comportament del menú inicial.
{
```

```
    public void Jugar() // Mètode que es crida per començar el joc.
    {
        SceneManager.LoadScene("SampleScene"); // Carrega l'escena "SampleScene" quan es prem el botó de jugar.
    }

    public void Sortir() // Mètode que es crida per sortir del joc.
    {
        Debug.Log("Sortir..."); // Mostra un missatge a la consola quan es prem sortir.
        Application.Quit(); // Tanca l'aplicació o el joc.
    }
}
```

MOSCA NEGRA

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using TMPro;

public class MoscaScript : MonoBehaviour // Classe que gestiona el comportament de la mosca.
{
    public Transform player; // Referència al jugador.
    public float detectionRadius = 6.0f; // Radi de detecció de la mosca.
    public float speed = 2.5f; // Velocitat de la mosca.
    public float cronometre; // Cronòmetre per controlar el moviment de la mosca.
    public Animator Animator; // Referència al component Animator per animacions.

    private float Horizontal; // Per controlar la direcció horitzontal.
```

```
    private Rigidbody2D rb; // Per controlar el moviment físic de la mosca.
    private Vector2 movement; // Vector de moviment per moure la mosca.
    public float fadeDistance = 15.0f; // Distància per controlar el volum de l'àudio.
    private AudioSource mosca; // Per reproduir els sons de la mosca.

    void Start() // Inicialització de components.
    {
        rb = GetComponent<Rigidbody2D>(); // Obté el Rigidbody2D per al moviment físic.
            mosca = GetComponent<AudioSource>(); // Obté el component AudioSource per controlar el so.
    }

    void Update() // Actualitza cada frame.
    {
            float distanceToPlayer = Vector2.Distance(transform.position, player.position); // Calcula la distància del jugador.
        cronometre += 1 * Time.deltaTime; // Incrementa el cronòmetre.

        if (distanceToPlayer < detectionRadius) // Si el jugador està dins del radi de detecció.
        {
            Vector2 direction = (player.position - transform.position).normalized; // Direcció cap al jugador.

            movement = new Vector2(direction.x, direction.y); // Mou la mosca en direcció al jugador.
            rb.MovePosition(rb.position + movement * speed * Time.deltaTime); // Aplica el moviment al Rigidbody2D.

                Horizontal = transform.position.x - player.transform.position.x; // Calcula la direcció horitzontal per a l'escala.
```

```
        // Canvia l'escala per mirar cap al jugador.
        if (Horizontal > 0.0f) transform.localScale = new Vector3(1.0f, 1.0f, 1.0f);
        else if (Horizontal < 0.0f) transform.localScale = new Vector3(-1.0f, 1.0f, 1.0f);
    }
    else // Si el jugador no està dins del radi de detecció.
    {
        if (cronometre >= 4) // Moviment alternatiu per a la mosca.
        {
            transform.localScale = new Vector3(-1.0f, 1.0f, 1.0f);
            transform.Translate(Vector3.right * speed * Time.deltaTime);
        }
        else
        {
            transform.localScale = new Vector3(1.0f, 1.0f, 1.0f);
            transform.Translate(Vector3.left * speed * Time.deltaTime);
        }

        if (cronometre >= 8) cronometre = 0; // Reseteja el cronòmetre.
    }

      if (vidatext != null) // Si existeix el text de vida, ajusta la seva escala segons
l'orientació de la mosca.
    {
        if (transform.localScale.x < 0)
        {
            vidatext.transform.localScale = new Vector3(-1.0f, 1.0f, 1.0f);
        }
        else
        {
```

```
            vidatext.transform.localScale = new Vector3(1.0f, 1.0f, 1.0f);
        }
    }

    // Controla el volum de l'àudio de la mosca en funció de la distància al jugador.
    float volume = Mathf.Clamp01(1 - (distanceToPlayer / fadeDistance));
    mosca.volume = volume;

     if (volume > 0 && !mosca.isPlaying) // Reprodueix el so si la mosca es troba prou a prop del jugador.
    {
        mosca.Play();
    }
    else if (volume <= 0 && mosca.isPlaying) // Atura el so si la mosca està massa lluny.
    {
        mosca.Stop();
    }
}

public int health = 6; // Vida de la mosca.
public int damageAmount = 1; // Dany que fa la mosca al jugador.
public float damageAnimationDuration = 0.3f; // Durada de l'animació de dany.
 [SerializeField] TextMeshProUGUI vidatext; // Referència al text de vida per mostrar-lo a la pantalla.

private void ActualitzaVida() // Actualitza el text de la vida.
{
    vidatext.text = health + " ♥"; // Mostra la vida de la mosca.
}
```

```
    void OnTriggerEnter2D(Collider2D other) // Si la mosca toca un objecte amb el tag "Pedra".
    {
        if (other.gameObject.CompareTag("Pedra"))
        {
            TakeDamage(damageAmount); // Redueix la vida.
            StartCoroutine(PlayDamageAnimation()); // Reprodueix l'animació de dany.
        }
    }

    private IEnumerator PlayDamageAnimation() // Animació de dany.
    {
        Animator.SetTrigger("mal"); // Activa l'animació de dany.
         yield return new WaitForSeconds(damageAnimationDuration); // Espera la durada de l'animació.
        Animator.ResetTrigger("mal"); // Restableix l'animació després de la seva durada.
    }

    void TakeDamage(int damage) // Redueix la vida de la mosca.
    {
        health -= damage; // Descompta el dany de la vida.
        ActualitzaVida(); // Actualitza el text de vida.
        Debug.Log("L'enemic ha rebut mal! Vida restant: " + health); // Mostra a la consola.

        if (health <= 0) // Si la vida arriba a zero, la mosca mor.
        {
            Die();
        }
    }
```

```
    void Die() // Quan la mosca mor.
    {
        Debug.Log("L'enemic ha mort!"); // Mostra a la consola.
        Destroy(gameObject); // Destrueix la mosca.
        mosca.Stop(); // Atura el so quan la mosca mor.
    }
}
```

CENTRAL NUCLEAR

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class NuclearScript : MonoBehaviour // Classe que gestiona el comportament d'un objecte nuclear.
{
    public Transform player; // Referència al jugador per controlar la distància.
    public float fadeDistance = 20.0f; // Distància a partir de la qual el so serà nul.

     private AudioSource nuclear; // Referència al component AudioSource per reproduir el so nuclear.

    void Start() // Inicialització dels components.
    {
            nuclear = GetComponent<AudioSource>(); // Obté el component AudioSource per controlar el so nuclear.
    }

    void Update() // Actualització cada frame.
```

```
    {
        float distance = Vector2.Distance(transform.position, player.position); // Calcula la distància entre l'objecte nuclear i el jugador.
        float volume = Mathf.Clamp01(1 - (distance / fadeDistance)); // Calcula el volum basat en la distància. A mesura que el jugador s'allunya, el volum disminueix.
      nuclear.volume = volume; // Aplica el volum calculat al component AudioSource.

       if (volume > 0 && !nuclear.isPlaying) // Si el volum és positiu i el so no està sonant, reprodueix-lo.
      {
         nuclear.Play();
      }
       else if (volume <= 0 && nuclear.isPlaying) // Si el volum és zero o negatiu i el so està sonant, atura'l.
      {
         nuclear.Stop();
      }
    }
}
```

MENÚ DE PAUSA

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement; // Gestió del canvi d'escenes.

public class PausaScript : MonoBehaviour // Classe que gestiona la pausa del joc.
{
    public GameObject Pausa; // Referència a l'objecte de la UI per mostrar el menú de
```

```
pausa.
    private bool isPaused = true; // Indica si el joc està pausat o no.
    public float Temps; // Temps que ha passat des que es va iniciar la pausa.

    void Start() // Inicialització al començar.
    {
        Pausa.SetActive(false); // El menú de pausa es desactiva al principi.
        Time.timeScale = 1f;  // Es garanteix que el temps del joc corre normalment.
        isPaused = false; // El joc no està pausat inicialment.
    }

    private void Update() // Actualització cada frame.
    {
        Temps += Time.deltaTime; // Acumula el temps que passa.

        // Comprova si es prem la tecla Escape.
        if (Input.GetKeyDown(KeyCode.Escape))
        {
            Debug.Log("Tecla Escape premuda!"); // Imprimeix un missatge per a depuració.

            if (isPaused == false) // Si el joc no està pausat, l'atura.
            {
                Pausar(); // Activa el menú de pausa.
                Temps = 0f; // Reseteja el comptador de temps.
            }
            else // Si ja està pausat, el joc es reprèn.
            {
                Continuar(); // Desactiva el menú de pausa.
            }
        }
```

```
    }

    public void Pausar() // Funció per activar la pausa del joc.
    {
        Debug.Log("Intentant mostrar el menú."); // Missatge de depuració.
        Pausa.SetActive(true); // Mostra el menú de pausa.

        // Si ha passat més de 0.1 segons, es pausa el joc.
        if (Temps >= 0.1f)
        {
            Pausa.SetActive(true); // Assegura't que el menú de pausa es mostri.
            Time.timeScale = 0f; // Atura el temps del joc.
            isPaused = true; // Marca el joc com a pausat.
        }
    }

    public void Continuar() // Funció per reprendre el joc.
    {
        Pausa.SetActive(false); // Desactiva el menú de pausa.
        Time.timeScale = 1f; // Reprèn el temps del joc.
        isPaused = false; // Marca el joc com a no pausat.
         Input.ResetInputAxes(); // Restableix els eixos d'entrada, eliminant qualsevol entrada de teclat residual.
    }

    public void TornarComençar() // Funció per reiniciar la partida.
    {
        Time.timeScale = 1f; // Reprèn el temps.
         SceneManager.LoadScene(SceneManager.GetActiveScene().name); // Carrega de nou l'escena actual, reiniciant el joc.
```

```
        isPaused = false; // El joc ja no està pausat.
    }

    public void Sortir() // Funció per sortir del joc.
    {
        Debug.Log("Sortir..."); // Missatge de depuració per sortir.
        Application.Quit(); // Tanca l'aplicació.
    }
}
```

PEDRA

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PedraScript : MonoBehaviour // Gestiona el comportament de les pedres.
{
    private Rigidbody2D Rigidbody2D; // Referència al component Rigidbody2D de la pedra.
    public float Speed; // Velocitat a la qual es desplaça la pedra.
    private Vector3 Direction; // Direcció en què es mou la pedra.
    public float tempsPedra = 2; // Temps fins a la destrucció de la pedra.

    void Start()
    {
        Rigidbody2D = GetComponent<Rigidbody2D>(); // Obté el component Rigidbody2D.
        Destroy(gameObject, tempsPedra); // Destrueix l'objecte després de 'tempsPedra' segons.
    }
```

```
    void OnTriggerEnter2D(Collider2D other) // Quan la pedra entra en contacte amb un altre objecte.
    {
        if (other.gameObject.CompareTag("Enemy")) // Si l'objecte té l'etiqueta "Enemy".
        {
            Destroy(gameObject); // Destrueix la pedra.
        }
    }

    void Update() // Actualització cada frame.
    {
            Rigidbody2D.velocity = Direction * Speed; // Aplica la velocitat a la pedra en la direcció especificada.
    }

    // Funció per establir la direcció de la pedra.
    public void SetDirection(Vector2 direction)
    {
        Direction = direction; // Estableix la direcció de la pedra.
    }
}
```

SILUR

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class SiluroScript : MonoBehaviour // Classe que gestiona el comportament de
```

```
l'enemic Siluro.
{
   public Transform player; // Referència al jugador per determinar la seva posició.
   public float speed = 5.0f; // Velocitat de desplaçament del Siluro.
     public float cronometre; // Cronòmetre per gestionar els moviments alternatius (cap amunt o cap avall).

    private Rigidbody2D rb; // Referència al component Rigidbody2D per gestionar la física 2D.
   private Vector2 movement; // Vector per emmagatzemar el moviment.

   void Start()
   {
      rb = GetComponent<Rigidbody2D>(); // Obté el component Rigidbody2D del Siluro.
   }

   void Update()
   {
         cronometre += 1 * Time.deltaTime; // Incrementa el cronòmetre amb el temps que passa entre frames.

      // Moviment alternatiu (cap amunt o cap avall)
      if (cronometre >= 2)
      {
         transform.position += Vector3.down * speed * Time.deltaTime; // Mou el Siluro cap avall.
               transform.localScale = new Vector3(-1.0f, 1.0f, 1.0f); // Inverteix la direcció del Siluro.
      }
      else
```

```
        {
            transform.position += Vector3.up * speed * Time.deltaTime; // Mou el Siluro cap amunt.
            transform.localScale = new Vector3(1.0f, 1.0f, 1.0f); // Manté la direcció normal del Siluro.
        }

        // Reinicia el cronòmetre després de 4 segons.
        if (cronometre >= 4)
        {
            cronometre = 0;
        }
    }

    public int health = 4; // Vida del Siluro.
    public int damageAmount = 1; // Quantitat de dany que rep el Siluro.

    // Funció per quan el Siluro entra en contacte amb altres objectes.
    void OnTriggerEnter2D(Collider2D other)
    {
        if (other.gameObject.CompareTag("Pedra")) // Si el Siluro toca una "Pedra".
        {
            TakeDamage(damageAmount); // Rebi dany.
        }
    }

    // Funció per reduir la vida del Siluro quan rep dany.
    void TakeDamage(int damage)
    {
        health -= damage; // Redueix la vida.
```

```
        Debug.Log("L'enemic ha rebut mal! Vida restant: " + health); // Mostra el missatge en la consola.

        if (health <= 0) // Si la vida arriba a 0 o menys.
        {
            Die(); // El Siluro mor.
        }
    }

    // Funció per destruir el Siluro quan mor.
    void Die()
    {
        Debug.Log("L'enemic ha mort!"); // Missatge en la consola.
        Destroy(gameObject); // Destrueix l'objecte Siluro.
    }
}
```

VIDA

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
using TMPro;

public class VidaScript : MonoBehaviour // Script per gestionar la vida del jugador.
{
    public int health = 5; // Vida inicial del jugador.
     public int damageAmount = 1; // Quantitat de dany que rep el jugador quan col·lisiona amb un enemic.
```

```
    public Animator Animator; // Referència a l'animador per activar animacions de dany.
    public int tempsEspera = 1; // Temps d'espera per activar la recuperació de dany.

    private bool NoPotRebreMal = false; // Si el jugador no pot rebre mal durant un temps.
     [SerializeField] TextMeshProUGUI vidatext; // Referència al TextMeshPro per mostrar la vida del jugador.

    void Update()
    {
            // Actualitza l'escala del text per reflectir la direcció del jugador (mirant cap a l'esquerra o la dreta).
        if (vidatext != null)
        {
            if (transform.localScale.x < 0)
            {
                vidatext.transform.localScale = new Vector3(-1.0f, 1.0f, 1.0f); // Si el jugador mira cap a l'esquerra.
            }
            else
            {
                vidatext.transform.localScale = new Vector3(1.0f, 1.0f, 1.0f); // Si el jugador mira cap a la dreta.
            }
        }
    }

    // Quan el jugador deixa de col·lisionar amb un enemic.
    void OnCollisionExit2D(Collision2D other)
    {
        if (other.gameObject.CompareTag("Enemy"))
```

```
    {
        Animator.SetBool("mal", false); // Desactiva l'animació de rebre mal.
    }
}

// Quan el jugador deixa de tocar un enemic en un trigger.
void OnTriggerExit2D(Collider2D other)
{
    if (other.gameObject.CompareTag("Enemy"))
    {
        Animator.SetBool("mal", false); // Desactiva l'animació de rebre mal.
    }
}

// Quan el jugador col·lisiona amb un enemic.
void OnCollisionEnter2D(Collision2D other)
{
    if (other.gameObject.CompareTag("Enemy"))
    {
        TakeDamage(damageAmount); // El jugador rep dany.
        Animator.SetBool("mal", true); // Activa l'animació de rebre mal.
    }
}

// Quan el jugador entra en un trigger amb un enemic.
void OnTriggerEnter2D(Collider2D other)
{
    if (other.gameObject.CompareTag("Enemy") && !NoPotRebreMal)
    {
        TakeDamage(damageAmount); // El jugador rep dany.
```

```
            StartCoroutine(ActivarNoRebreMal()); // Activa la protecció temporal contra més danys.
            Animator.SetBool("mal", true); // Activa l'animació de rebre mal.
        }
    }

    // Funció per activar la protecció temporal contra més danys.
    private IEnumerator ActivarNoRebreMal()
    {
        NoPotRebreMal = true; // El jugador no pot rebre més mal.
        yield return new WaitForSeconds(tempsEspera); // Espera durant el temps especificat.
        NoPotRebreMal = false; // El jugador pot tornar a rebre danys.
    }

    // Funció per restaurar la salut del jugador.
    public void RestoreHealth()
    {
        health = 5; // Restaura la vida a la màxima.
        Debug.Log("Vida restaurada al màxim!"); // Missatge a la consola.
        ActualitzaVida(); // Actualitza la visualització de la vida.
    }

    // Funció per actualitzar el text de la vida en la UI.
    private void ActualitzaVida()
    {
        vidatext.text = health + " ♥"; // Mostra la vida actual amb el símbol de cor.
    }

    // Funció per gestionar la recepció de dany.
    void TakeDamage(int damage)
```

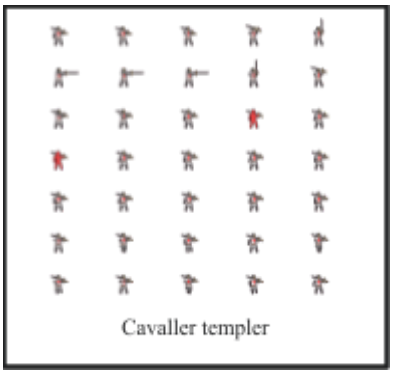
```
    {
        health -= damage; // Redueix la vida.
        ActualitzaVida(); // Actualitza la visualització de la vida.
         Debug.Log("El jugador ha rebut mal! Vida restant: " + health); // Mostra el missatge a
la consola.

        // Si la vida arriba a 0, el jugador mor.
        if (health <= 0)
        {
            Die();
        }
    }

    // Funció per gestionar la mort del jugador.
    void Die()
    {
        Debug.Log("El jugador ha mort!"); // Mostra el missatge a la consola.
        Destroy(gameObject); // Destrueix l'objecte del jugador.
        GameOver(); // Crida la funció per mostrar la pantalla de Game Over.
    }

    // Funció per carregar l'escena de Game Over.
    void GameOver()
    {
                    SceneManager.LoadScene("GameOver");  // Carrega l'escena anomenada
"GameOver".
    }
}
```

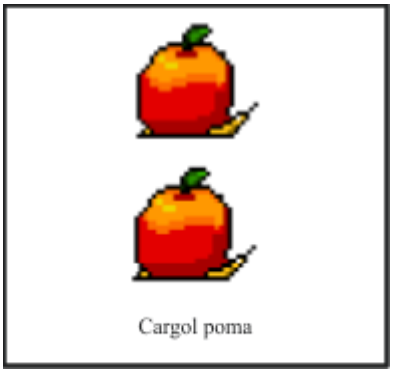
7.2 Disseny dels sprites del joc

En aquest apartat dels annexos es presenten tots els sprites que hem utilitzat per al nostre videojoc ordenats alfabèticament. En els respectius peus d'imatge es trobarà una petita definició de què representa cadascuna. Si en una imatge es troben diferents sprites vol dir que aquesta imatge es tracta d'una animació en el videojoc, que no és més que una seqüència de fotogrames. Aquesta secció permet entendre com són els personatges i els escenaris que es troben implementats, així com les seves textures i la seva gamma de colors.

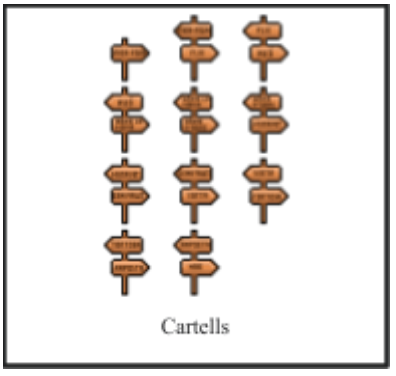

Aigua del riu Ebre

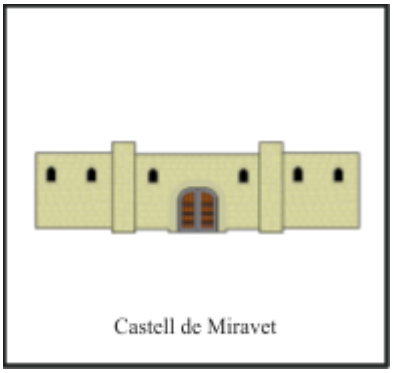

Arbre

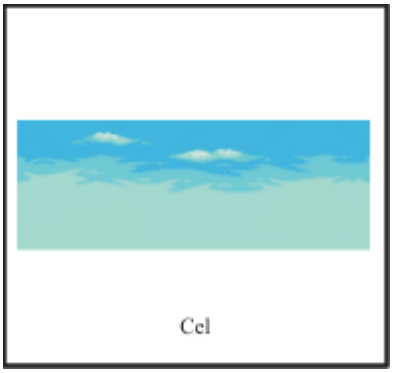

Cavaller templer

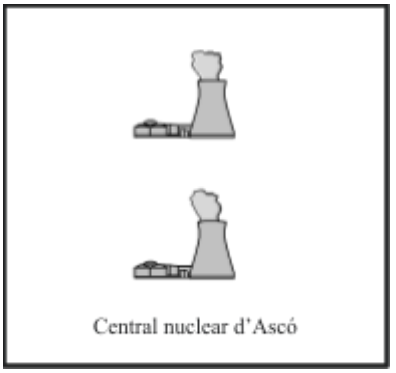

Cargol poma


Carxofa


Cartells


Castell de Miravet

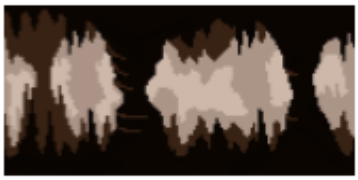

Cel


Central nuclear d'Ascó

Cigonya

Columna de pedra de les coves de Benifallet

Entrada a la cova de Benifallet

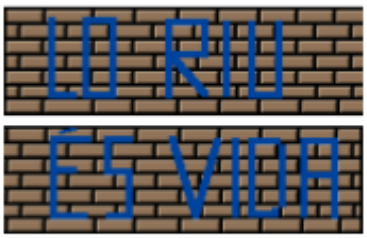


Fons de la cova de Benifallet

Porc senglar

Muntanyes de fons

Mosca negra

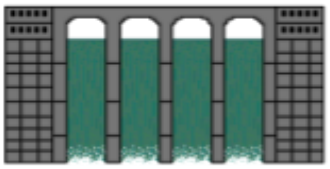


Mural "Lo riu és vida"

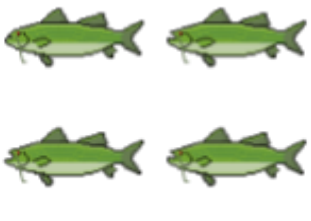
Pedra

Pescador

Pont d'arcades Móra d'Ebre i Móra la Nova



Presa de Riba-roja d'Ebre



Silur

Para-sol 1

Para-sol 2

Para-sol 3

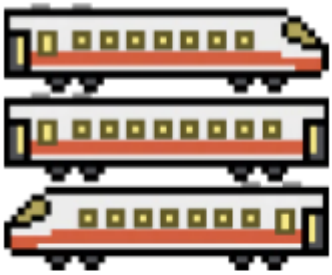
Sorra

Taronger

Terra

Terra de la cova

Terra del pont d'arcades



Tren de Móra la Nova

7.3 Captures del videojoc en funcionament

A continuació, mostrem diverses captures de pantalla del videojoc en funcionament. Aquestes imatges permeten veure com apareixen els personatges i els escenaris durant la seva execució. Les captures inclouen moments específics de la jugabilitat, com el lloc d'aparició i la interacció amb el cavaller. També afegim captures de les pantalles de menú que hem dissenyat.

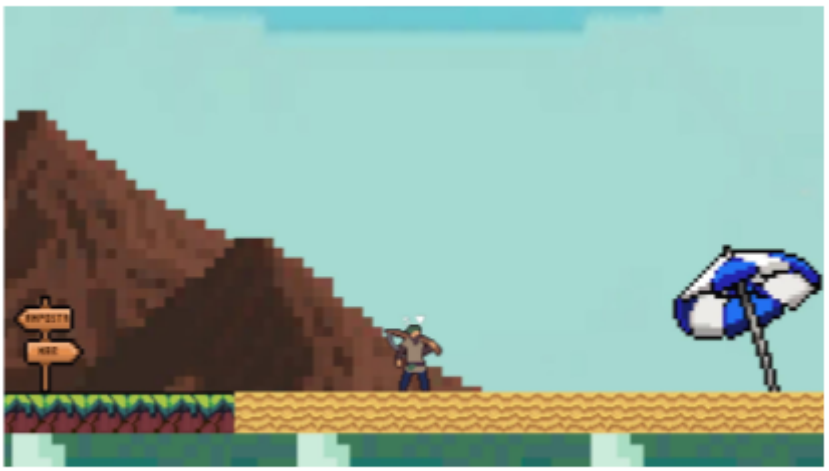
Menú d'inici

Lloc d'aparició

Menú de pausa

Lluita amb el cavaller

Pantalla de mort



Captura del final del videojoc

Pantalla de victòria

7.4 Àudios utilitzats en el joc

En aquest apartat presentem els àudios que hem utilitzat durant el desenvolupament del videojoc. Els sons inclouen efectes sonors com els passos del personatge, els sons dels enemics i els sons d'ambient, així com la música de fons que acompanya l'experiència de joc.

Tots els fitxers d'àudio han estat integrats a "Unity". Per permetre un accés ràpid a aquests recursos, hem creat una carpeta de "Google Drive" amb els àudios utilitzats.

Enllaç a la carpeta: Feu clic aquí per accedir als àudios

7.5 Tràiler i enllaç del videojoc

En els següents enllaços podreu accedir a les nostres xarxes socials, on trobareu el tràiler del nostre videojoc i més informació del llançament oficial.

- Youtube
- Instagram

També oferim un enllaç per jugar el videojoc creat durant el projecte. El joc es troba en la pàgina web "itch.io" i es pot jugar en línea.
Per jugar el videojoc, seguiu aquests passos:

- Feu clic a l'enllaç següent: Jugueu el videojoc.
- Premeu el botó "Run game".

FI DEL TREBALL