Universitat Oberta de Catalunya

uoc.edu

Disseny i desenvolupament d'un tower defense Earth Defense

Ferran Domènech Abril
Grau en enginyeria informàtica
05.640 TFG - Videojocs educatius

Helio Tejedor Navarro i Jordi Duch Gavaldà
Joan Arnedo Moreno

01/2019

FITXA DEL TREBALL FINAL

Títol del treball:	*Disseny i desenvolupament d'un tower defense Earth Defense*
Nom de l'autor:	*Ferran Domènech Abril*
Nom del consultor/a:	*Helio Tejedor Navarro* *Jordi Duch Gavaldà*
Nom del PRA:	*Joan Arnedo Moreno*
Data de lliurament (mm/aaaa):	*01/2019*
Titulació o programa:	*Grau en enginyeria informàtica*
Àrea del Treball Final:	*05.640 TFG - Videojocs educatius*
Idioma del treball:	*Català*
Paraules clau	*Unity, Tower defense, C#*

Resum del Treball (màxim 250 paraules):

El meu treball de final de grau consta de la creació d'un videojoc. El nom que li he posat és **Earth Defense** i el gènere de joc és ***tower defense***, un tipus el qual solia jugar molt quan era petit i que sempre havia volgut fer un projecte de l'estil.

Per a dur a terme el projecte he utilitzat el motor gràfic **Unity** i el llenguatge de programació ha estat el C#, un llenguatge que no havia utilitzat mai però que he vist que és molt senzill d'utilitzar i dona una gran versatilitat per a poder arribar a fer projectes més o menys complicats.

La plataforma de destí escollida ha estat ordinador Windows ja que, donat el temps i la novetat de treballar amb un motor gràfic que no havia utilitzat mai, sumat a un llenguatge nou per a mi, volia estar familiaritzat, al menys, amb el destí d'aquest projecte.

El resultat ha estat un videojoc molt bàsic però que compleix els requisits mínims dins del gènere; un joc que t'engresca i et repta a continuar i acabar els nivells i, a més, t'incita a descobrir quins tipus d'enemics o defenses hi ha dins dl joc.

En conclusió estic bastant satisfet amb el resultat esperat ja que, per a ser el meu primer joc, el temps limitat que he tingut i les eines, també noves per a mi, he pogut assolir els objectius esperats i acadèmics posats en ell.

Índex

1. Introducció

1.1 Context i justificació del Treball

Avui en dia, per a millor o pitjor, quasi tot el que es consumeix d'oci pel què fa a jocs en general, són els videojocs. Podríem dir que els jocs de taula, dins del món occidental i bona part de l'oriental, han passat a segon pla. És per això que contínuament es busquen millores i cada cop hi ha més empreses que busquen *el* videojoc que pugui desmarcar-se de tota la resta i obtenir uns bons resultats (tant a nivell econòmic com a nivell de consumidors). És per això que, dins de l'àmbit de la informàtica, és un tema que em crida molt l'atenció i, a més, en soc un consumidor habitual.

A més a més també hi ha diferents gèneres amb unes característiques bastant marcades. Un exemple seria el gènere del *Tower defense*, que és el gènere que utilitzaré en aquest treball final de grau, amb un *core* comú per a tots els videojocs d'aquest tipus: una sèrie d'enemics intenten acabar amb un nucli defensiu i el jugador l'ha de protegir construint defenses al llarg del recorregut.

Aquest gènere sol ser bastant bàsic però molt addictiu ja que es crea un repte al jugador que haurà d'anar millorant nivell rere nivell. També podríem dir que té molta part d'estratègia ja que s'ha de pensar molt bé com es ficaran les defenses, quin tipus de defenses s'han de posar per a segons quins enemics, veure quines són les millors opcions en cada moment (si rapidesa, força, habilitat, etc.) i adaptar-se bastant a cada nivell i les possibilitats que aquest ofereixi.

1.2 Objectius del Treball

Els objectius d'aquest treball final de carrera són:

OBJECTIU	DESCRIPCIÓ
Fer un joc amb Unity	Saber construir un prototip de videojoc i desenvolupar-lo satisfactòriament
Entendre el procés que segueix la creació d'un videojoc	Des de la idea i els esbossos, passant pel disseny i la implementació i acabant per tests a terceres persones alienes al projecte
Aprendre a fer ús de l'entorn Unity	Aprendre les funcionalitats que permet aquest entorn ja que, en la meva opinió, és una eina molt potent
Aprendre a programar en C#	Sabent com s'implementa codi C, C++ o Java, entre d'altres, un objectiu més d'aquest TFG és aprendre a codificar en C#
Introduir-me al món de les apps amb videojocs	Al finalitzar el TFG vull haver assolit un cert nivell de coneixement per a poder extrapolar aquest projecte a altres projectes fets per a plataformes Android (això requerirà més temps, ja que mai m'he enfocat cap aquesta direcció) i, d'aquesta manera, introduir-me dins del món de les apps de cara al meu futur professional

1.3 Enfocament i mètode seguit

Per a poder fer el projecte, la idea ha estat crear un joc des de zero i no fer-ne una adaptació o modificació de productes que ja estiguin al mercat. Per això, primerament he hagut de definir sobre quina plataforma volia treballar on, descartant-ne molts, tenia les tres principals opcions de:

1. PC
2. Android
3. iOS

La plataforma triada ha estat l'ordinador ja que estic més familiaritzat amb la programació per a aquest tipus de màquines i no sobre telèfons o tauletes (entre d'altres) que portin els altres dos sistemes.

A més, també he hagut de pensar sobre quin entorn volia fer tot el disseny i programació. Ha estat bastant fàcil ja que, a més de l'objectiu acadèmic de fer un joc "complet", tenia l'objectiu personal d'aprendre i arribar a utilitzar Unity d'una manera fluïda. És per això que l'entorn de programació ha estat aquest esmentat.

Finalment i possiblement el què m'ha costat més decidir i definir ha estat el tipus de joc i la seva història. Els gèneres principals els quals he estat dubtant han estat:

- Plataformes
- *Tower defense*
- Puzle
- Rol

Donada la complexitat que significava crear un videojoc de rol i el poc temps que tenia, aquest, el vaig descartar directament. Plataformes i puzles podrien haver estat una bona opció però requerien massa despesa de temps en quan a disseny de gràfics, animacions, etc. i no he cregut que sigui completament l'àmbit en el qual m'havia de centrar. *Tower defense*, en canvi, he cregut que podia aportar una càrrega de

treball de disseny normal combinada amb una càrrega, també estàndard, de programació.

En conclusió, un joc tipus *tower defense* per a ordinador i dissenyat amb el Unity ha estat una bona opció per a poder dur a terme un projecte en el qual, a més d'objectius acadèmics, també hi ha hagut uns objectius personals que, de ben segur, utilitzaré en algun moment de la meva carrera professional.

1.4 Planificació del Treball

1.4.1 Recursos principals

Per a poder dur a terme el meu treball de final de grau he necessitat, bàsicament, un motor de videojoc anomenat **Unity**. Gràcies a **Unity** es poden crear videojocs per a diferents plataformes i, segons les opcions instal·lades, utilitzar el llenguatge de programació que es desitgi. També és molt útil en quan a tractament d'esdeveniments ja que ofereix una gran quantitat de mètodes ja dissenyats per a més comoditat en el desenvolupament del codi. La part de disseny, a nivells bàsics, és molt intuïtiva i el codi implementat es pot complementar amb opcions i seleccions bastant senzilles d'utilitzar, incorporant una gran quantitat d'elements de desenvolupament com poden ser les físiques, la detecció de col·lisions, animacions, audio, materials, etc. A més Unity incorpora l'IDE **Visual Studio** per a poder programar i debugar de manera fàcil i ràpida.

També juga un bon paper el fet que **Unity** téla capacitat de compilar simulant diferents plataformes (iOs, Android, Linux, web, etc.)

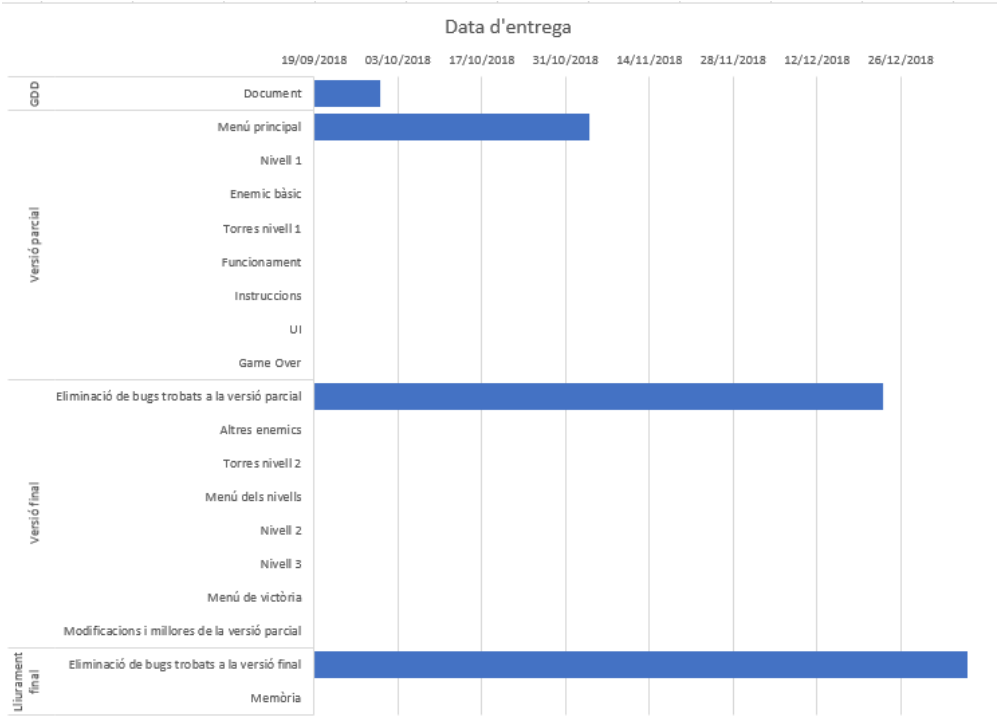
1.4.2 Recursos secundaris

A banda de Unity, també he necessitat altres recursos per a poder completa el meu treball:

- **GitHub** → Amb aquesta eina he creat un repositori per a poder compartir tota la feina amb els consultors de l'assignatura. El repositori creat ha estat el https://github.com/fdomenech9.
- **Unity Connect** → En aquesta comunitat he pogut actualitzar un *Showcase* del meu joc. La pàgina creada ha estat la https://connect.unity.com/u/5bb8d071edbc2a545d50d892.
- **Apowerrec** → Amb aquesta aplicació he pogut gravar els vídeos mostrant les diferents característiques del meu joc.
- **Youtube** → En aquesta xarxa he penjat els vídeos que es demanaven. La pàgina creada ha estat la https://youtu.be/9tdsPLm_C6c.

1.4.3 Tasques i planificació

Tasca	Subtasca	Data d'inici	Data d'entrega
GDD	Document	19/09/2018	30/09/2018
Versió parcial	Menú principal	01/10/2018	04/11/2018
	Nivell 1		
	Enemic bàsic		
	Torres nivell 1		
	Funcionament		
	Instruccions		
	UI		
	Game Over		
Versió final	Eliminació de bugs trobats a la versió parcial	05/11/2018	23/12/2018
	Altres enemics		
	Torres nivell 2		
	Menú dels nivells		
	Nivell 2		
	Nivell 3		
	Menú de victòria		
	Modificacions i millores de la versió parcial		
Lliurament final	Eliminació de bugs trobats a la versió final	24/12/2019	06/01/2019
	Memòria		


Data d'entrega
19/09/2018
03/10/2018
17/10/2018
31/10/2018
14/11/2018
28/11/2018
12/12/2018
26/12/2018
GDD
Document
Versió parcial
Menú principal
Nivell 1
Enemic bàsic
Torres nivell 1
Funcionament
Instruccions
UI
Game Over
Versió final
Eliminació de bugs trobats a la versió parcial
Altres enemics
Torres nivell 2
Menú dels nivells
Nivell 2
Nivell 3
Menú de victòria
Modificacions i millores de la versió parcial
Lliurament final
Eliminació de bugs trobats a la versió final
Memòria

1.5 Breu sumari de productes obtinguts

El videojoc Earth Defense té:

- Menú principal
- Menú de selecció de nivells
- Un primer nivell on tenim enemics bàsics
- Un segon nivell on els enemics són més ràpids i, conseqüentment, amb un augment de dificultat
- Un tercer nivell amb enemics amb molta més resistència un amb una dificultat afegida
- Un menú de final de joc que apareix un cop tots els nivells han estat completats

1.6 Breu descripció dels altres capítols de la memòria

- Arquitectura del joc → Diagrama de fluxos per a veure quina és l'arquitectura del joc.
- Implementació → Totes les APIs i els arxius de codi .cs que s'han implementat durant el projecte descrivint quines i per què són les seves funcionalitats.
- Actors → En aquest capítol es veuran les fitxes de tots els objectes principals obtinguts en el desenvolupament de Earth Defense.
- Mapa → Breu descripció dels elements principals que tenim en el mapa que són.
- User Interface → En aquest capítol es descriurà quins són els elements de la interfície d'usuari cm són.
- Animacions → Quines són les animacions que s'han creat per al projecte.
- Nivells → Descripció dels nivells i rondes.
- Unity Assets → Assets utilitzats per a poder dur a terme el projecte tal i com és.
- Versions → Quins han estat els continguts de les diferents versions.
- Experiència d'usuari → Valoracions d'usuaris que han provat el joc.

2. Context i marc teòric

2.1 Idea del joc

Earth Defense és un videojoc dins del gènere *tower defense* on, des d'un punt del mapa el qual l'usuari té en tot moment davant seu, es generen enemics que intentaran acabar amb el poblat.

Per a poder-se defensar dels enemics, el jugador haurà de comprar torres per acabar amb ells abans no arribin al seu objectiu amb els diners que es van guanyant un cop morts. A més, també hi ha un nombre límit de vides abans el jugador no mori. Aquestes torres, també, es podran pujar de nivell adquirint noves habilitats a mesura que la dificultat per a matar els enemics augmenta.

Un dels exemples més clàssics de jocs tipus *tower defense* és el **Plants vs Zombies** o el **Kingdom Rush.**

Kingdom Rush

Digfender

Plants vs Zombies

2.2 Conceptualització

El videojoc *Earth Defense* està ambientat en un món ple de bandolers que contínuament intenten robar als pobres pagesos del poble.
La interacció que hi ha dins del joc és, sobretot, “torre ataca a enemic” o “enemic ataca punt defensiu” en cas que arribessin a creuar les defenses.
També he intentat fer èmfasi en el tema dels nivells, on m’he inspirat en jocs com *Cute the Rope*, el qual tenim nivells bloquejats perquè no hi hem arribat, però en tot moment podem escollir si juguem l’últim nivell possible o qualsevol altra.

2.3 Desenvolupament i roadmap

Per a dissenyar i programar el videojoc tinc diferents entorns els quals podria utilitza:

- Unity → És un motor de joc molt complet. Gràcies a Unity es poden crear videojocs per a diferents plataformes i, segons les opcions instal·lades, utilitzar el llenguatge de programació que es desitgi.
  També és molt útil en quan a tractament d'esdeveniments ja que ofereix una gran quantitat de mètodes ja dissenyats per a més comoditat en el desenvolupament del codi.
  A més, per la part de disseny (la qual nosaltres no tocarem massa), és possible que sigui utilitzat amb grans programes com el 3d Studio Max o el mateix Photoshop.
  És una bona opció a utilitzar ja que la part de disseny, a nivells bàsics, és molt intuïtiva i el codi implementat es pot complementar amb opcions i seleccions bastant senzills d'utilitzar.

- Unreal → Aquest és un motor de joc dedicat als ordinadors i consoles. Aquesta eina utilitza, bàsicament, el llenguatge C++ i és un motor que des d'un inici es va crear per a jocs en primera persona tot i que també s'utilitza en jocs en tercera persona.
  Descarto aquesta opció ja que, si es donés el cas, voldria adaptar el meu videojoc per a plataformes Android.

- GameMaker → Aquesta eina està basada en el llenguatge de programació Delphi i és una bona eina si no es sap programar. A més dels scripts per a poder personalitzar els codis implementats, utilitza un tipus de "programació" per caixes on, simplement arrastrant i deixant anar per la interfície es van generant una sèrie d'esdeveniments.

  Descarto aquesta eina.

- Cocos2d → És un framework on es codifica amb llenguatge Python. També incorpora un motor i va destinat a les plataformes PC, Xbox, Linux, web, Android i iOS. Té un editor d'animació incorporat el qual treballa amb sprites.

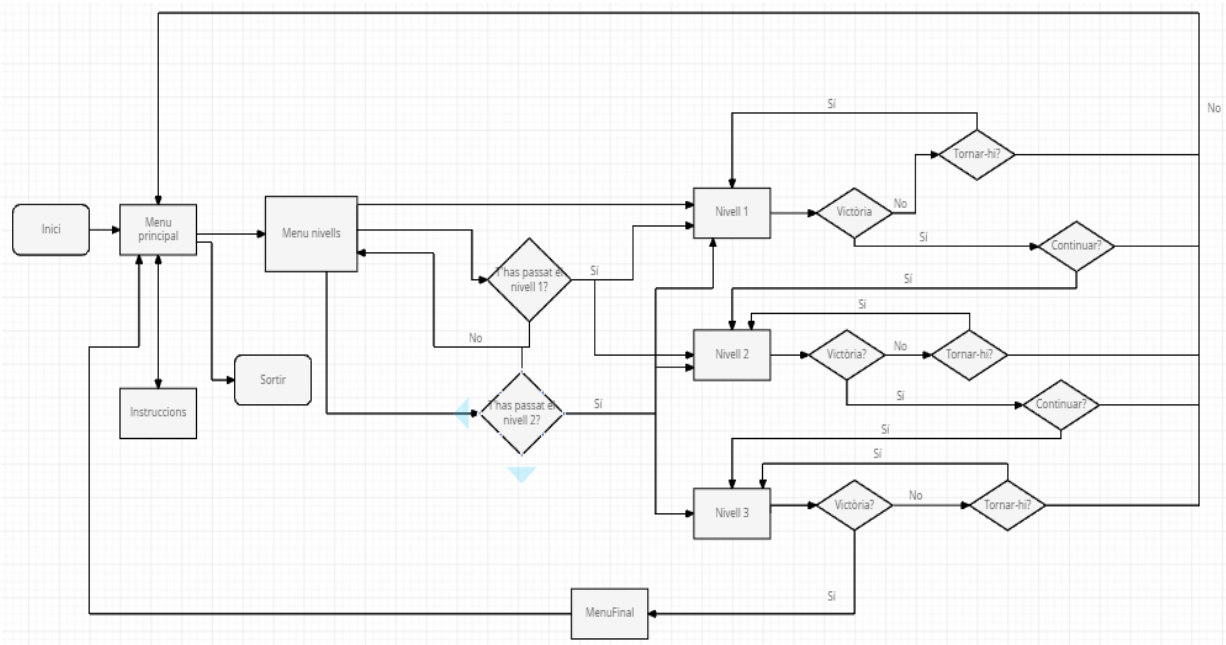
  Tot i que el llenguatge bàsic és Python, es poden afegir altres llenguatges de programació com poden ser el JavaScript o el C++.

  També és una bona opció de cara al joc que intento desenvolupar.

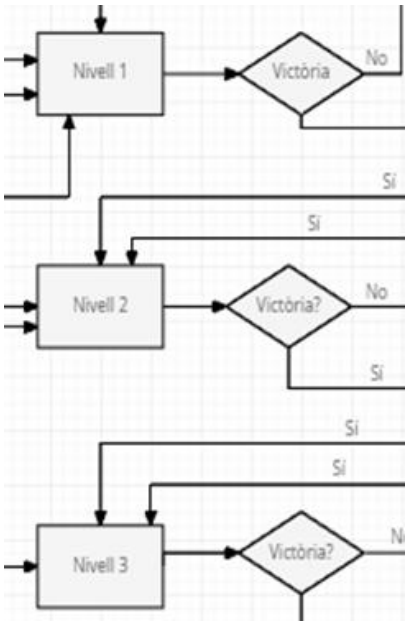
En conclusió, el programa que utilitzaré és el Unity ja que és el que m'ofereix, per al meu tipus de videojoc, les millors característiques. A més, és també un entorn amb un llenguatge de programació que personalment tinc molt interès en aprendre.

3. Arquitectura del joc

El diagrama de fluxos que presenta el joc és el següent:



*Cal dir que, a mesura que el joc s'actualitzi i s'afegeixi un nivell més, el fluxe anirà creixent cap avall de la mateixa manera:



4. Implementació

4.1 Classes

CreadorEnemics

Aquesta classe és l'encarregada de crear els enemics sobre el terreny de joc. Controla també la distància en la que apareixen els enemics entre ells i, a més, és l'encarregat de veure si tenim o no enemics sobre el terreny de joc, o bé si ja no hi ha més rondes en el nivell actual en el qual ens trobem.

En aquesta classe trobem la funció:

- **CrearOleada()**, la qual retorna el temps entre els enemics de la mateixa ronda i actualitza la ronda un cop acabada.

També tenim el mètode:

- **CrearEnemic(GameObject enemic),** que instancia cadascun dels enemics que es van creant dins la ronda.

Enemics

Classe que defineix les característiques bàsiques dels enemics com són el moviment, la vida i els diners que deixa al morir. En el seu **Update()** calculem les coordenades de tal forma que si arriba a un punt de ruta anirà a buscar el següent. També mira que la vida no sigui zero o inferior per saber si està viu o mort.

Dins de la classe trobem els mètodes:

- **SeguentPunt()**, el qual retornarà el següent punt amb el canvi de direcció de l'objecte si s'escau o bé l'eliminarà si el punt de ruta és l'últim.
- **RestarVida()**, per a restar la vida de l'enemic un cop rebi un atac.

EscollirNivell

Aquesta és la classe que controla quins nivells es poden o no jugar. Els nivells queden guardats en un array de Buttons el qual anomeno **nivells**.

L'únic mètode que té aquesta classe és el de **Eleccio()** per a poder carregar el nivell que li passem, però el que és més interessant és la variable **progres**, que

és on es guarda el número de nivell pel qual anem i, d'aquesta manera, desactivar els nivells superiors.

Fi

Aquesta és una classe molt simple la qual s'utilitza quan ens derroten els enemics en alguna de les rondes dels nivells.

Tenim els mètodes:

- **TornarHi()**, que recarrega el nivell que ens trobem reiniciant així tots els valors a com eren inicialment.
- **Sortir()**, que serveix només per a carregar l'escena del menú principal.

Instruccions

Classe per a poder parar el temps en la pantalla d'instruccions del joc i tirar enrere altra cop al menú principal.

Aquesta classe només té el mètode **Sortir()**, que serveix per a carregar l'escena del menú principal i reinicialitzar el temps. El fet que hagi decidit parar el temps, ha estat perquè hi havia un *bug* el qual els botons del menú principal es podien clicar tot i estar a la pantalla de les instruccions. D'aquesta manera, a l'aturar el temps, els botons quedaven inservibles.

Jugador

En aquesta classe es defineixen les característiques principals del jugador, així com les variables de la UI. Per una banda tenim definits els diners, els punts, les vides, la ronda actual i el següent nivell a l'actual i, per altra banda, el text d'aquests paràmetres per a poder ensenyar-los visualment en la UI mentre juguem. Aquesta classe, també, inicia el paràmetre de la velocitat de refresc dels frames.

Hi trobem dues funcions:

- **QuantsDiners()**, que retornarà els diners per saber si podem o no comprar una torre.
- **Esperar(int r)**, que s'utilitza per afegir un *delay* al temps en el qual el número de la ronda canvia.

D’altra banda tenim els mètodes:

- **RestarVides(int vida)**, que resta les vides del jugador.
- **RestarSumarDiners(int monedes)**, que suma o resta els diners en funció de si gastem en una compra o rebem d’una venta o mort de l’enemic.
- **SumarPunts(int punt)**, que suma els punts totals del jugador.
- **SumarRondes(int r)**, que ens llançarà l’animació del canvi de ronda.
- **Victoria()**, que ens aturarà el temps i ens obrirà la pantalla de victòria o, si és el final del joc, la pantalla de final de joc. També actualitzarà la variable **progres** que, com he esmentat abans, determina per quin nivell vas i desbloquejar el menú de nivells.
- **EliminarObjectes()**, que eliminarà tots els objectes creats sobre el terreny de joc gràcies als *tags* que tenen. Això vaig decidir fer-ho perquè podíem trobar algun *bug* tipus que apareixia la pantalla de victòria o derrota i el menú de torre estava seleccionat, o bé que un enemic podia sobresortir per sobre del text.

Menu

Aquesta és la classe que té el menú principal simplement determinar si juguem o tanquem el joc.

Els mètodes són:

- **Jugar()**, que carregarà la pantalla de selecció de nivells.
- **Sortir()**, que tancarà el videojoc.

ModificadorTemps

Classe que determina quin és el temps de refresc dels *frames* i utilitzat en la UI.

Els mètodes creats per aquesta classe són:

- **Pausa()**, canvia el temps de refresc dels *frames* del joc a 0, donant la sensació de pausa.
- **Continua()**, canvia el temps de refresc a 0.6, que és el refresc de temps normal del joc.
- **AnarRapid()**, canvia el refresc a 1, que és més ràpid que 0.6 i, per tant, dona sensació d’anar a càmera ràpida.

Oleada

Aquesta classe és una mica diferent a les demés ja que s'utilitza una API especial per a poder tenir propietats en l'inspector. Les propietats són el número d'enemics que apareixeran en aquella ronda concreta, el temps entre els enemics que apareixen en la mateixa ronda i, finalment, el tipus d'enemic que apareix. Hi havia la possibilitat d'utilitzar llenguatge un fitxer **JSON** per a definir com havien de ser els nivells, però he trobat que fer-ho d'aquesta manera, aprofitant els mateixos recursos del **Unity**, era més senzill i fàcil d'implementar.

Projectil

Aquesta classe és on es defineixen les característiques principals dels projectils com la velocitat o el mal que fan sobre els enemics. Dins de la classe haurem de fer un quadrant amb la distància que hi ha entre l'objectiu enemic i el propi projectil per a poder determinar si l'hem tocat o no.

Aquesta classe té 3 mètodes:

- **SetMal(float malPassat)**, que determina el mal que fa en funció de la torre que hagi creat el projectil.
- **BuscarEnemic(GameObject enemic)**, que determina quin és l'objectiu enemic al qual es dirigirà.
- **ObjectiuAconseguit()**, que restarà la vida de l'enemic en funció dl mal que té el projectil i eliminarà l'objecte del terreny de joc.

RajolaTorre

Aquesta és la classe més complexa de totes. És on es troba el *core* del funcionament de la construcció de defenses del joc. Aquí definim, també, certes característiques de les torres com els diners que costen, el tipus que construïm o que hi ha construïda, així com tots els textos per a poder referenciar els menús de construcció amb els valors que guardem de la classe.

A l'hora de construir mira quina és la tecla que hem premut, i en funció de paràmetres com els diners del jugador o de si hi ha torre o no i quina, et deixar fer unes coses o altres.

En aquesta classe tenim dos mètodes, a més de l'**Update()**:

- **OnMouseDown()**, que recull la funcionalitat que ha de dur a terme en cas que premem el ratolí sobre una rajola d'herba. Primer mirarà que no hi hagi un altra menú obert (en cas que l'hi hagi el tancarà) i, posteriorment, obrirà un menú o altre en funció de si hem clicat en una rajola sense torrem amb una torre de nivell 1 d'un tipus concret, o bé una torre de nivell 2.
- **Resetejar()**, que esborrarà el menú que tinguem obert i reiniciarà els paràmetres on es guarda si hi ha un menú obert, quin menú estava obert i el temps que triga un menú a tancar-se automàticament.

Ruta

La classe ruta serveix per a guardar una llista amb els diferents punts de ruta ordenats amb les seves posicions per a poder dir als objectes enemics cap a on han d'anar. Aquesta classe no tenim cap mètode ni funció més que el **Awake()** per a guardar i inicialitzar les variables abans no es carregui el joc.

Sang

Aquesta és una de les classes més simples implementades en el videojoc. L'únic que fa és destruir l'objecte sang que es crea al matar a un enemic per tal que no s'acumulin i poder arribar a saturar els recursos de la màquina on s'executa el videojoc.

Torre

Classe on es recullen totes les característiques bàsiques d'una torre; el rang, la velocitat d'atac i el mal. També determina l'objectiu enemic i quin tipus de projectil invoca.

Un altra afegit d'aquesta classe és que per a determinar quin és el seu objectiu mira el que té més a prop dins del seu rang.

Tenim un mètode:

- **Dispar()**, que instancia el projectil pertinent segons el tipus de torre que és, executa també el soroll de dispar que li pertoca i li passa el mal que ha de fer, així com l'objectiu a la classe de projectil que ha instanciat.

Victoria

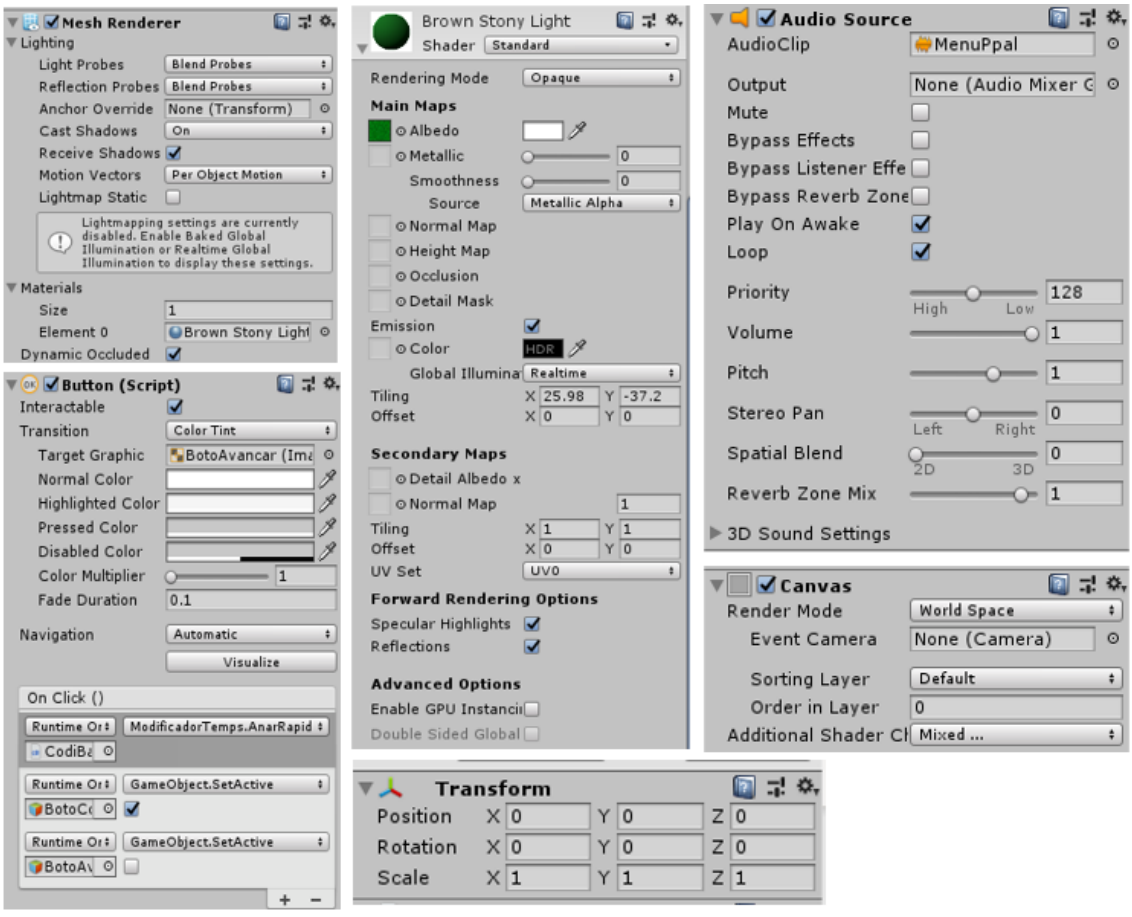
Aquesta classe s'utilitza un cop ens hem passat el nivell.

Els mètodes utilitzats són:

- **Seguent()**, que carregarà el següent nivell.
- **Sortir()**, que carregarà el menú principal del videojoc.

4.2 APIs

El programa Unity, a més d'oferir la possibilitat de combinar disseny amb programació, ofereix una sèrie de APIs que són molt senzilles d'utilitzar i les quals et permeten, d'una manera molt intuïtiva, aplicar diferents característiques als objectes:



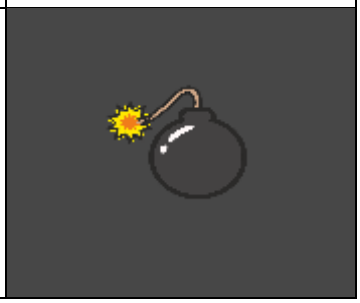
Aquests són alguns dels exemples gràfics de les APIs que ofereix Unity, a més de tenir-ne, també, altres purament de codi.

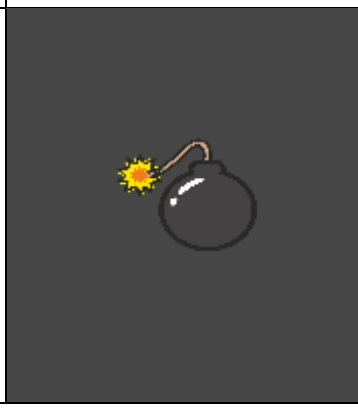
5. Actors

5.1 Torres

Tipus	Normal N.1
Descripció	Atac i velocitat normals
Cost	5
Rang de visió	16
Velocitat d'atac	0.9
Mal	30
Projectil	

Tipus	Normal N.2
Descripció	Nivell avançat de la torre normal
Cost	45
Rang de visió	19
Velocitat d'atac	0.6
Mal	50
Projectil	

Tipus	Forta N.1
Descripció	Torre lenta però amb mal i rang de visió elevats
Cost	15
Rang de visió	20
Velocitat d'atac	1.5
Mal	80
Projectil	

Tipus	Forta N.2
Descripció	Nivell avançat de la torre forta
Cost	90
Rang de visió	25
Velocitat d'atac	1.2
Mal	100
Projectil	

Tipus	Ràpida N.1
Descripció	Torre amb un mal i rang de visió reduïts però amb molta velocitat d'atac
Cost	20
Rang de visió	13
Velocitat d'atac	0.5
Mal	18
Projectil	

Tipus	Ràpida N.2
Descripció	Nivell avançat de la torre ràpida
Cost	120
Rang de visió	15
Velocitat d'atac	0.3
Mal	30
Projectil	

5.2 Enemics

Tipus	Bàsic
Descripció	Enemic amb un nivell de vida i velocitat normals
Moviment	10
Vida	170
Diners	2

Tipus	Ràpid
Descripció	Enemic amb un nivell de vida baix però velocitat ràpida
Moviment	10
Vida	170
Diners	2

Tipus	Fort
Descripció	Enemic amb un nivell de elevat però velocitat lenta
Moviment	10
Vida	170
Diners	2

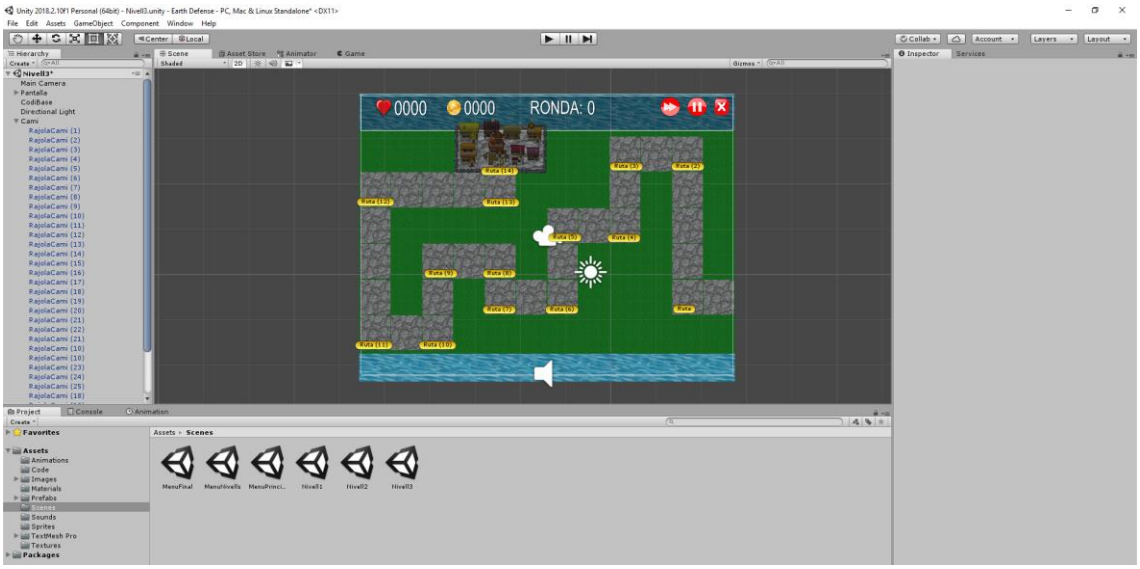
5.3 Poble

Descripció	El poble significa el punt defensiu que s'ha de protegir així que ni juga cap paper ni té cap característica. Podríem dir que és simple estètica per saber quin és el punt final del recorregut dels enemics.

6. Mapa

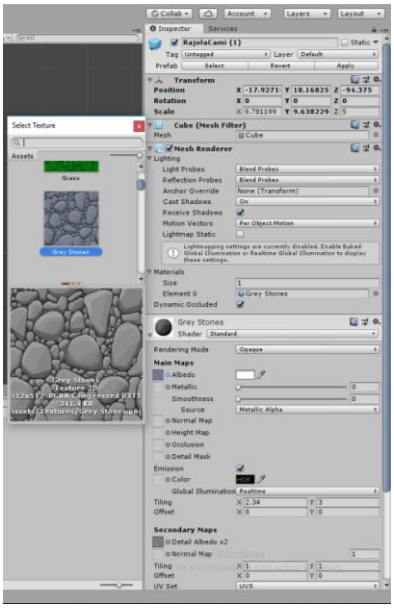
Rajola de tipus camí

Aquest tipus de rajola són el camí que els enemics utilitzaran per a poder arribar al poble. Per a poder anar delimitant el recorregut dels enemics dins d'aquestes rajoles, a més, he utilitzat punts de ruta per determinar, cap a quina direcció han d'anar els enemics:



Com es pot veure en la imatge, els punts de ruta segueixen un ordre cardinal per a poder ordenar adequadament el camí a seguir.

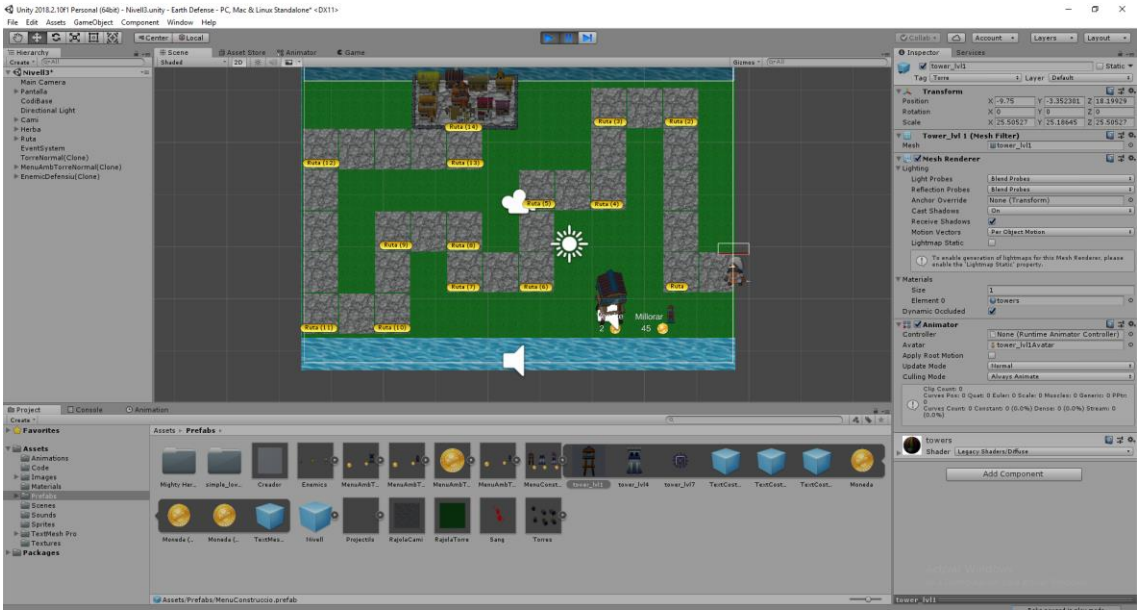
A més, per a poder donar-li un toc de camí, a sobre del cub li he aplicat una textura de roques.



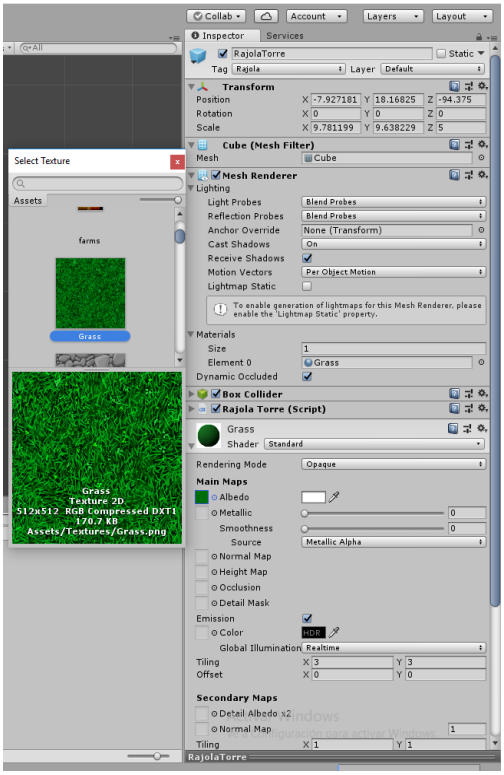
Rajola de tipus herba

Aquest altra tipus de rajola és el que li servirà al jugador per a interactuar amb el mapa. Clicant sobre aquestes rajoles apareixerà el menú de creació de torres i, sobre les mateixes, es construiran les defenses que el jugador hagi seleccionat.

A més, un cop hi hagi una torre ja construïda donarà la possibilitat de vendre-la o millorar-la amb un nou menú de construcció.

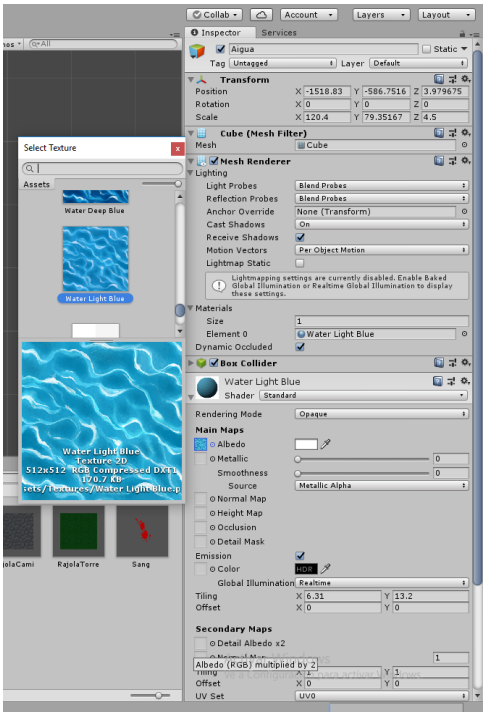


A aquest tipus de rajola, a més, li he aplicat una textura d'herba per donar la sensació de prat.



Rajola de tipus Aigua

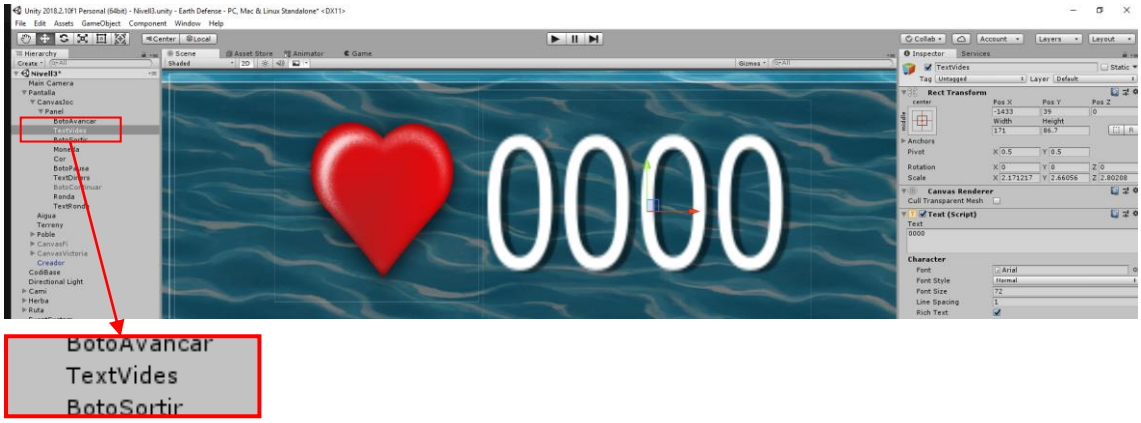
Aquesta rajola no té cap funcionalitat dins del joc, és purament visual. Li he afegit la textura d'aigua per a donar l'efecte de mar.



7. User Interface

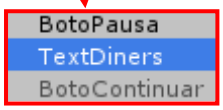
Vides

Amb el cor es fa una clàssica referència a les vides que té el jugador. El número de vides està referenciat pel camp **TextVides**, que recull quantes vides té actualment el jugador en el codi i, d'aquesta manera, refrescar en temps real el valor.



Diners

Amb la moneda fem una referència als diners que té el jugador. La quantitat de diners està referenciada pel camp **TextDiners**, que recull quants diners té actualment el jugador en el codi i, d'aquesta manera, refrescar en temps real el valor.



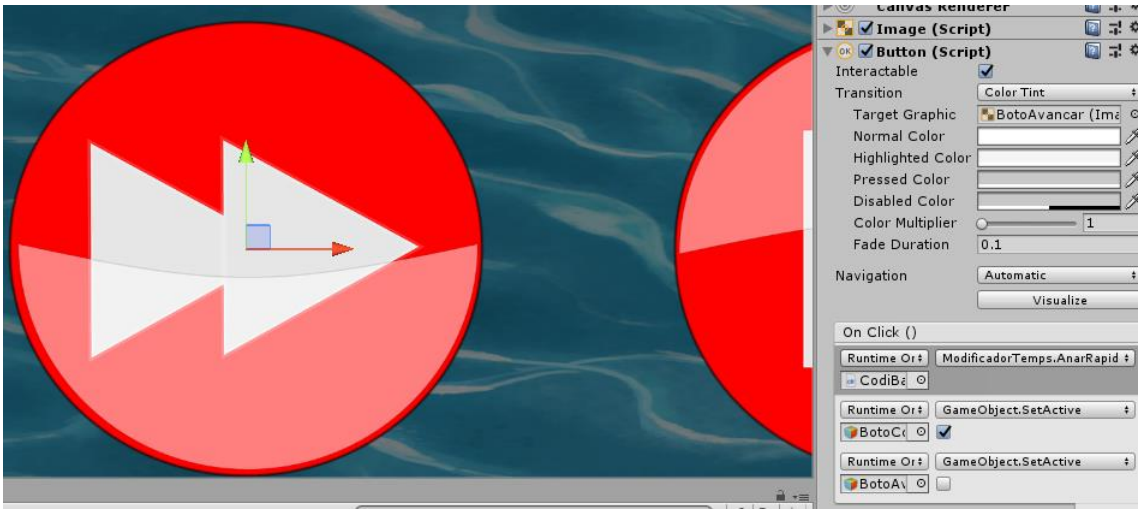
Rondes

Text on podem veure la ronda del nivell en la qual es troba. El mecanisme és idèntic al de les vides i els diners, la ronda està referenciada pel camp **TextRonda**, que recull per quina ronda va el jugador en el codi i, d'aquesta manera, refrescar en temps real el valor.

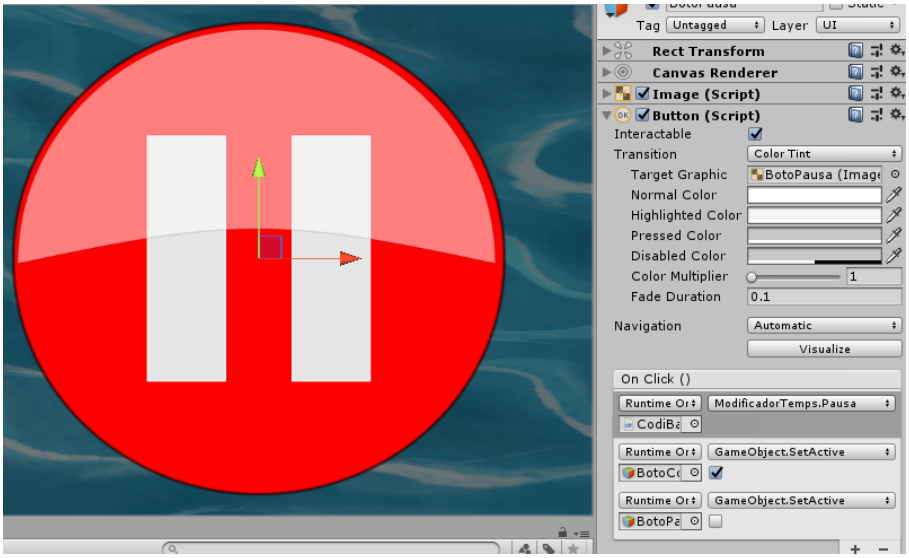
Botons d'usuari

Aquests botons ens serviran per a poder pausar o reprendre el joc, canviar-li la velocitat o bé sortir al menú principal.

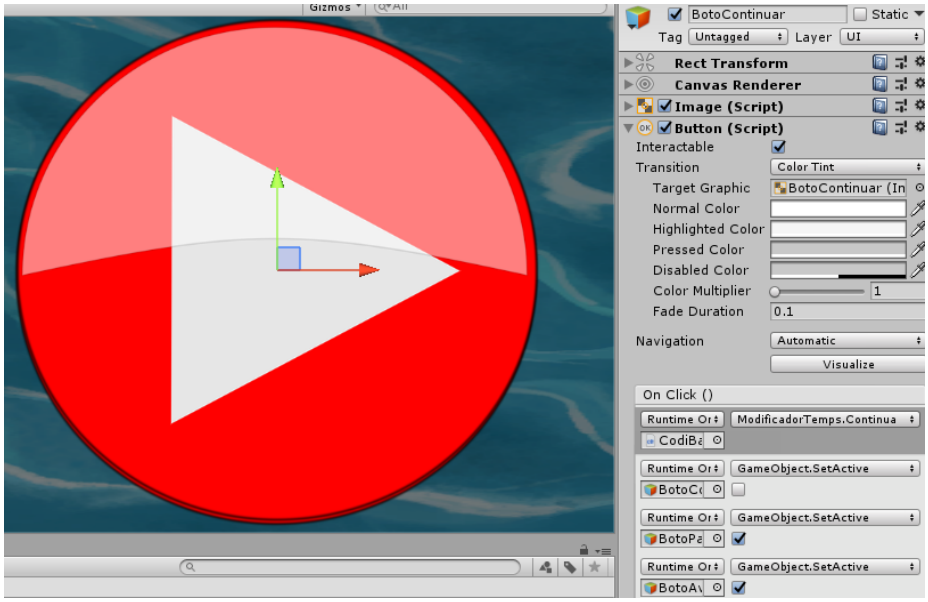
- Botó avançar → serveix per a fer que el temps vagi més ràpid. Al prémer aquest botó es cridarà al mètode AnarRàpid() per a modificar el temps i, a més, desapareixerà la imatge d'aquest botó per a fer aparèixer la imatge de tornar el temps a la normalitat.



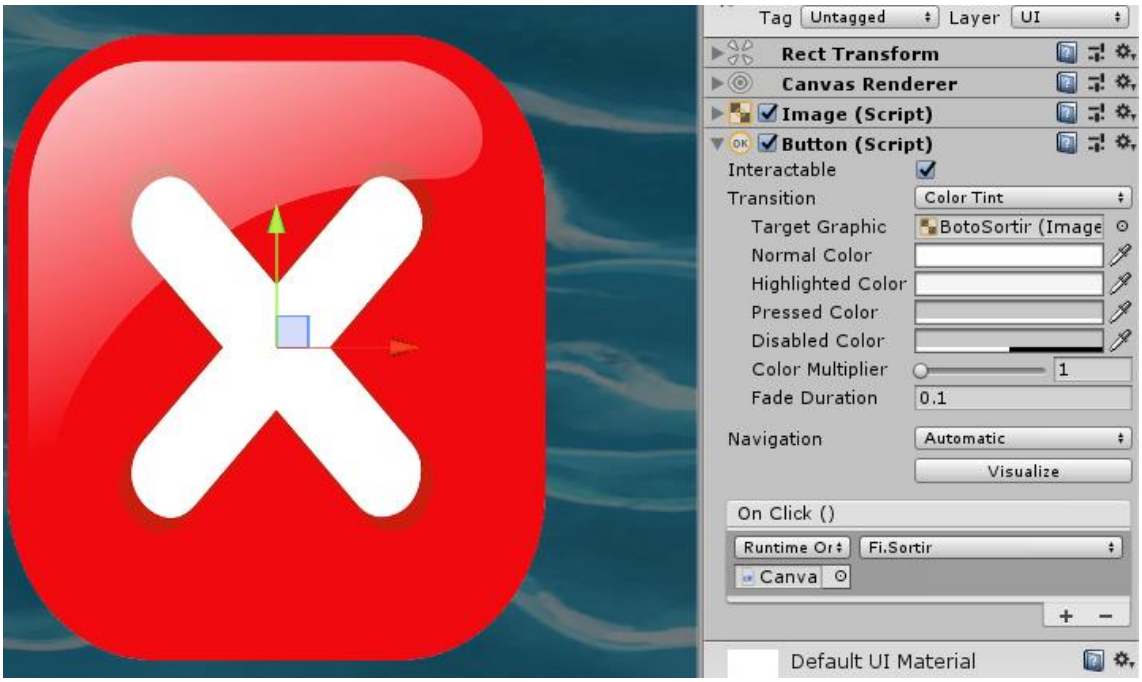
- Botó de pausa → Gràcies a aquest botó aturarem el joc. Aquest botó cridarà al mètode Pausa() que aturarà el temps i, igual que el botó d'avançar, desapareixerà la imatge d'aquest botó per a fer aparèixer la imatge de tornar el temps a la normalitat.



- Botó de continuar → Aquest botó només apareixerà en cas que la velocitat estigui avançada (en aquest cas farà que la velocitat sigui la normal) o haguem pausat el joc (en aquest cas el reprendrà). En aquest cas, el mètode cridat serà el de Continua() i, posteriorment, farà desaparèixer aquesta imatge per a fer reaparèixer la que hi havia prèviament.



- Botó per sortir →Amb aquest botó podrem sortir del nivell en el qual ens trobem per anar directament al menú principal de joc. En aquest cas el botó l'unic que farà serà cridar el mètode Sortir() el qual ens carregarà l'escena del menú principal.

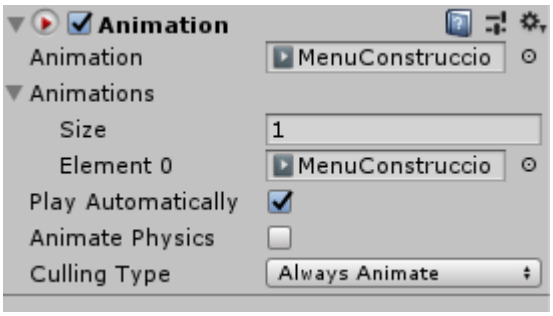


8. Animacions

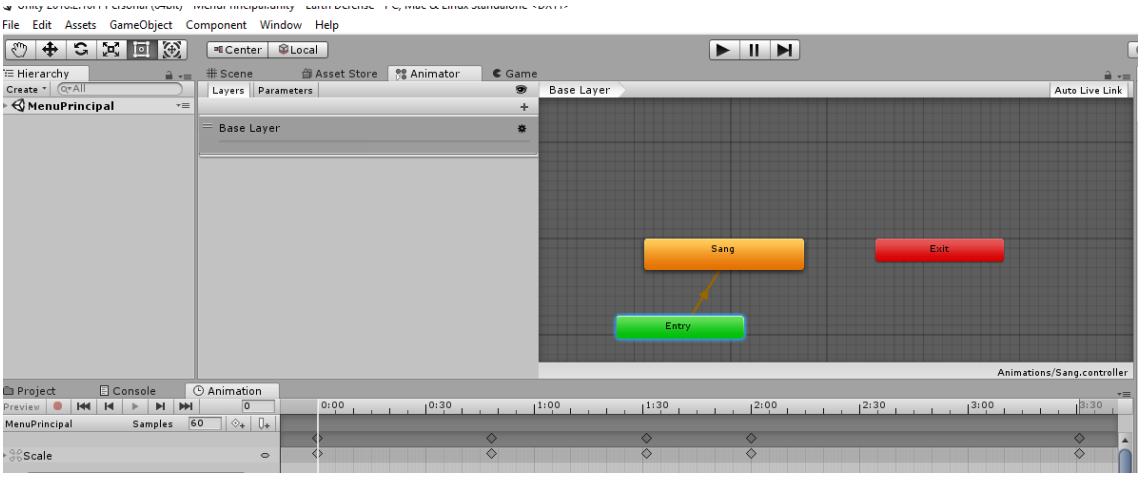
Per a donar-li un toc més fluid i vistós al joc, he creat una sèrie de petites animacions on, la majoria, són petits canvis d'escalatge dels objectes però suficient per fer el joc una mica més atractiu:

- **CanviRonda** → Aquesta animació emfatitza visualment l'apartat de les rondes de la UI. Fent un petit canvi d'escalatge i una rotació, canvia el número de l ronda un cop ens la passem.

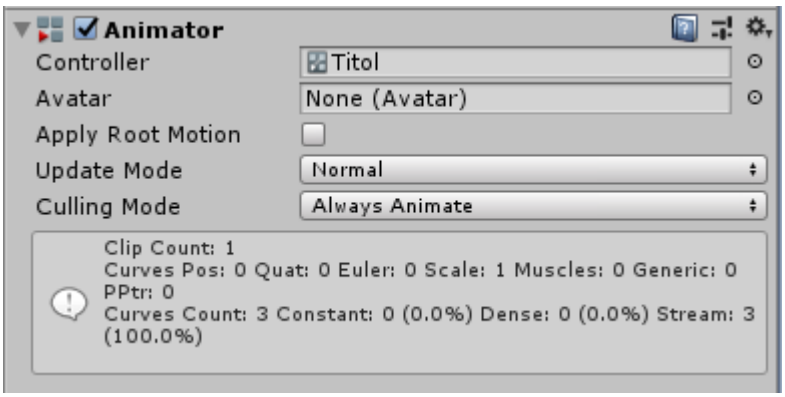
- **MenuConstruccio** → Animació a l'obrir-se el menú de construcció quan es clica sobre una rajola d'herba o una torre. L'únic que varia és l'escalatge a més gran.



- **Sang** → Aquesta és una animació per donar el feedback a l'usuari avisant que ha matat a un enemic. És una animació que fa que el toll de sang, un cop apareix, es vagi reduint poc a poc netejant el terreny de joc.



- **Titol** → Animació que se li dona a les lletres de *EARTH DEFENSE* a la pantalla del menú principal del joc. És un simple canvi d'escalatge però fer la pantalla del menú menys estàtica.

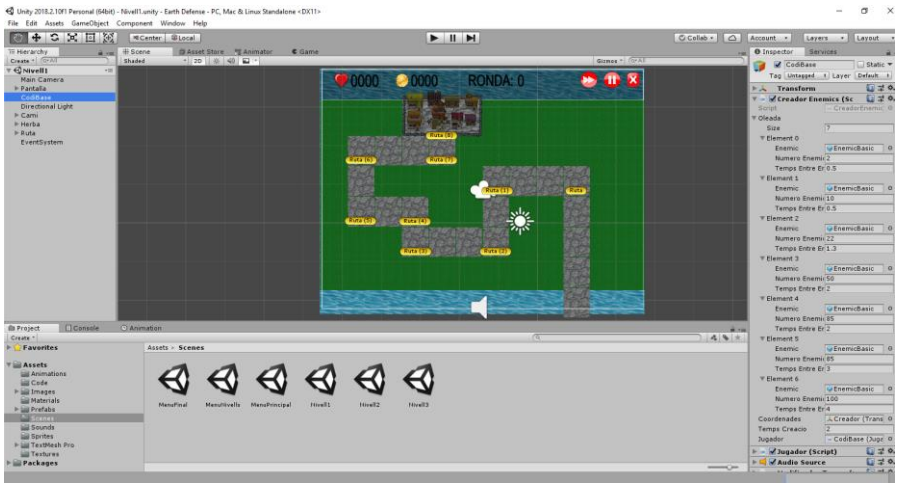


9. Nivells

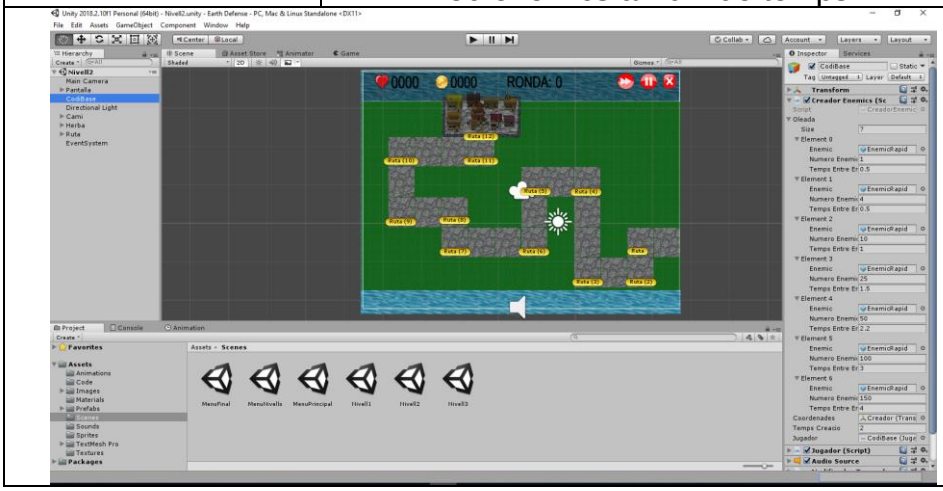
Earth Defense, en l'actualitat, consta de 3 nivells diferents. La idea és que en un projecte continuat es vagin actualitzant la quantitat de nivells amb diferents tipus d'enemics i afegint dificultat a mesura que el nombre de nivells creix.

La forma de crear nous nivells és bastant senzilla, ja que només s'han de disposar els cups d'una manera diferent (actualitzant els punts de ruta) i amb la classe **Oleada()** que crida **CreadorEnemics()** escollir quantes rondes es volen crear, arrastrar el *prefab* de l'enemic corresponent i escollir el temps entre enemics.

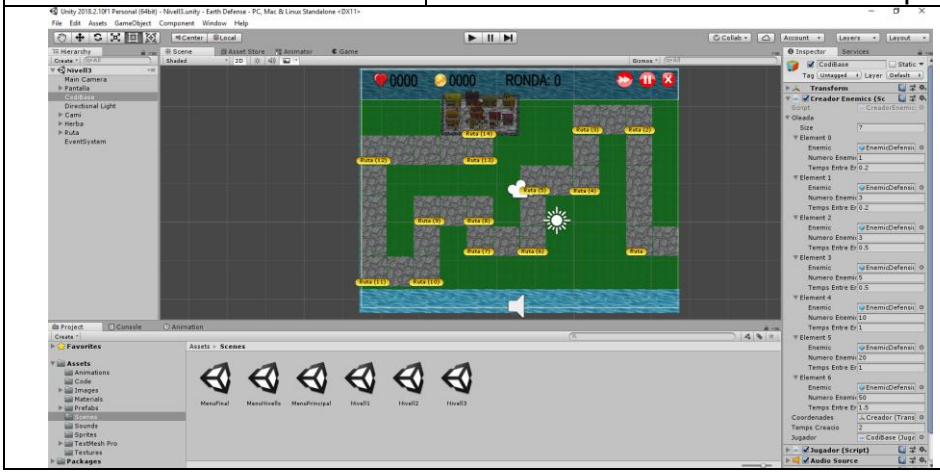
Nivell 1	
Vides Inicials	5
Diners Inicials	10
Tipus d'enemics	Bàsic
Rondes del nivell	1. 2 enemic amb 0.5 de temps 2. 10 enemics amb 0.5 de temps 3. 22 enemics amb 1.3 de temps 4. 50 enemics amb 2 de temps 5. 85 enemics amb 2 de temps 6. 85 enemics amb 3 de temps 7. 100 enemics amb 4 de temps



Nivell 2	
Vides Inicials	5
Diners Inicials	15
Tipus d'enemics	Ràpid
Rondes del nivell	1. 1 enemic amb 0.5 de temps 2. 4 enemics amb 0.5 de temps 3. 10 enemics amb 1 de temps 4. 25 enemics amb 1.5 de temps 5. 50 enemics amb 2.2 de temps 6. 100 enemics amb 3 de temps 7. 150 enemics amb 4 de temps



Nivell 3	
Vides Inicials	5
Diners Inicials	50
Tipus d'enemics	Fort
Rondes del nivell	1. 1 enemic amb 0.2 de temps 2. 3 enemics amb 0.2 de temps 3. 3 enemics amb 0.5 de temps 4. 5 enemics amb 0.5 de temps 5. 10 enemics amb 1 de temps 6. 20 enemics amb 1 de temps 7. 50 enemics amb 1 de temps

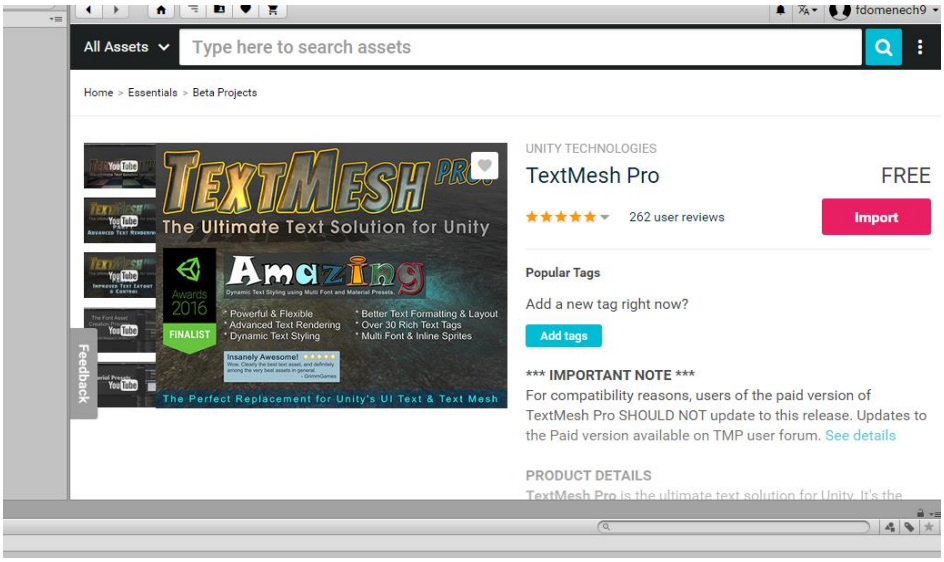


10. Assets

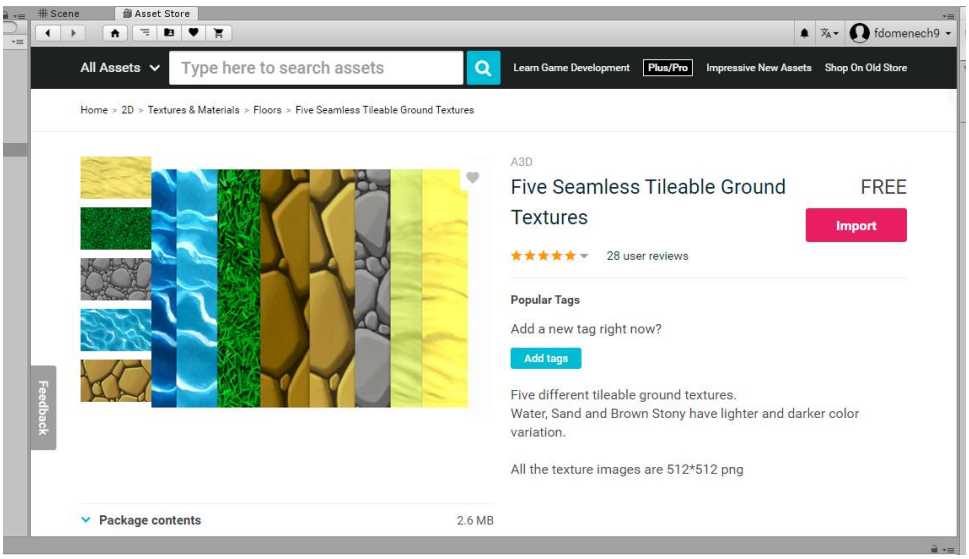
Per a poder dur a terme l'aspecte visual del joc he hagut d'utilitzar una sèrie d'assets descarregat mitjançant la mateixa asset Store que ofereix Unity. Cal dir que cap d'ells ha tingut cap cost ja que buscava sempre els gratuïts.

Els assets utilitzats han estat:

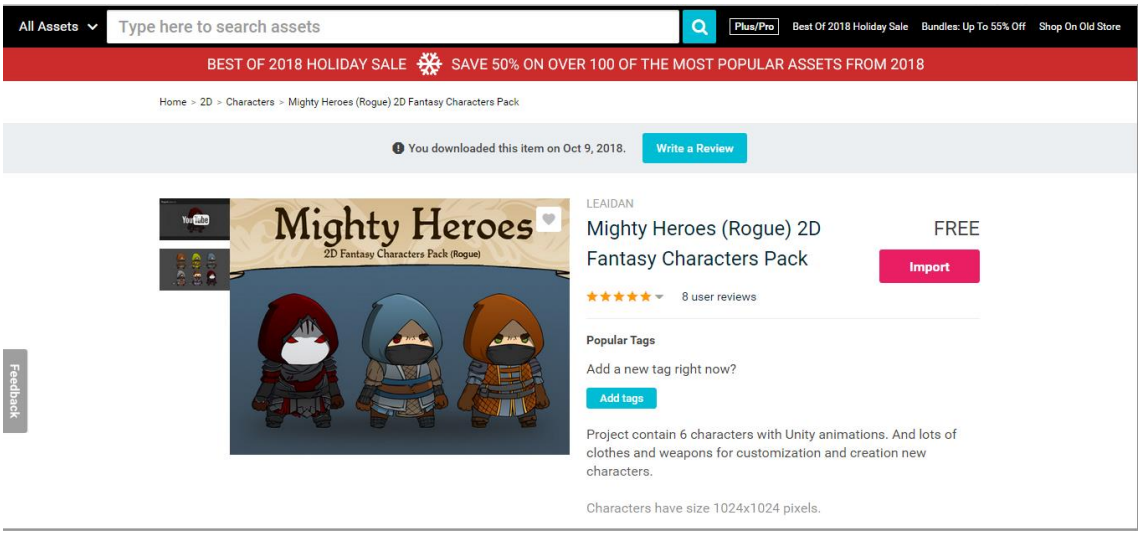
- TextMesh Pro → Utilitzat per a crear textos més vistosos, he utilitzat aquest asset per al títol o altres textos.



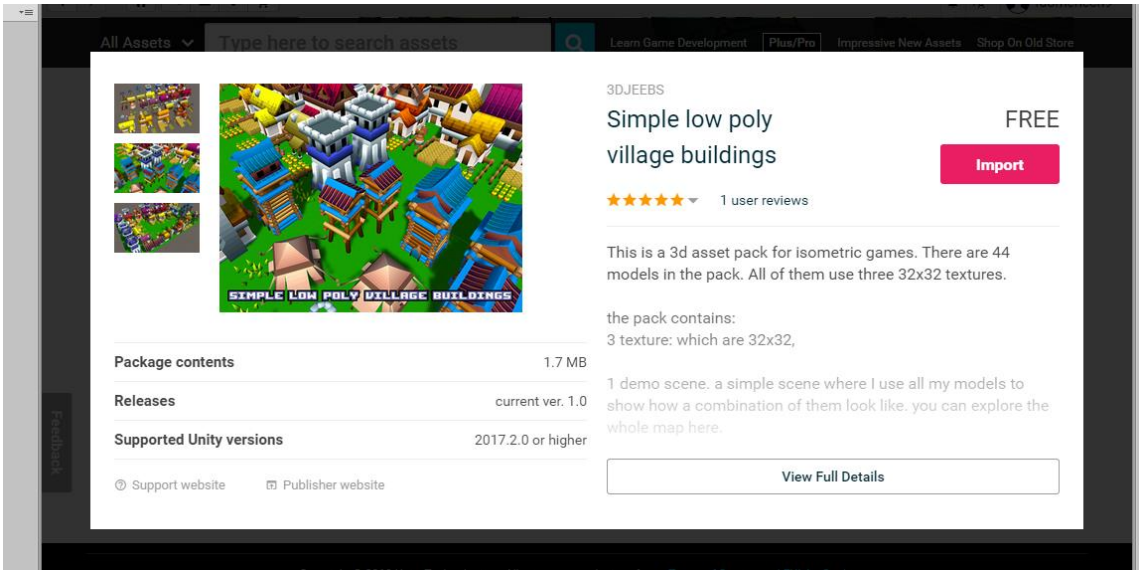
- Five seamless Tileable Ground Textures → Utilitzat per a crear les textures de l'herba o de les pedres del camí principal.



- Mighty Heroes (Rogue) 2D Fantasy Characters Pack → Aquest pack l'he utilitzat per a poder crear els 3 tipus d'enemics que hi ha en el joc.



- Simple low poly village buildings → Asset utilitzat per a tenir les diferents estructures que hi ha al joc, tant les torres defensives com els edificis del poble.



A més, m'he descarregat diferents músiques i efectes sonors per donar-li al joc més dinàmica:

- Fantasy Music Collection Lite → Utilitzat per a les músiques dels menús i dels nivells.

- Medieval Action -FX Pack 2.0 → Utilitzat per als efectes sonors dels projectils de les torres normals.

També m'he descarregat altres efectes sonors del banc de sons i imatges de l'estat per a tenir els sons dels projectils de les torres ràpida i forta i la imatge de la sang.

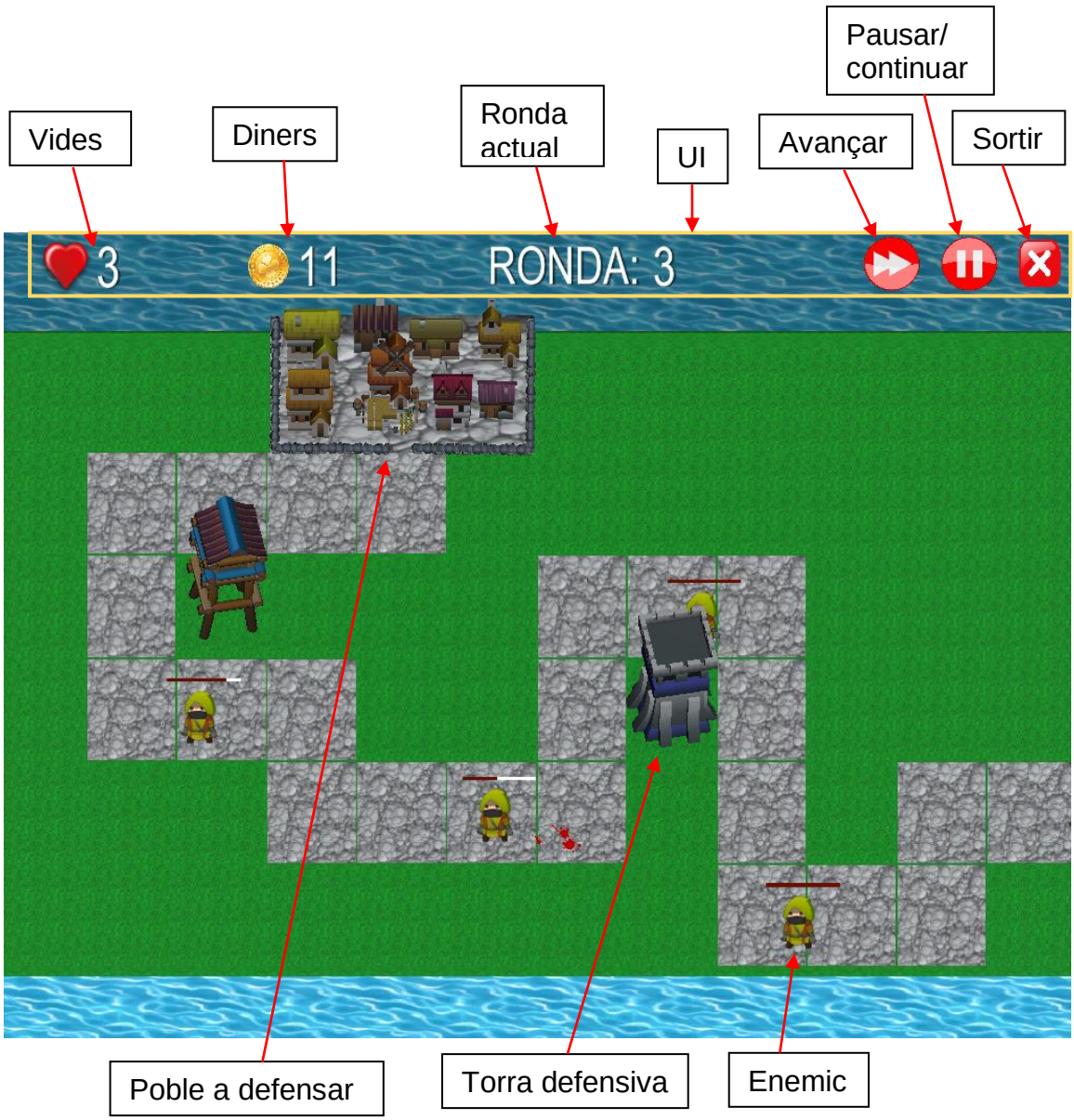
11. Versions

Versió	Alpha
Continguts	➢ Pantalla de menú principal ➢ Pantalla d'instruccions ➢ Tres tipus d'enemics ➢ Tres tipus de torres de primer nivell ➢ Tres tipus de torre de segon nivell ➢ Menú de construcció bàsic ➢ Un nivell ➢ Pantalla de derrota

Versió	Beta
Millores	➢ Menú principal més vistós ➢ Menú de nivells ➢ Tres nivells diferents ➢ UI avançada amb la possibilitat de canviar la velocitat del joc ➢ Efecte de sang ➢ Menú de construcció avançat ➢ Pantalla de victòria

12. Disseny de nivells

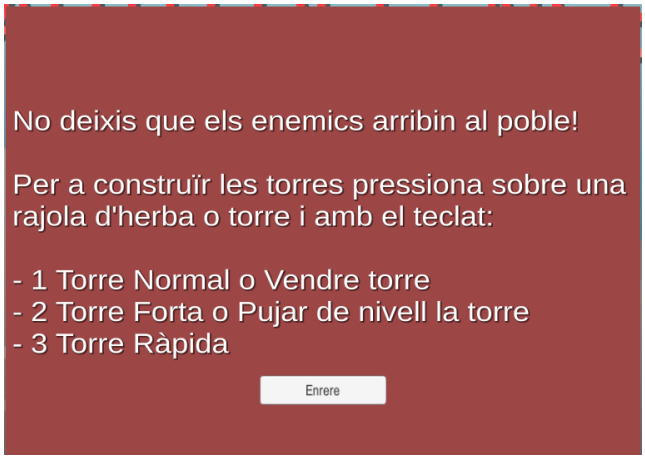
Les pantalles de *Earth Defense* segueixen la mateixa estructura, on la UI la tenim a la pantalla posterior de la pantalla en una zona que no molesti.



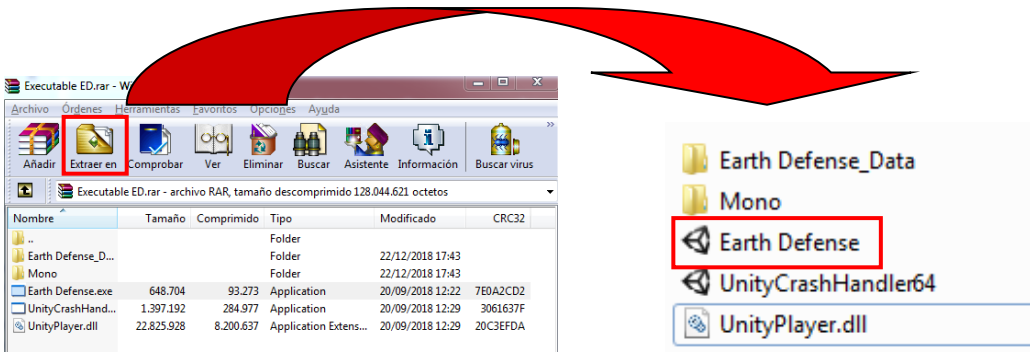
Aquest és l'aspecte que tenen els nivells dels jocs bàsic de tipus *tower defense*, on a més de la UI amb el mínim de les vides i els diners i un botó per a poder sortir del joc, el mapa consta amb almenys un recorregut dels enemics i uns espais de construcció.

13. Manual d'usuari

Els controls que utilitza aquest joc són mínims i s'expliquen en un menú d'instruccions. El jugador només haurà de clicar a sobre d'una rajola d'herba i, mitjançant els botons 1, 2 i 3 del teclat, decidir què fer.



Per a instal·lar el joc només cal descomprimir els arxius de Executable ED.rar i executar l'arxiu Earth Defense.exe.



Requisits mínims:

- Sistema operatiu Windows 7 SP1 o superior
- Tarja de vídeo amb capacitat pel DX10 o superior
- CPU compatible amb el conjunt d'instruccions SSE2

14. Experiència d'usuari

Per a poder posar punt i final al projecte i tenint en compte que la finalitat de qualsevol joc creat és que els usuaris el juguin, s'ho passin bé i, si s'escau, el comprin, s'ha realitzat una sèrie de tests a diferents persones. Posteriorment se'ls ha passat un qüestionari amb la finalitat de poder fer una valoració del videojoc.

Els usuaris que han provat el joc han estat:

Albert	
Edat	29
Ofici	Dissenyador
Valoració	El joc no té massa dificultat l'aspecte gràfic és molt millorable

Axel	
Edat	22
Ofici	Programador
Valoració	El joc és entretingut i estaria bé que hi haguessin més torres i enemics

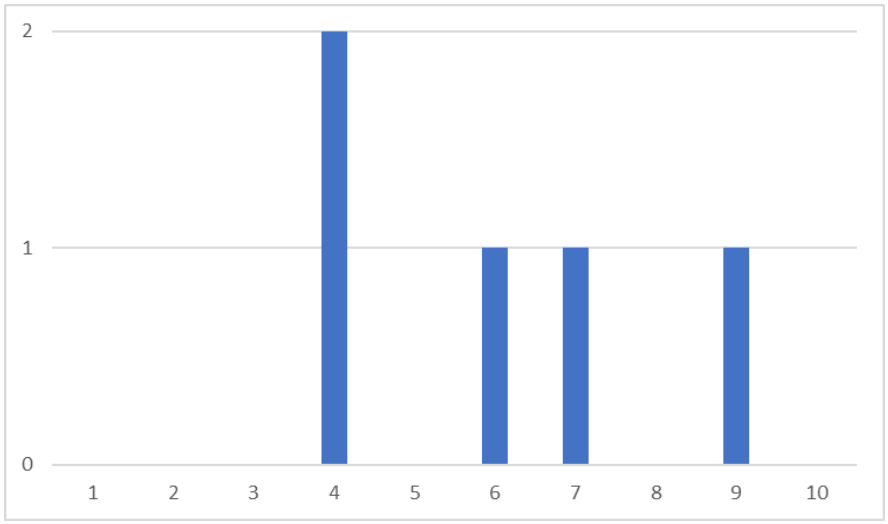
Carles	
Edat	20
Ofici	Estudiant d'informàtica
Valoració	Els controls del joc no són massa agradables, s'hauria de poder fer tot amb el ratolí

Joana Aina	
Edat	26
Ofici	Infermera
Valoració	Els colors i les lletres tant gran no m'agraden. Milloraria l'aspecte visual

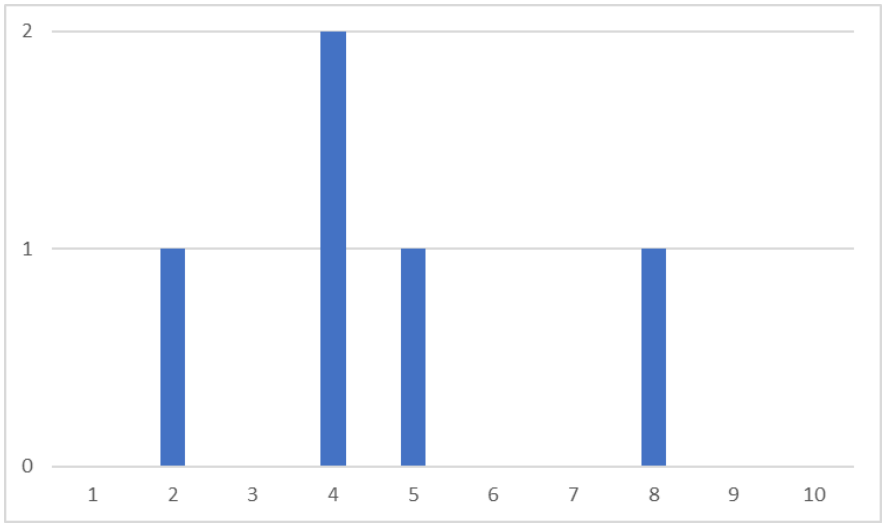
Mireia	
Edat	33
Ofici	Mestra
Valoració	El principi és avorrit però acaba “picant” a passar-te els nivells

Les preguntes i les respostes han estat les següents:

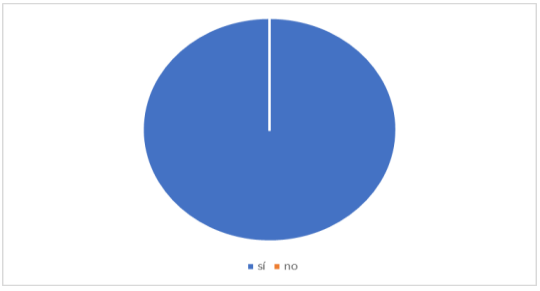
- Quin és el nivell de dificultat? (1-10, sent 1 més fàcil i 10 més complicat)



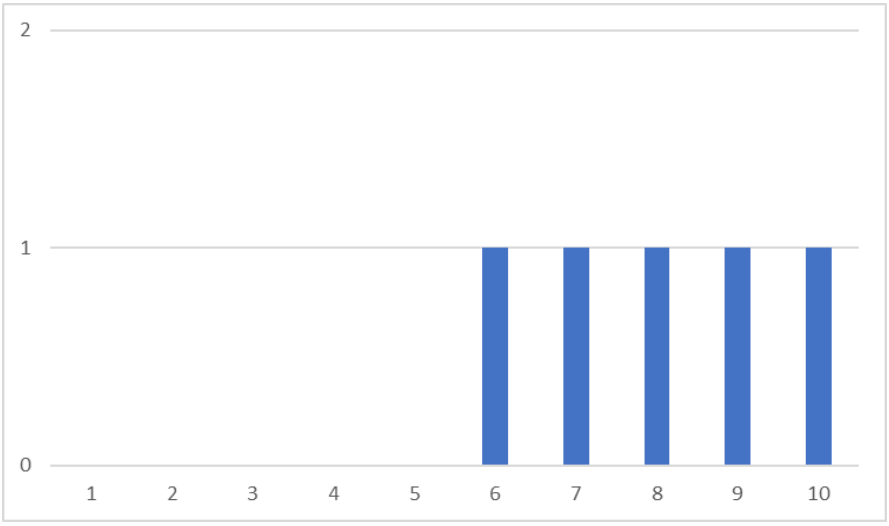
- Quina puntuació li donaries a l’estètica del joc? (1-10, sent 1 menys estètic i 10 més estètic)



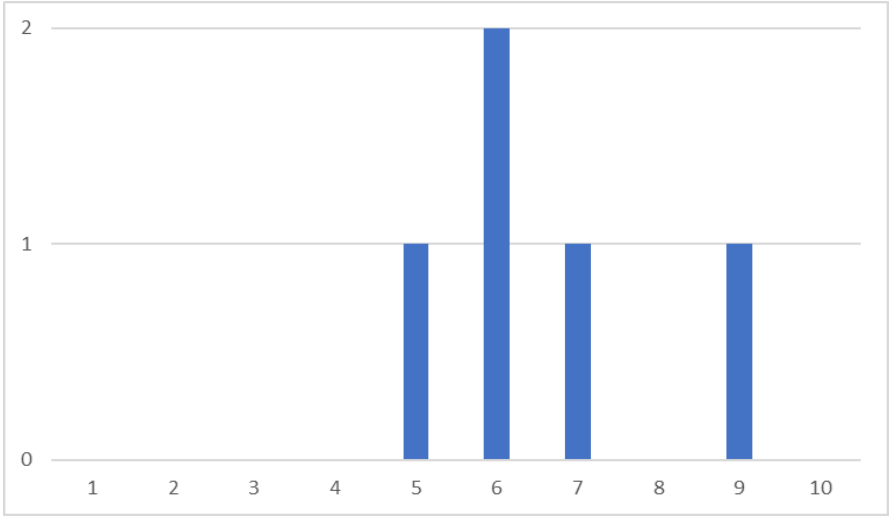
- En cas que s'actualitzessin nous nivells (amb possibles millores de torres), estaries disposat a continuar jugant-hi (sí/no)?



- Creus que el joc és auto-explicatiu? (1-10 sent 1 la més baixa i 10 la més alta)



- Quina puntuació general li dones al joc? (1-10 sent 1 la més baixa i 10 la més alta)



15. *Conclusions*

Gràcies a aquest treball i l'esforç que hi he dedicat m'ha donat una visió i uns coneixements sobre el motor de Unity i el llenguatge C# que abans no tenia. És per això, que tot i les hores dedicades i moltes frustracions que he patit pel camí estic molt content d'haver dut a terme aquest projecte i de la manera que ho he fet.

Cal afegir que m'hauria agradat disposar de més temps material per a poder dedicar-li més esforç al tema gràfic i haver personalitzat més el projecte, així com el seu atractiu visual.

A més, considero que he complert els objectius marcats inicialment. Crec que he assolit un mínim de coneixement de programació amb C# i utilització del Unity, que més enllà de tenir un producte més o menys complexa i/o elaborat, desitjava tenir des de feia molt de temps.

Tot i així, no he acabat de seguir tota la planificació amb tots els objectius interns del joc que em vaig marcar en un inici. Això ha estat degut a la complexitat de certes metes que he anat veient durant tot el procés del videojoc. Així mateix, he desenvolupat altres tipus d'objectius que no m'havia marcat inicialment però que tot i així m'han estat molt útils per a ampliar coneixements.

També vull remarcar que m'he quedat amb més ganes encara de continuar aprenent sobre aquest motor de videojoc i aquest llenguatge. És per això que m'he plantejat dur a terme el Master en disseny i programació de videojocs per assolir un nivell de coneixements superior i dur a terme projectes propis.

16. Glossari

APP → Aplicació informàtica que pot ser per a telefonia mòbil, web, tauletes, etc. És un programa que pot tenir diferents dissenys i/o finalitats per a usuaris que utilitzen un dispositiu electrònic.

API → Application Programming Interface. Conjunt de funcions i mètodes ja fets i emmagatzemats en una biblioteca per tal de reutilitzar-ho en diferents programes.

Bug → Error de funcionament d'una aplicació.

UI → Interfície d'usuari on es recull en els botons de menú que el jugador utilitza en un joc.

Asset → És una representació de qualsevol tipus d'ítem (recurs gràfic, àudio, imatge, etc.) que es carrega externa o internament al programa.

Prefab → És un tipus d'Asset que s'emmagatzema amb les característiques creades i, d'aquesta manera, tenir una plantilla d'un objecte. A més, si es canvia aquesta plantilla original, tots els objecte creats sobre l'escenari també canviaran automàticament.

SSE2 → *Streaming "Single Instruction Multiple Data" Extensions 2*. Conjunt d'instruccions per a la CPU de l'ordinador.

17. Bibliografia

1. https://answers.unity.com
2. https://www.appgamer.com/castle-defense/strategy-guide/tower-types
3. https://es.wikipedia.org/wiki/Unity_(motor_de_juego)
4. https://www.unrealengine.com/en-US/what-is-unreal-engine-4
5. http://www.cocos2d-x.org/
6. http://www.visualnovelty.com/
7. https://www.youtube.com/channel/UCYbK_tjZ2OrIZFBvU6CCMiA
8. https://www.youtube.com/channel/UCmtyQOKKmrMVaKuRXz02jbQ
9. http://recursostic.educacion.es/bancoimagenes/web/

18. Annexos

Codi font

CreadorEnemics.cs

```
using UnityEngine;
using System.Collections;

public class CreadorEnemics : MonoBehaviour
{

    public Oleada[] oleada;//posició de l'enemic
    public Transform coordenades;//lloc on neixerà l'enemic
    public float tempsCreacio = 5f;//temps entre oleadas
    public static int enemicsCreats;//enemics sobre el terreny de joc
    public Jugador jugador;// variable per referenciar la classe Jugador().

    private int numeroOleada;//número de oleada

    private void Start()
    {
        numeroOleada = 0;
        enemicsCreats = 0;
    }

    //En Cas que hi hagi enemics sobre el terreny de joc, no farà res. Si no,
mirarà que ja hagin passat totes les oleades i, si es dóna el cas, cridar
    //a la subrutina CrearOleada() i reinicialitzar el temps
    void Update()
    {
        if (enemicsCreats>0)
        {
            return;
        }
        if ((numeroOleada == oleada.Length) && (enemicsCreats == 0))//Si ja no
queden oleades ni enemics sobre el mapa, guanyarem el nivell.
        {
            jugador.Victoria();
        }
        if (tempsCreacio <= 0f) //si el tempsCreacio és 0 creem una oleada i
restablim el tempsCreacio
        {
            StartCoroutine(CrearOleada());

GameObject.Find("CodiBase").GetComponent<Jugador>().SumarRondes(numeroOleada +
1);
            tempsCreacio = 5f; //Reinicialitzem el temps enrere
            return;
        }
        tempsCreacio -= Time.deltaTime; //Resta un cada segon
    }

    //Amb la següent funció crearem els enemics guardats a l'array de la classe
Oleada
    IEnumerator CrearOleada()
    {
        Oleada ol = oleada[numeroOleada];
        for (int i = 0; i < ol.numeroEnemics; i++)
        {
```

```
            CrearEnemic(ol.enemic);
            yield return new WaitForSeconds(1f/ol.tempsEntreEnemics);//temps de
creació entre enemic i enemic que podem controlar des de fora
        }
        numeroOleada += 1;
    }

    //Amb aquest mètode instanciarem cada enemic
    void CrearEnemic(GameObject enemic)
    {
        enemicsCreats += 1;
        Instantiate(enemic, coordenades.position, coordenades.rotation);//cridem
a un enemic en les coordenades de creació
    }
}
```

Enemics.cs

```
using UnityEngine;
using UnityEngine.UI;

public class Enemics : MonoBehaviour {

    public float moviment;//Rapidesa de moviment de l'enemic
    public float vida;//Vida de l'enemic
    public int diners;//Diners que deixa caure l'enemic un cop mort
    public Image barraVida;//Barra de la vida dels enemics
    public GameObject sang;//Sang al morir

    private float auxVida;//Auxiliar pel càlcul de la barra de vida
    private int punt = 0; //Punt de ruta actual
    private Vector3 coordenades; //Cap a quines coordenades han d'anar els
enemics

    void Start () {
        coordenades = Ruta.ruta[0].position;
        auxVida = vida;
    }

       void Update () {

        //Primer determinarem, en coordenades, cap a quina direcció es mouen els
enemics.
        //A continuació, amb la funció predefinida Translate() mourem l'objecte
en funció de la variable direcció que hem calculat abans.
        //Finalment, si ens trobem a la vora de les coordenades, cridem a la
funcio SeguentPunt(). Això és així perquè, i no he trobat
        //el motiu, si coincidia l'objecte amb el punt exacte de les coordenades
generava un bug i no es movia d'allà.
        //En cas que la vida caigui completament, eliminarem l'objecte i sumarem
els diners que ens dóngui. Restem els enemics que hi ha.
        Vector3 direccio = coordenades - transform.position;
        transform.Translate(direccio.normalized * moviment * Time.deltaTime,
Space.World);
        if (Vector3.Distance(transform.position, coordenades)  < 0.5f)
        {
            SeguentPunt();
        }

        if (vida <= 0f)
        {
            sang = Instantiate(sang, transform.position, transform.rotation);
```

```
            Destroy(gameObject);

GameObject.Find("CodiBase").GetComponent<Jugador>().RestarSumarDiners(diners);
            GameObject.Find("CodiBase").GetComponent<Jugador>().SumarPunts(1);
            CreadorEnemics.enemicsCreats -= 1;
        }
    }

    //Amb el mètode SeguentPunt() escollirem el següent punt a l'actual i, en cas
que sigui l'ultim, destruirem l'objecte, restarem una vida i restarem un enemic.
    void SeguentPunt()
    {
        if (punt >= Ruta.ruta.Length - 1)
        {
            GameObject.Find("CodiBase").GetComponent<Jugador>().RestaVides(1);
            Destroy(gameObject);
            CreadorEnemics.enemicsCreats -= 1;
            return;
        }
        if (punt > 0)
        {
            if ((Ruta.ruta[punt + 1].position.x > Ruta.ruta[punt].position.x) &&
(Ruta.ruta[punt - 1].position.x > Ruta.ruta[punt].position.x))
            {
                gameObject.transform.Rotate(0, 180, 0);
            }
        }
        punt++;
        coordenades = Ruta.ruta[punt].position;
    }

    //Amb aquest mètode restarem la vida de l'enemic i modificarem la barra de
vida
    public void RestarVida(float mal)
    {
        vida -= mal;
        barraVida.fillAmount = vida/auxVida;
    }
}
```

EscollirNivells.cs

```
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.UI;

public class EscollirNivell : MonoBehaviour {

    public string nNivell;//Número del nivell
    public Button[] nivells;//Array amb tots els nivells que té el joc

    private void Start()
    {
        int progres = PlayerPrefs.GetInt("progres", 1);//Això ens servirà per
saber per quin nivell anem. L'inicialitzem a 1 perquè serà el primer nivell que
jugarem.
        for (int i = 0; i < nivells.Length; i++)//En cas que no ens passem els
nivells, no podrem accedir a nivells superiors.
        {
            if (i + 1 > progres)
            {
            nivells[i].interactable = false;
```

```
            nivells[i].image.color = Color.gray;
            }
        }
    }
    //Cridem el nivell.
    public void Eleccio()
    {
        SceneManager.LoadScene(nNivell);
    }
}
```

Fi.cs

```
using UnityEngine;
using UnityEngine.SceneManagement;

public class Fi : MonoBehaviour {

    //Mètode per començar de nou el nivell
       public void TornarHi()
    {
        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);
    }

    //Mètode per tornar al menú principal
    public void Sortir()
    {
        SceneManager.LoadScene("MenuPrincipal");
    }
}
```

Instruccions.cs

```
using UnityEngine;
using UnityEngine.SceneManagement;

public class Instruccions : MonoBehaviour {

    private void Start()
    {
        Time.timeScale = 0;//Pararem el temps per a no poder apretar cap botó que
tenim radere del panell d'instruccions (era un bug que he arreglat així).
    }

    //Mètode per tornar al menú principal
    public void Sortir()
    {
        Time.timeScale = 0.6f;
        SceneManager.LoadScene("MenuPrincipal");
    }
}
```

Jugador.cs

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

public class Jugador : MonoBehaviour {
```

```
    public GameObject fi;//Per activar el fi del joc
    public GameObject victoria;//Per activar el fi del joc
    public Text textVides;//Text de les vides
    public Text textDiners;//Text dels diners
    public Text textPunts;//Text dels punts
    public Text textRonda;//Text de la ronda
    public int diners;//diners del jugador
    public string seguentNivell;//Text del següent nivell
    public int numSeguentNivell;//Número del següent nivell

    private int punts = 0;//Punts del joc
    private int vides = 5;//vides del jugador
    private int ronda = 0;//Ronda del nivellc

    private void Start()
    {
        Time.timeScale = 0.6f;
    }

    void Update () {

        if (vides <= 0)//Controlem que no ens hagin matat. Si ens maten aturarem
el temps i eliminarem tot de la pantalla per evitar bugs.
        {
            Time.timeScale = 0;
            EliminarObjectes();
            fi.SetActive(true);
            textPunts.text = punts.ToString();
        }
        textRonda.text = ronda.ToString();
        textVides.text = vides.ToString();
        textDiners.text = diners.ToString();
    }

    //Funció per retornar els diners
    public int QuantsDiners()
    {
        return diners;
    }

    //Mètode per sumar o restar les vides
    public void RestaVides(int vida)
    {
        vides -= vida;
    }

    //Mètode per sumar o restar els diners
    public void RestarSumarDiners(int monedes)
    {
        diners += monedes;
    }

    //Mètode per sumar els punts
    public void SumarPunts(int punt)
    {
        punts += punt;
    }

    //Mètode per sumar la ronda
    public void SumarRondes(int r)
    {
```

```
        GameObject.Find("TextRonda").GetComponent<Animation>().Play();
        StartCoroutine(Esperar(r));
    }

    //Aquesta funció és per esperar a canviar el número del nivell i que quadri
amb l'animació que hi ha feta.
    IEnumerator Esperar(int rd)
    {
        yield return new WaitForSeconds(0.5f);
        ronda = rd;
    }

    //Si acabem amb les oleades passem al següent nivell. En cas que la variable
sigui 0 vol dir que hem guanyat tots els nivells.
    public void Victoria()
    {
        Time.timeScale = 0f;
        EliminarObjectes();
        if (numSeguentNivell != 0){
            PlayerPrefs.SetInt("progres", numSeguentNivell);
            victoria.SetActive(true);
        }
        else{
            SceneManager.LoadScene("MenuFinal");
        }
    }

    // Mètode per eliminar els objectes de l'escenari creats
    private void EliminarObjectes()
    {
        GameObject[] destruir;

        destruir = GameObject.FindGameObjectsWithTag("Enemic");
        foreach (GameObject enemic in destruir)
        {
            Destroy(enemic);
        }

        destruir = GameObject.FindGameObjectsWithTag("Torre");
        foreach (GameObject torre in destruir)
        {
            Destroy(torre);
        }

        destruir = GameObject.FindGameObjectsWithTag("Sang");
        foreach (GameObject sang in destruir)
        {
            Destroy(sang);
        }
    }
}
```

Menu.cs

```
using UnityEngine;
using UnityEngine.SceneManagement;

public class Menu : MonoBehaviour
{

    public void Jugar()
    {
```

```
        SceneManager.LoadScene("MenuNivells");//Carreguem l'escena.
    }

    public void Sortir()
    {
        Application.Quit();//Tanquem l'aplicació
    }
}
```

ModificadorTemps.cs

```
using UnityEngine;

public class ModificadorTemps : MonoBehaviour {

    //Funció per a pausar el joc
       public void Pausa()
    {
        Time.timeScale = 0;
    }

    //Funció per a tornar al temps normal
    public void Continua()
    {
        Time.timeScale = 0.6f;
    }

    //Funció per anar més ràpid
    public void AnarRapid()
    {
        Time.timeScale = 1;
    }

}
```

Oleada.cs

```
using UnityEngine;

//Aquesta classe ens servirà per guardar un array dels enemics que apareixen en
cada oleada
[System.Serializable]
public class Oleada{

    public GameObject enemic;
    public int numeroEnemics;
    public float tempsEntreEnemics;
}
```

Projectil.cs

```
using UnityEngine;

public class Projectil : MonoBehaviour {

    private float moviment = 50f;//Rapidesa de moviment del projectil
    private GameObject objectiu = null;//Objectiu del projectil
    private float mal;//El mal que ha de fer vindrà passat des de la torre
```

```
    //Primer mirarem que tinguem un objectiu assignat i, a continuació, farem
moure el projectil fins a l'objectiu  fins que el toqui
    //per a cridar el mètode ObjectiuAconseguit().
    void Update () {

        if (objectiu != null)
        {
            Vector3 direccio = objectiu.transform.position - transform.position;
            transform.Translate(direccio.normalized * moviment * Time.deltaTime,
Space.World);

            if (Vector3.Distance(transform.position, objectiu.transform.position)
<= 1.3f)
            {
                ObjectiuAconseguit();
            }
        }
        else
        {
            Destroy(gameObject);
        }
    }

    //Passem el mal de la torre que crea el projectil
    public void SetMal(float malPassat)
    {
        mal = malPassat;
    }

    //Amb aquest mètode assignarem l'enemic més proper amb la seva posició com a
objectiu
    public void BuscarEnemic(GameObject enemic)
    {
        objectiu = enemic;
    }

    //Amb aquest mètode borrarem l'objecte i restarem la vida de l'enemic
        public void ObjectiuAconseguit()
    {
        objectiu.GetComponent<Enemics>().RestarVida(mal);
        Destroy(gameObject);
    }
}
```

RajolaTorre.cs

```
using UnityEngine;
using TMPro;

public class RajolaTorre : MonoBehaviour {

    public TextMeshPro textCostNormal;//text del que costa la torre normal
    public TextMeshPro textVendreNormal;//text del que es rep al vendre la torre
normal
    public TextMeshPro textCostNormalNv2;//text del que costa la torre normal
nivell 2
    public TextMeshPro textCostForta;//text del que costa la torre forta
    public TextMeshPro textVendreForta;//text del que es rep al vendre la torre
forta
    public TextMeshPro textCostFortaNv2;//text del que costa la torre forta
nivell 2
    public TextMeshPro textCostRapida;//text del que costa la torre ràpida
```

```
    public TextMeshPro textVendreRapida;//text del que es rep al vendre la torre
ràpida
    public TextMeshPro textCostRapidaNv2;//text del que costa la torre ràpida
nivell 2
    public TextMeshPro textVendreNv2;
    public GameObject menuConstruccio;//Menu per construir la torre
    public GameObject menuConstruccioNormal;//Menu per vendre i millorar les
torres normals
    public GameObject menuConstruccioForta;//Menu per vendre i millorar les
torres fortes
    public GameObject menuConstruccioRapida;//Menu per vendre i millorar les
torres ràpides
    public GameObject menuConstruccioAvansat;//Menu per vendre torres de nivell 2
    public GameObject torreNormal;//Objecte torre normal
    public GameObject torreNormalNv2;//Objecte torre normal
    public GameObject torreForta;//Objecte torre ràpida
    public GameObject torreFortaNv2;//Objecte torre ràpida
    public GameObject torreRapida;//Objecte torre ràpida
    public GameObject torreRapidaNv2;//Objecte torre ràpida
    //public GameObject jugador;//Objecte jugador
    GameObject[] rajoles;//array per a guardar totes les rajoles construïbles del
joc

    private int costNormal = 5;//El que costa la torre normal
    private int costNormalNv2 = 45;//El que costa la torre normal nivell 2
    private int costForta = 15;//El que costa la torre forta
    private int costFortaNv2 = 90;//El que costa la torre forta nivell 2
    private int costRapida = 20;//El que costa la torre ràpida
    private int costRapidaNv2 = 120;//El que costa la torre ràpida nivell 2
    private bool menuObert = false;//Boleà per saber si estem amb el menú de
construcció
    private GameObject obj;//Objecte utilitzat per a referenciar els objectes
torre
    private float tempsMenu = 4f;//Temps que el menú estarà obert abans no es
tanqui
    private GameObject menu;//Objecte menu
    private bool esNormal = false;//booleà per saber a l'hora de pujar de nivell
o vendre quina torre tenim
    private bool esForta = false;//booleà per saber a l'hora de pujar de nivell o
vendre quina torre tenim
    private bool esRapida = false;//booleà per saber a l'hora de pujar de nivell
o vendre quina torre tenim
    private bool esMillorada = false;//booleà per saber si la torre que hi ha
creada és millorada o no

    private void Start()
    {
        rajoles = GameObject.FindGameObjectsWithTag("Rajola");
        textCostNormal.text = costNormal.ToString();
        textCostForta.text = costForta.ToString();
        textCostRapida.text = costRapida.ToString();
        textVendreNormal.text = (costNormal / 2).ToString();
        textVendreForta.text = (costForta / 2).ToString();
        textVendreRapida.text = (costRapida / 2).ToString();
        textCostNormalNv2.text = costNormalNv2.ToString();
        textCostFortaNv2.text = costFortaNv2.ToString();
        textCostRapidaNv2.text = costRapidaNv2.ToString();
    }

    //Un cop apretem a una rajola d'herba, si hi ha un altra menú obert el
destruirà i s'obrirà el menú nou.
```

```
    //En cas que trobem, gràcies als booleans, que tenim una torre construïda,
obrirà un menú diferent
    //en funció de la torre que hi hagi.
    private void OnMouseDown()
    {
        if ((GameObject.Find("MenuConstruccio(Clone)") != null) ||
(GameObject.Find("MenuAmbTorreNormal(Clone)") != null) ||
(GameObject.Find("MenuAmbTorreForta(Clone)") != null) ||
(GameObject.Find("MenuAmbTorreRapida(Clone)") != null) ||
(GameObject.Find("MenuAmbTorreNv2(Clone)") != null))
        {
            Destroy(GameObject.Find("MenuConstruccio(Clone)"));
            for (int i = 0; i < rajoles.Length; i++)
            {
                rajoles[i].GetComponent<RajolaTorre>().Resetejar();
            }
        }
        if ((menuObert == false) && (obj == null))
        {
            Vector3 posicioMenu = transform.position;
            posicioMenu.z = -12;
            menu = (GameObject)Instantiate(menuConstruccio, posicioMenu,
transform.rotation);
            menuObert = true;
        }

        if ((menuObert == false) && (obj == true) && (esMillorada == false))
        {
            Vector3 posicioMenu = transform.position;
            posicioMenu.z = -12;
            if (esNormal == true)
            {
                menu = (GameObject)Instantiate(menuConstruccioNormal,
posicioMenu, transform.rotation);
            }
            else if (esForta)
            {
                menu = (GameObject)Instantiate(menuConstruccioForta, posicioMenu,
transform.rotation);
            }
            else if (esRapida)
            {
                menu = (GameObject)Instantiate(menuConstruccioRapida,
posicioMenu, transform.rotation);
            }
            menuObert = true;
        }

        if ((menuObert == false) && (obj == true) && (esMillorada == true))
        {
            if (esNormal)
            {
                textVendreNv2.text = (costNormalNv2 / 2).ToString();
            }else if (esForta)
            {
                textVendreNv2.text = (costFortaNv2 / 2).ToString();
            }else if (esRapida)
            {
                textVendreNv2.text = (costRapidaNv2 / 2).ToString();
            }
            Vector3 posicioMenu = transform.position;
            posicioMenu.z = -12;
```

```
            menu = (GameObject)Instantiate(menuConstruccioAvansat, posicioMenu,
transform.rotation);
            menuObert = true;
        }
    }

    //Per a crear les torres utilitzarem les tecles 1, 2 o 3 del teclat. En cas
d'apretar-ne d'altres o no fer res el menú de construcció es quedarà en standby i
es tancarà
    //automàticament. En cada if es mira si es tenen prous diners i, en cas que
sí, es crea la torre, es resten els diners i activem el boleà que hi ha per
aquella torre.
    private void Update()
    {
        if (menuObert == true && (obj == null))
        {
            if (tempsMenu <= 0f)
            {
                Resetejar();
            }
            else
            {
                if (Input.GetKey("1"))//torre normal
                {
                    if
(GameObject.Find("CodiBase").GetComponent<Jugador>().QuantsDiners() >=
costNormal)
                    {

GameObject.Find("CodiBase").GetComponent<Jugador>().RestarSumarDiners(-
costNormal);
                        Vector3 posicioTorre = transform.position;
                        posicioTorre.z = 13;
                        Resetejar();
                        obj = (GameObject)Instantiate(torreNormal, posicioTorre,
transform.rotation);
                        obj.transform.Rotate(-55, -22, 15);
                        esNormal = true;
                        return;
                    }
                }
                if ((Input.GetKey("2") &&
GameObject.Find("CodiBase").GetComponent<Jugador>().QuantsDiners() >= costForta))
//torre forta
                {

GameObject.Find("CodiBase").GetComponent<Jugador>().RestarSumarDiners(-
costForta);
                    Vector3 posicioTorre = transform.position;
                    posicioTorre.z = 13;
                    Resetejar();
                    obj = (GameObject)Instantiate(torreForta, posicioTorre,
transform.rotation);
                    obj.transform.Rotate(-55, -22, 15);
                    esForta = true;
                    return;
                }
                if ((Input.GetKey("3") &&
GameObject.Find("CodiBase").GetComponent<Jugador>().QuantsDiners() >=
costRapida)) //torre ràpida
                {
```

```
GameObject.Find("CodiBase").GetComponent<Jugador>().RestarSumarDiners(-
costRapida);
                    Vector3 posicioTorre = transform.position;
                    posicioTorre.z = 13;
                    Resetejar();
                    obj = (GameObject)Instantiate(torreRapida, posicioTorre,
transform.rotation);
                    obj.transform.Rotate(-55, -22, 15);
                    esRapida = true;
                    return;
                }
            }
            tempsMenu -= Time.deltaTime; //Resta un cada segon
        }

        //En cas que tinguem una torre construïda (on sabrem quina gràcies al
boleà) tenim la opció de vendre a meitat de preu la torre o bé
        //millorar-la si es tenen prous diners
        if (menuObert == true && (obj == true) && (esMillorada == false))
        {
            if(tempsMenu <= 0f)
            {
                Resetejar();
            }

            if (Input.GetKey("1"))
            {
                Resetejar();
                if (esNormal == true)
                {

GameObject.Find("CodiBase").GetComponent<Jugador>().RestarSumarDiners(costNormal
/ 2);
                    esNormal = false;
                }
                else if (esForta == true)
                {

GameObject.Find("CodiBase").GetComponent<Jugador>().RestarSumarDiners(costForta /
2);
                    esForta = false;
                }
                else if (esRapida == true)
                {

GameObject.Find("CodiBase").GetComponent<Jugador>().RestarSumarDiners(costRapida
/ 2);
                    esRapida = false;
                }
                Destroy(obj);

            }
            if (Input.GetKey("2"))
            {
                if ((esNormal == true) &&
(GameObject.Find("CodiBase").GetComponent<Jugador>().QuantsDiners() >=
costNormalNv2))
                {
                    Destroy(obj);
```

```
GameObject.Find("CodiBase").GetComponent<Jugador>().RestarSumarDiners(-
costNormalNv2);
                    Vector3 posicioTorre = transform.position;
                    posicioTorre.z = 13;
                    Resetejar();
                    obj = (GameObject)Instantiate(torreNormalNv2, posicioTorre,
transform.rotation);
                    obj.transform.Rotate(-55, -22, 15);
                    esMillorada = true;
                    return;
                }
                else if ((esForta == true) &&
(GameObject.Find("CodiBase").GetComponent<Jugador>().QuantsDiners() >=
costFortaNv2))
                {
                    Destroy(obj);

GameObject.Find("CodiBase").GetComponent<Jugador>().RestarSumarDiners(-
costFortaNv2);
                    Vector3 posicioTorre = transform.position;
                    posicioTorre.z = 13;
                    Resetejar();
                    obj = (GameObject)Instantiate(torreFortaNv2, posicioTorre,
transform.rotation);
                    obj.transform.Rotate(-55, -22, 15);
                    esMillorada = true;
                    return;
                }
                else if ((esRapida == true) &&
(GameObject.Find("CodiBase").GetComponent<Jugador>().QuantsDiners() >=
costRapidaNv2))
                {
                    Destroy(obj);

GameObject.Find("CodiBase").GetComponent<Jugador>().RestarSumarDiners(-
costRapidaNv2);
                    Vector3 posicioTorre = transform.position;
                    posicioTorre.z = 13;
                    Resetejar();
                    obj = (GameObject)Instantiate(torreRapidaNv2, posicioTorre,
transform.rotation);
                    obj.transform.Rotate(-141, -362, 349);
                    esMillorada = true;
                    return;
                }
            }
            tempsMenu -= Time.deltaTime; //Resta un cada segon
        }
        //En cas que tinguem la torre millorada, mirem de quin tipus és per
retornar els diners que toquen.
        if (menuObert == true && (obj == true) && (esMillorada == true))
        {
            if (tempsMenu <= 0f)
            {
                Resetejar();
            }
            if (Input.GetKey("1"))
            {
                Resetejar();
                if (esNormal == true)
                {
```

```
GameObject.Find("CodiBase").GetComponent<Jugador>().RestarSumarDiners(costNormalN
v2 / 2);
                    esNormal = false;
                    esMillorada = false;
                }
                else if (esForta == true)
                {

GameObject.Find("CodiBase").GetComponent<Jugador>().RestarSumarDiners(costFortaNv
2 / 2);
                    esForta = false;
                    esMillorada = false;
                }
                else if (esRapida == true)
                {

GameObject.Find("CodiBase").GetComponent<Jugador>().RestarSumarDiners(costRapidaN
v2 / 2);
                    esRapida = false;
                    esMillorada = false;
                }
                Destroy(obj);
            }
            tempsMenu -= Time.deltaTime; //Resta un cada segon
        }
    }

    //Inicialitzem de nou les variables que controlen i destruïm el menú
    public void Resetejar()
    {
        Destroy(menu);
        menu = null;
        menuObert = false;
        tempsMenu = 4f;
    }
}
```

Ruta.cs

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

public class Ruta : MonoBehaviour {

    public static Transform[] ruta;//Array amb les posicions dels punts de ruta

       void Awake () {

        //Obtenim tots els elements de Ruta, amb les coordenades de cadascun
        ruta = new Transform[transform.childCount];
        for (int i = 0; i < ruta.Length; i++)
        {
            ruta[i] = transform.GetChild(i);
        }
       }
}
```

Sang.cs

```
using UnityEngine;
```

```
public class Sang : MonoBehaviour
{

    public float temps = 15f;

    void Update()
    {
        Destroy(gameObject, 15f);//Tot i que en l'animació de sang s'elimina
visualment, destruïm l'objecte per no acumular-los i poder arribar a saturar els
recursos de la màquina.
    }
}
```

Torre.cs

```
using UnityEngine;

public class Torre : MonoBehaviour {

    public float rang = 15f;//rang que té la torre
    public float velocitat = 0.75f;//velocitat d'atac de la torre
    public float mal = 10f;//mal que fa la torre
    public Transform objectiu;//Objectiu de la torre
    public GameObject projectil;//Per a crear les fletxes
    GameObject enemicObjectiu = null; // Enemic al que atacarem

    private bool atac = false;//booleà per saber si la torre ha atacat o no
    private float releaseAtac = 0f;//Temps fins el següent atac
       void Update () {

        //Primer mirarem quants enemics hi ha dins del terreny de joc per a saber
a qui atacar i, a continuació, calcularem la distància amb el que estigui més a
prop de la torre.
        //Després actualitzarem l'enemic objectiu en funció que en tinguem algún
encara més proper. En cas de tenir un objectiu i que aquest estigui dins del
rang, cridarem al
        //mètode Dispar() i restarem en segons amb la variable de velocitat per a
saber quan podem tornar a disparar.
        float posicioEnemic = 100f;//Inicialitzem la variable a un valor enorme
        GameObject[] enemics =
GameObject.FindGameObjectsWithTag("Enemic");//Creem un array amb tots els enemics
que hi ha sobre el terreny de joc
        enemicObjectiu = null; // Enemic al que atacarem

        for (int i = 0; i < enemics.Length; i++)
        {
            float distancia = Vector3.Distance(transform.position,
enemics[i].transform.position);

            if (distancia < posicioEnemic)
            {
                posicioEnemic = distancia;
                enemicObjectiu = enemics[i];
            }
        }

        if (posicioEnemic <= rang)
        {
            objectiu = enemicObjectiu.transform;
```

```
        }
        else
        {
            objectiu = null;
        }

        if ((objectiu != null) && (atac == false))
        {
            Dispar();
            atac = true;
            releaseAtac = velocitat;
        }

        if ((atac == true) && (releaseAtac <= 0f))
        {
            atac = false;
        }

        releaseAtac -= Time.deltaTime;
    }

    //El mètode Dispar() simplement crea un projectil que, a més, associarà a la
classe Projectil i així cridar el mètode SetMal passant-li el mal i BuscarEnemic
per a passar-li l'objectiu.
    //També executa el soroll del projectil,
    void Dispar()
    {
        Projectil fl = Instantiate(projectil, transform.position,
transform.rotation).GetComponent<Projectil>();
        GetComponent<AudioSource>().Play(0);
        fl.SetMal(mal);
        fl.BuscarEnemic(enemicObjectiu);
    }
}
```

Victoria.cs

```
using UnityEngine;
using UnityEngine.SceneManagement;

public class Victora : MonoBehaviour {

    public string seguentNivell;//Variable per determinar quin és el següent
nivell.

    public void Start()
    {
        transform.SetAsLastSibling();
    }

    //Mètode per començar amb el següent nivell
    public void Seguent()
    {
        SceneManager.LoadScene(seguentNivell);
    }

    //Mètode per tornar al menú principal
    public void Sortir()
    {
        SceneManager.LoadScene("MenuPrincipal");
    }
}
```