KERMOVITE DEFENDERS

Guillermo Bascuñana Soler
Engineyeria Informàtica
Videojocs

Joel Servitja Feu
Javier Luis Cánovas Izquierdo

05 de Gener de 2020

FITXA DEL TREBALL FINAL

Títol del treball:	*Kermovite Defenders*
Nom de l'autor:	*Guillermo Bascuñana Soler*
Nom del consultor/a:	*Joel Servitja Feu*
Nom del PRA:	*Javier Luis Cánovas Izquierdo*
Data de lliurament (mm/aaaa):	*05/2020*
Titulació o programa:	*Grau d'Enginyeria Informàtica*
Àrea del Treball Final:	*Videojocs*
Idioma del treball:	*Català*
Paraules clau	*Videojocs, Enginyeria, Informàtica*

Resum del Treball (màxim 250 paraules): *Amb la finalitat, context d'aplicació, metodologia, resultats i conclusions del treball*

Kermovite Defenders es un videojoc tower defense futurista d'estètica retro per PC i macOS. Aquest projecte s'ha realitzat amb l'engine Unity. El resultat ha sigut un producte mínim d'un videojoc complet, amb menú, selecció de tres nivells i partides totalment jugables.

Aquest joc tower defense s'intenta diferenciar ja que no existeix un total de vides que es perden al defensar un punt. Els enemics entren a robar 5 objectes que están al final d'un camí. Han de sortir amb ells i si son eliminats durant el camí deixen caure l'objecte, que pot ser agafat per un altre enemic

L'estratègia d'aquest joc canvia lleugerament d'altres tower defense degut a aquesta mecànica junt amb els interessos que s'obtenen sobre els diners que no es gasten. El joc comença fàcil per fer que el jugador aprengui les mecàniques, però la dificultat augmenta.

Al final del desenvolupament he aconseguit unes bases i coneixements de com fer un videojoc des de el plantejament de l'idea fins al producte final.

Abstract (in English, 250 words or less):

Kermovite Defenders is a futuristic tower defense videogame with retro aesthetics for PC and macOS. This project hs been developed in Unity engine.

The end result is a minimum viable product of a complete game, with a menu, level selection screen with three levels and a playable gameplay.

This tower defense game sets aside from it's other tower defense counterparts due to the fact that there is no specific point where, if enemies reach it, you loose life. The enemies will try to steal an object and leave the map with it. If the enemy reaches the object and is destroyed on it's way out, the object drops on the map floor, making it available for other enemies.

This slightly changes the strategy, and can be combined with the fact that you get an interest rate from money not spent. The game starts easy so the player can get used to all of it's mechanics, but it slightly gets more difficult.

At the end of the development I have achieved a solid foundation and knowledge on how to create a videogame project, from the idea to the final product.

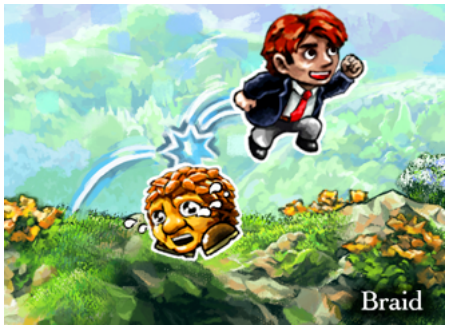
Índex

Llista de figures

1. Introducció

1.1 Context i justificació del Treball

Des de que va aparèixer el videojoc PONG a les maquines recreatives de molts llocs públics l'industria dels videojocs no ha deixat de créixer fins a arribar a un punt on ja és més gran que les industries del cinema i la música juntes [1]. Això ho converteix en un sector del desenvolupament de software molt atractiu. Sobretot des de l'aparició de les descarregues digitals en comptes del format físic com a mètode de venda. Això ha facilitat a persones independents o organitzacions petites la incorporació a un mercat on abans hi havia bastantes barreres d'entrada.



Il·lustració 1 - Braid, un dels primers jocs indie d'èxit desenvolupat per una sola persona (Jonathan Blow, 2008)

Des de bastant petit m'havia sentit molt atret per els videojocs, sobretot durant l'era dels 90 amb la **Super Nintendo, GameBoy i la Nintendo 64**. Amb els anys vaig anant perdent l'interès en aquest mon, en part degut als tipus de jocs cap a on tendia el mercat, cada vegada més intent de fer-los realistes i jugabilitats menys imaginatives, de fet em semblaven tots iguals. Això va començar a canviar aquesta última dècada amb el boom de desenvolupadors independents[2]. Sense una gran empresa darrera i sense arriscar tants recursos, els desenvolupadors **indie** podien experimentar més, crear idees originals, tornar a formats antics o trencar tots els esquemes amb propostes que mai arribarien a producció en una empresa com Electronic Arts. Molts d'aquests videojocs no tenen res a envejar de produccions **AAA**, alguns d'ells han superat a aquests en valoracions i vendes, fins i tot jocs desenvolupats integrament per una sola persona. Fins hi tot crear un videojoc indie que simula la creació de videojocs indie, com es el cas de Game Dev Tycoon. Un exemple molt clar seria el increïblement popular minecraft, un joc desenvolupat per una sola persona que s'ha convertit en el joc més venut de la història, per sobre de productes com Tetris o GTA V[3]. Això va tornar a despertar en mi l'interès pels videojocs, però lo que realment em va cridar l'atenció era el seu desenvolupament.

Il·lustració 2 - Hollow Knight (Team Cherry, 2017)

Fa 3 anys que vaig començar a dedicar a la programació, concretament al **desenvolupament web full-stack**, però sempre m'ha agradat la idea de traslladar aquests coneixements per començar a crear videojocs indie. L'idea de crear alguna cosa com Hollow Knight o Braid i llançar-la al públic em faria sentir més realitzat que una vida de dedicació al meu camp actual. Durant aquests anys també es una cosa que mai he començat, poder he llegit alguna cosa i he tingut Unity

instal·lat des de fa mitja dècada, però sense arrencar pràcticament mai. Es per tant que he decidit escollir aquest tema, ja que ho trobo l'excusa perfecte per aprendre les bases del desenvolupament de videojocs.

En quant a l'estètica i temàtica del joc ha sigut un equilibri entre gust personal i senzillesa, ja que es el meu primer intent en aquest mon. Els jocs d'aquest gènere son petits i senzills, normalment creats per **jocs flash** que funcionen al navegador web. L'estètica, un estil retro, també es una forma senzilla de dibuixar i apte per gent com jo, sense gaires coneixements artístics.

Il·lustració 3 - Game Dev Tycoon (Greenheart Games. 2012), un joc que simula la creació d'una empresa de videojocs indie

1.2 Objectius del Treball

Els objectius del treball es poden separar en tres tipus diferents, d'aprenentatge, d'aplicació de coneixements i de resultats:

Objectius d'aprenentatge

- Aprendre el funcionament del motor Unity 2D
- Aprendre els fonaments del desenvolupament de videojocs
- Aprendre tècniques bàsiques de píxel art

Objectius d'aplicació de coneixements

- Aplicar els coneixements de programació orientada a objectes obtinguts durant el grau
- Aplicar els coneixements de eficiència algorísmica obtinguts durant el grau
- Aplicar els coneixements d'estructures de dades i planificació obtinguts durant el grau

Objectius de resultats

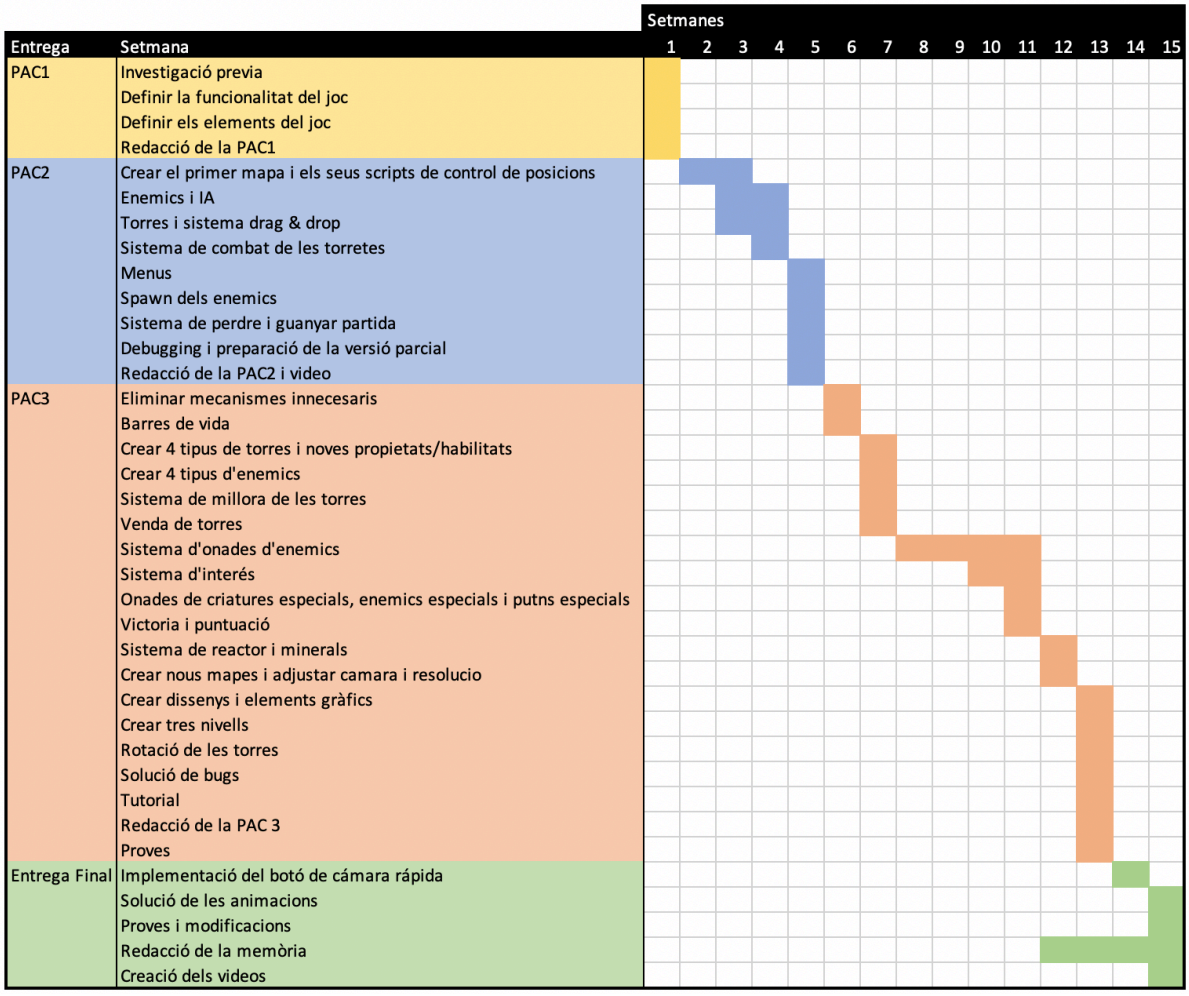
- Crear una versió MVP d'un joc tower defense jugable
- Crear 3 nivells per el joc amb dificultat variable
- Crear dissenys de píxel art propis per tot el joc
- Manipular els valors fins a arribar a un equilibri que faci el joc divertit

1.3 Enfocament i mètode seguit

El mètode seguit ha sigut en tres parts. En primer lloc m'he dedicat a documentar-me i fer un tutorial sobre l'eina Unity i C#. Una vegada aprés com funciona d'una manera més general he construït les bases del joc amb dos tutorials del gènere concret, ja que com és un tema ampli necessitava veure per on començar. A partir d'aquí he anat desenvolupant totalment pel meu compte, anant investigant cada funcionalitat nova si calia. La tècnica per portar el control de la funcionalitat que anava desenvolupant i les tasques que em quedaven ha sigut fer servir un panell de SCRUM a de Trello i fer servir diferents branques de Git etiquetades amb cada tasca. Parlaré d'això amb més profunditat a l'apartat 4.4.

1.4 Planificació del Treball

La planificació del treball ha variat una mica de la establerta al principi (PAC1), però no s'ha allunyat gaire. Per portar-ho a terme he fet servir un panell de **SCRUM** que proporciona l'eina Trello (www.trello.com) i Github (www.github.com). Entraré més en detall d'aquest sistema des d'un punt de vista més tècnic a l'apartat 4.3. Amb això els resultats han sigut els següents:



1.5 Breu sumari de productes obtinguts

El resultat ha estat un joc funcional per **OSX** i Windows de tower defense amb tres nivells. El joc ha sigut polit per evitar errors tant de programació com de disseny d'experiència d'usuari. Per una altre banda això conformaria un MVP i no un producte que podria ser comercial, ja que la quantitat de contingut encara es baixa.

1.6 Breu descripció dels altres capítols de la memòria

En els següents capítols s'analitzarà amb detall els diferents elements del producte obtingut, les seves característiques tècniques i el seu desenvolupament.

Després hi ha un manual d'instruccions del joc, creat com els manuals d'instruccions en el mateix estil dels manuals que incloïen els videojocs a la época de la **NES** i **SNES**[3]

Al final hi hauran unes conclusions finals on s'analitzen els resultats, el procés i es fa una crítica d'aquests.

Com a últims elements hi hauran totes les referències bibliogràfiques i un glossari de termes per la millora de la comprensió del text.

2. Estat de l'Art

2.1 Gènere del joc: Tower Defense amb estil retro

El gènere del tower defense pertany al grup dels jocs d'estratègia. Comunment, es basa en defensar un punt concret d'enemics que van apareixent contínuament. Aquesta defensa s'aconsegueix construint torres sobre un terreny perquè automàticament vagin disparant als enemics. El ***game loop*** es la planificació i l'avaluació continua dels resultats de col·locar les torres.

Il·lustració 4 - Line Tower, Warcraft III (Blizzard Entretainment, 2002)

Aquest genere te semblances amb altres jocs d'estratègia com els jocs d'estrategia en temps real (*Age of Empires, Starcraft, Warcraft*). De fet jocs com Warcraft 3 han popularitzat **mods** del genere tower defense, com pere exemple *line tower*. Peró els precursors d'aquest genere son molt anteriors a les series esmentades. Els elements d'aquest genere es poden trobar a jocs tant antics com *Gandalf the Sorcerer* (1984, Commodore64, Game Gems) o *Greenhouse* (1982, Game & Watch, Nintendo), peró el joc amb les bases de lo que es ara el gènere seria *Rampart* (1990, PC, Atari). En aquest joc el jugador construeix torres per defensar un castell d'altres jugadors[4].

Il·lustració 5 - Rampart (Atari, 1990)

Des de que aquestes bases es van establir el gènere ha passat per moltes fases. Fins al voltant del 2009, els tower defense eren comunament jocs de flash o mods d'altres jocs d'estratègia com l'esmentat anteriorment. Va ser després, amb la popularització dels smartphones i les botigues digitals que es van començar a fer molts jocs del gènere, molts d'ells adaptacions dels jocs de flash que portaven una dècada existint. El més popular d'aquests nous jocs va ser *Plants vs Zombies* (2009, multiplataforma, Electronic Arts), que va ser descarregat més de 300.000 vegades en els primers 9 dies després del seu llançament[5] i va guanyar diversos premis de la indústria. Avui dia es una de les franquícies de videojocs amb més èxit.

En quant a la part visual i l'identitat del joc, aquest pertany al gènere de ciència ficció amb una estètica retro construïda amb píxel art. Això es una pràctica molt comú en el mont dels desenvolupadors indies i altres aficionats als videojocs. Aquest estil era així degut a les limitacions de hardware del maquinari antic. Les imatges eren reduïdes a un número petit de píxels i s'havia de treballar amb això. Avui dia, encara que no hi hagin tantes limitacions de hardware, si que n'hi han de capacitat de

desenvolupament, sobretot amb els perfils esmentats. Crear jocs amb aquest estil permet crear videojocs amb menys esforços i sense necessitat d'un gran equip multidisciplinari, un sol desenvolupador pot fer totes les parts del joc, des de l'art fins a la programació. Avui dia aquest estil es molt present en jocs indie desenvolupats per una sola persona o grups petits.

Il·lustració 6 - Stardew Valley (Eric Barone/Chucklefish, 2016), joc indie amb un estil Pixel art popular, desenvolupat per una sola persona

2.2 Tecnologies

Antigament, per desenvolupar videojocs, calia fer-los en **ensamblador** i tenint en compte l'arquitectura de la màquina per la qual es desenvolupava. Això era un problema ja que era un procés molt més lent i costós i requeria personal especialitzat. Avui dia es fan servir dos camins, el desenvolupament d'un **engine** amb llenguatges propers a la màquina a desenvolupar i crear un joc amb ell o fer servir els engines disponibles del mercat. La primera opció requereix molt més temps i més personal especialitzat, es un mètode comú en grans organitzacions. El segon mètode permet a desenvolupadors independents crear jocs sense aplicar coneixements de baix nivell.

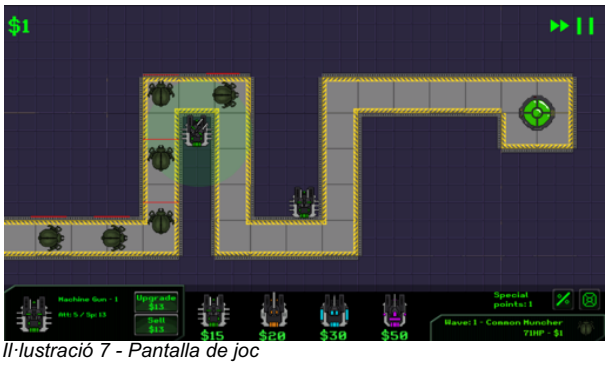
N'hi han bastants engines disponibles, com poden ser Unity, Unreal Engine, Game Maker Studio, Godot o RPG Maker. Per aquest projecte s'ha escollit Unity, s'entrarà en més detalls sobre aquesta decisió a l'apartat 4.1

3. Definició del Joc

3.1 Visió general

El videojoc consta d'un escenari amb una càmera fixa. Aquests escenaris poden variar depenent de la dificultat escollida, n'hi ha tres, fàcil, mitjana i difícil. Aquests escenaris tenen un recorregut que comença en una obertura i termina en un reactor.

El jugador te la opció d'enviar una onada d'enemics que aniran fent el recorregut fins arribar al reactor. Quan arribin al reactor agafaran un dels 5 minerals que aquest conté. En aquest moment els enemics començaran a tornar cap al principi del recorregut. Si els enemics amb els minerals aconsegueixen tornar amb els 5 minerals el jugador perd la partida.



Il·lustració 7 - Pantalla de joc

Per evitar això el jugador ha de construir torres de defensa al voltant del recorregut. Aquestes torres apareixen a la part interior de la interfície i tenen un cost econòmic. Les torres tenen diferents comportaments i atacs. Aquests es poden millorar gastant diners. Les torres tenen un determinat rang i disparen automàticament als enemics que es trobin a dins d'aquest. Quan un enemic mor el jugador obté diners. Si l'enemic portava un mineral aquest cau i es queda on s'ha matat a l'enemic. El proper enemic que passi per on està el mineral l'agafarà i tornarà a l'inici sense haver de fer tot el recorregut, això fa que sigui més perillós. Al final de cada onada el jugador obté un percentatge d'interès sobre els diners que no hagi gastat, per tant ha d'aprofitar això calculant quin es el mínim de torres que necessitarà per la següent onada tenint en compte els enemics que venen i la posició i quantitat dels minerals.

Apart de tot això, cada 5 onades apareixerà un enemic molt més fort i difícil de matar, si el jugador sobreviu a aquest se li donaran uns punts especials, aquests es poden gastar en augmentar un 1 per cent l'interès que es rep entre torns o retornar tots els minerals que estiguin fora del reactor al reactor. Els minerals que hagin sortit del mapa no es poden recuperar. La partida termina quan el jugador perd tots els minerals o sobreviu a totes les onades.

3.2 Elements del joc

Enemics

Els enemics venen en onades d'entre 30 i 50 unitats, tenen una determinada vida i velocitat. Alguns d'ells tenen propietats especials com armadura o velocitat augmentada. L'objectiu del jugador es destruir-los abans de que robin els minerals. N'hi ha de 5 tipus:

- Normal (*Common Muncher*): Aquests enemics tenen un comportament estàndard i venen en onades de 30. Son els més fàcils d'eliminar ja que no tenen cap habilitat especial que suposi un repte.

Il·lustració 8 - Sprite Common Muncher

- Grupal (*Groupal Critter*): Aquests tenen un comportament igual que els anteriors amb la diferència de que venen en onades de 50 però molt més junts, es a dir son més fàcils d'eliminar si es danyen a tots a la vegada però molt més difícils d'un en un.

Il·lustració 9 - Sprite Groupal Critter

- Ràpid: (*Fast Crawler*): Venen en onades de 30 i son molt més ràpids que la resta. El repte està en que fan el recorregut molt més ràpid i això suposa que son molt més difícils de matar.

Il·lustració 10 - Sprite Fast Crawler

- Armadura: (*Armored Eater*): Aquest també venen en onades de 30 i tenen l'atribut d'armadura. Això fa que els torres més bàsiques tinguin un infligeixin un dany reduït sobre ells. Es necessiten torres avançades per destruir-los efectivament. També tenen un increment de salut respecte als altres.

Il·lustració 11 - Sprite Armored Eater

- Lord: Versions especials de la resta, venen en onades de un i tenen un increment molt alt de salut. Matar-los dona al jugador punts especials.

Torres

Les torres son l'element principal que fa servir el jugador per jugar la partida, aquestes es construiran a les zones habilitades i dispararan automàticament als enemics. Les torres es poden millorar per incrementar els seus atributs o vendre per un 80% del total invertit en elles (compra mes millores). N'hi ha 4 tipus i cada una té unes propietats diferents:

- Torre metralladora: La torre més bàsica i barata. Dispara a un enemic a la vegada amb una velocitat ràpida i infligeix un dany lleu. És efectiva amb els enemics normals i ràpids, no ho és amb els que tinguin armadura o amb onades grans.

Il·lustració 12 - Sprite Torre Metralladora

- Torre de canó: Aquesta torre danya a molts enemics a la vegada amb un nivell d'atac similar al de la torre metralladora. És una torre més lenta i si es millora incrementa el rang amb el que es danya als enemics del voltant. Es especialment efectiva amb enemics que s'agrupen com el *Groupal Critter* o enemics s'hagin alentit prèviament.

Il·lustració 13 - Sprite Torre Canó

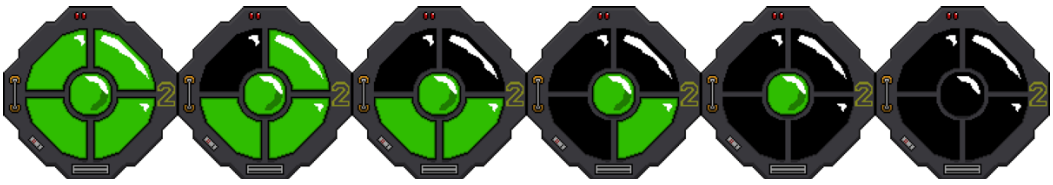
- Torre congeladora: Aquesta torre no causa dany però alenteix als enemics durant un cert període de temps. Millorar-la augmenta aquest temps i la seva velocitat. Es recomanable per enemics ràpids i combina molt be amb la torre de canó.

Il·lustració 14 - Sprite Torre Congeladora

- Torre làser: Aquesta es la torre amb l'atac més alt de totes i la més cara. Per una altre banda també es la més lenta de totes. El seu atac pot destruir enemics d'un sol cop però ho farà molt a poc a poc. Es molt inefectiva amb enemics ràpids o onades molt grans. Apart d'això perfora l'armadura dels enemics i a aquests els hi genera un dany extra. Es especialment efectiva amb les criatures especials i les criatures del tipus *Armored Eater*.

Il·lustració 15 - Sprite Torre Làser

Minerals (*Kermovite*)



Il·lustració 16 - Sprite del reactor amb tots els seus possibles estats

Aquests representen la vida del jugador. N'hi ha 5 i en primer lloc estan a dins del reactor, representats per les 5 seccions de color verd. Quan un enemic arriba al reactor el treurà, si l'enemic es eliminat el deixarà caure i llavors figurarà com a la il·lustració 17 fins que retorni al reactor. Quan es treu un mineral

Il·lustració 17 - Sprite del mineral

del reactor el segment es torna negre representant el buit. El jugador perdrà si els enemics s'emporten els 5 minerals.

Diners

Els diners dels que disposa el jugador, aquests aniran augmentant matant enemics o amb els interesos entre onades, també disminuiran quan es comprin torres o es millorin.

Terreny

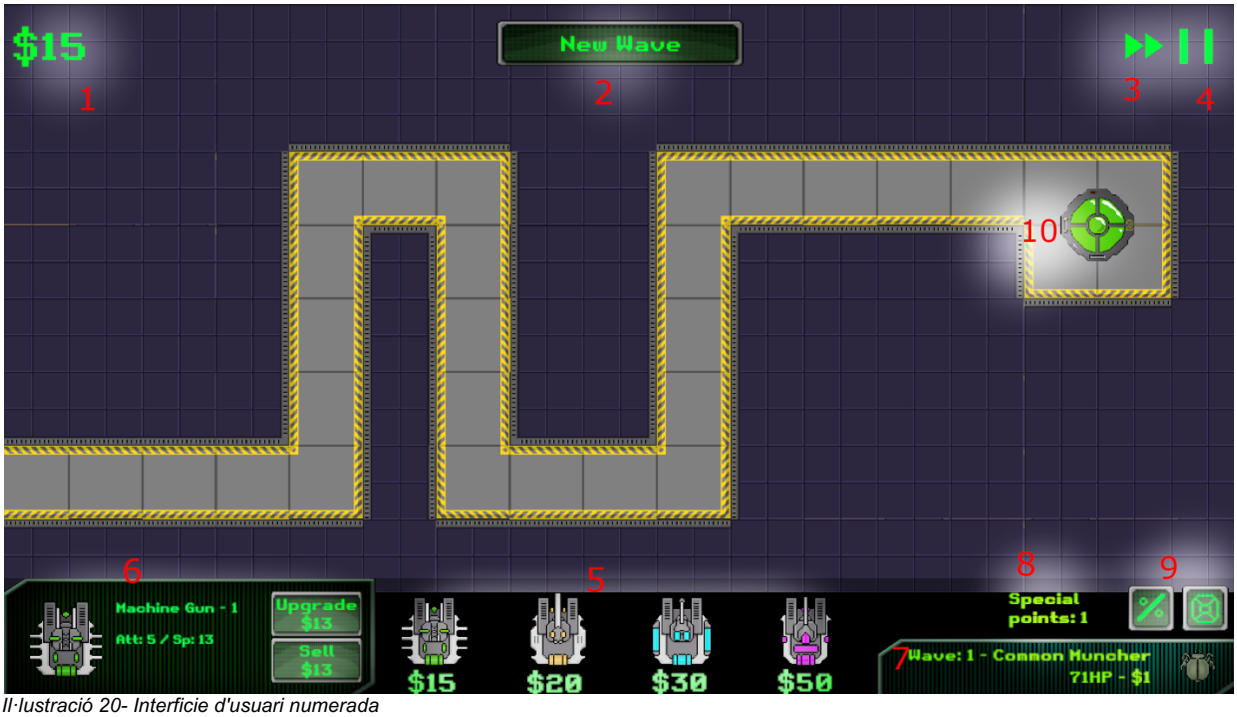
El terreny del joc es on es desenvolupa la partida, n'hi ha de dos tipus, el terreny per on passen els enemics, on no es pot construir, i el terreny construïble, on es poden construir les torres. S'han afegit assets amb diferents detalls per a cada tipus però es diferencien l'un de l'altre per el color, sent blau fosc el construïble i gris per on passen els enemics.

Il·lustració 19 - Sprite de terreny construible

Il·lustració 18 - Sprite de terreny caminable

3.3 Interacció de l'usuari amb els elements del joc

El jugador interactua amb el joc amb l'interficie, aquesta es un menu amb diverses opcions i informació:



Il·lustració 20- Interficie d'usuari numerada

1. Diners restants del jugador
2. Botó per enviar la següent onada
3. Botó per accelerar la velocitat, canvia a tornar a velocitat normal quan estigui actiu
4. Botó de pausa, també obre el menú de pausa on es pot sortir de la partida
5. Torres per construir, el jugador les selecciona clicant i després clica a la casella en la qual la vol construir
6. Informació de la torre seleccionada, mostra els atributs de la torre i dona dos opcions, millorar-la o vendre-la
7. Informació dels enemics de la següent onada
8. Punts especials disponibles
9. Opcions especials on el jugador gasta els punts especials, n'hi ha dos opcions, incrementar l'interés entre torns o retornar els minerals al reactor
10. Reactor, aquest sprite canvia segons els minerals que contingui, sent cada part verda un contenidor de mineral (o negre si esta buit)

3.4 Objectius, beneficis i puntuació

L'objectiu del joc es protegir els minerals i sobreviure a totes les onades, però com a objectiu més petits te el de planificar be les onades trobant un equilibri entre construcció de torres i estalvi. El jugador rebrà un interès entre torns, i si es va acumulant aquest comportarà un interès major més endavant. A mesura que la partida avança els enemics es fan més forts així que tenir més diners en el futur per construir o millorar torres serà sempre millor que gastar-los abans. El jugador ha de trobar el punt mínim de despeses per fer front a la següent onada estalviant i sense perdre minerals, que son prioritaris. Per motivar aquest comportament, al final de cada partida el jugador rebrà punts segons aquests factors seguint la següent fórmula:

$$(int)\left(\frac{dificultatOnada \cdot nEnemicsEliminats \cdot nMineralsEnPartida + diners}{10}\right)$$

Si es superen totes les onades es mostrarà la puntuació final. S'ha de tenir en compte que sobreviure es l'objectiu principal, si no es sobreviu la puntuació total no es mostra.

3.5 Fi de la partida

La partida finalitza de dues maneres, quan el jugador perd tots els minerals i, per tant, la partida, o quan el jugador supera totes les onades. En les dues situacions pot tornar al menú principal o tornar a jugar el mateix nivell.

4. Disseny Tècnic

4.1 Entorn de desenvolupament

Il·lustració 21 - Logo de Unity

L'engine de desenvolupament escollit ha sigut Unity, concretament la versió 2018.3.7f1. He pres aquesta decisió basant-me en els següents aspectes:

- Gratuït: Unity es un engine gratuït sempre que la persona o l'empresa que el faci servir ingressi menys de 100.000 dolars a l'any.

- Gran comunitat: Aquest engine es molt popular, sobretot en la comunitat de desenvolupadors indie. Es per això que existeixen milers de recursos o problemes resolts a la xarxa. Això facilita molt més el desenvolupament ja que si trobo qualsevol problema es molt probable que pugui trobar la solució.

- Tutorials: Per arrencar el projecte necessitava començar amb tutorials sobre l'eina, i per lo que he vist a primera vista la gran majoria eren de Unity.

Per la creació de *sprites* he fet servir l'eina Aseprite, un software de disseny per píxel art. A diferència d'altres eines com Photoshop, Aseprite es molt senzill i la seva elecció a sigut per recomanació. El software no es gratuït però te un preu molt baix.

Il·lustració 22 - Logo de Aseprite

Per portar el control de les diferentes tasques i les versions del desenvolupament e fet servir Git. La metodologia l'explicaré més a detall a l'apartat 4.4.

En quant al so de joc he agafat diferents sons del portal freesound.org, on n'hi han milers de sons amb llicencia creative commons. Els sons els he editat amb el programa ableton live. No es el mes optim per aquesta tasca pero tinc molta experiencia amb aquest programa i vaig molt mes rapid.

Com a últim punt, el llenguatge utilitzat per programar el projecte ha sigut C#, un llenguatge de programació orientat a objectes molt similar a Java. Aquest es el llenguatge amb el que funciona Unity, fa uns anys també acceptava Javascript però en les versions mes actuals ja no.

4.2 Requeriments tècnics

Per ambdues eines, Unity i Aseprite, cal tenir un sistema operatiu Windows, Linux o OSX.
Les especificacions mínimes detallades son les següents:

Unity

OS	Windows 7 SP1+, 8, 10 (64 bit only) macOS 10.12+ Ubuntu 16.04, 18.04. CentOS7
CPU	SSE2 instruction set suport
GPU	Tarjeta gràfica amb DX10 (shader model 4.0)

[6]

Aseprite

OS	Windows Vista macOS 10.9 Ubuntu 14.04
CPU	Intel Core 2 Duo E8400
GPU	NVIDIA GeForce 510
RAM	128MB

[7]

La terminal feta servir per el desenvolupament es un Macbook Pro amb les especificacions següents:

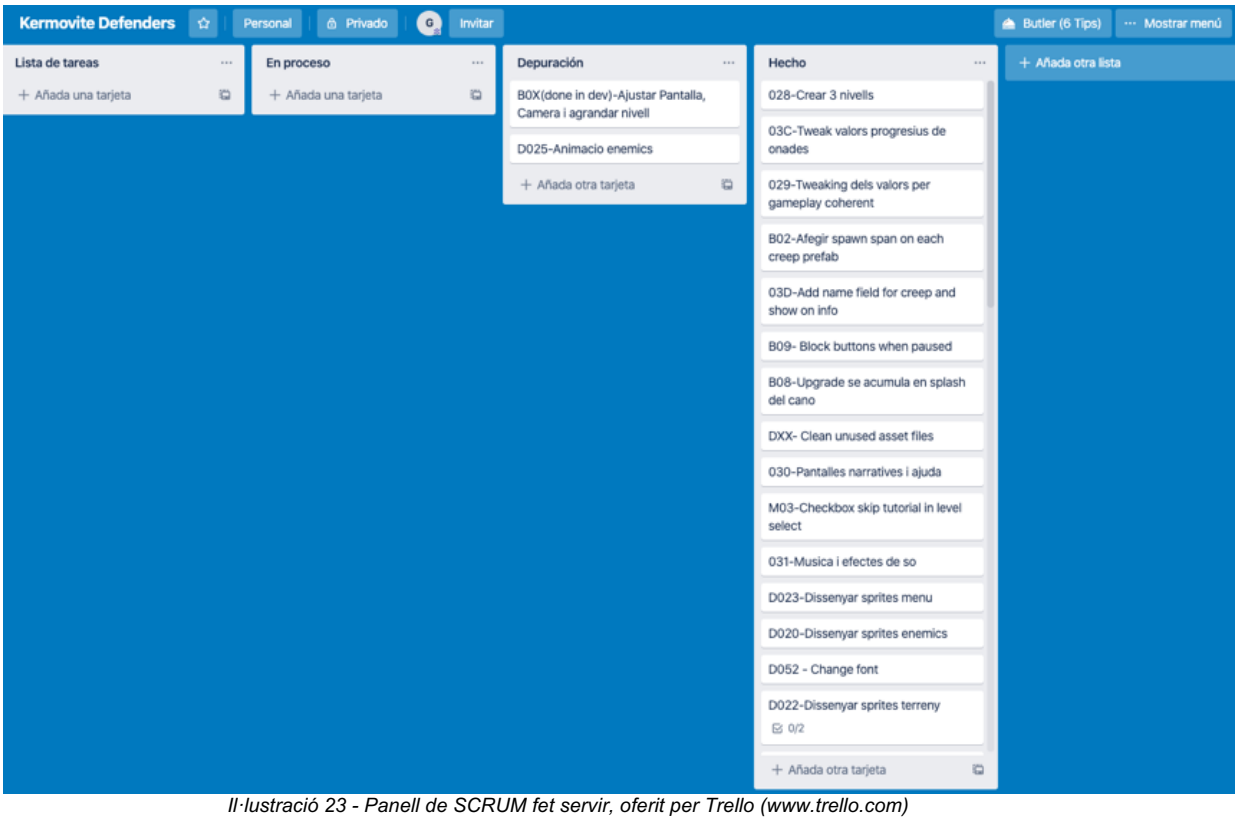
OS	macOS Catalina 10.15.1
CPU	2,4GHz Intel Core i9 8 core
GPU	AMD Radeon Pro 5500M 8 GB
RAM	32 GB 2667 MHz DDR4

4.3 Control de versions

Per desenvolupar el projecte d'una manera ordenada he fet servir el control de versions Git junt amb l'eina Trello, que proporciona un panell de SCRUM. La metodologia seguida es la següent:

En primer lloc he creat totes les tasques plantejades al panell de SCRUM. Cadascuna ve etiquetada amb un codi de text que combina text i número bassant-me en uns criteris:

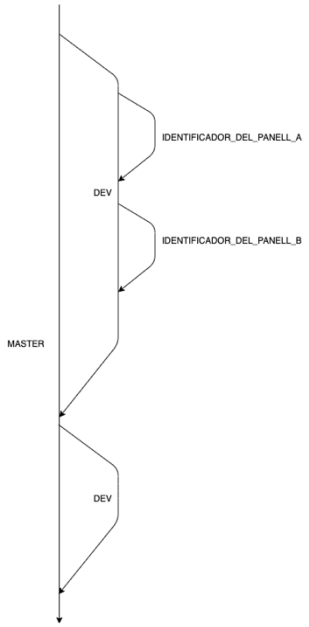
- Número de 3 xifres XXX: Tasca normal, el número es aleatori però mes o menys ve determinat per la seva data de creació.
- Número de 3 xifres seguit d'una lletra: Sub-tasca derivat d'una tasca normal, el numero determina de quina tasca deriva. Aquestes tasques sorgien sobre la marxa quan una era més complexe.
- D + número de 3 xifres: Tasca de disseny
- B + número de 3 xifres: **Bug** que ha sortit sobre la marxa. El número no indica la tasca de la qual deriva sinó que segueix el mateix criteri que el de les tasques normals
- K + número de 3 xifres: He creat aquest grup per separar les tasques relacionades amb el reactor i els minerals, per la complexitat de tasques i dependència entre elles.



Il·lustració 23 - Panell de SCRUM fet servir, oferit per Trello (www.trello.com)

Això s'ha creat per portar el control dels canvis a git. La manera de treballar es la següent, primer es puja una base a la branca master, que es el producte final. D'aquesta es fa una branca anomenada dev, que es l'entorn de desenvolupament. Cada vegada que es comença una tasca es crea una branca a partir de dev, aquesta branca fa servir l'identificador del panell com a nom. Quan s'acaba i es proba la tasca s'incorporen els canvis a la branca dev. Quan hi ha una entrega, després de provar que tot funcioni a dev s'incorporen els canvis a la branca master.

Això te molts beneficis. En primer lloc aïlla els canvis de la resta de branques. També permet revertir canvis concrets d'una tasca o eliminar un plantejament de com abordar un problema, sense afectar al producte final. També la combinació permet der un seguiment de tot el desenvolupament.



Il·lustració 24 - Esquema de branques del projecte

4.4 Assets

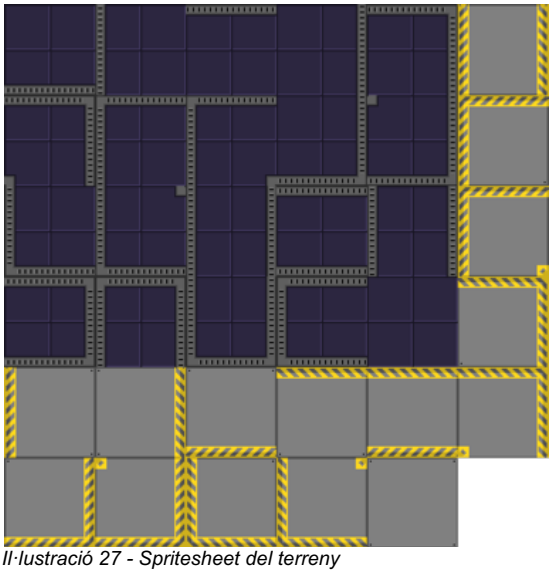
Packages de unity

Tot hi haver-hi moltes opcions hi haver probat algun paquet durant el desenvolupament, només s'ha fet servir un, Auto Letterbox. Aquest paquet soluciona un problema de camara que tenia, la camara de Unity no es fixe, i depenent de la resolució algunes parts del nivell no es veia. En aquest tipus de joc cal veure tot el terreny tota la estona com si es tractes d'un tauler de joc de taula, independentment de la mida de pantalla. Lo que fa Auto Letterbox es forçar una resolució concreta i incloure dues barres negres horitzontals quan la resolució no encaixi, com si fos una pel·licula. De fet aquest efecte de les pel·lícules s'anomena letterbox.

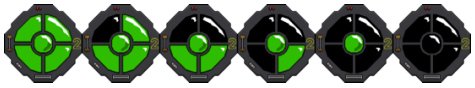
En un principi s'havia fet servir el paquet TextMesh Pro, que dona moltes opcions més pels textos. Però al final, degut a la senzillesa necessitada d'aquests en el producte final, ha sigut eliminat.

Spritesheets propis

Tots els **sprites** amb les seves animacions han sigut creats des-de 0 amb Aseprite. Els sprites son els de les torres, els enemics, el terreny, els minerals, el reactor, els botons i els panells de menú. Quan calia, han estat juntats en diferents spritesheets per tipus. Els sprites per enemics, terreny i torres son de 64x64.



Il·lustració 27 - Spritesheet del terreny



Il·lustració 26 - Spritesheet dels estats del reactor

Il·lustració 25 - Spritesheet dels minerals

Il·lustració 28 - Spritesheet amb les torres i la versio amb els canon per separat per fer rotacions

Il·lustració 29 - Spritesheet dels enemics i les seves animacions

Il·lustració 30 - Botó gran

Il·lustració 32 - Botó petit

Il·lustració 31 - Panell d'informació gran

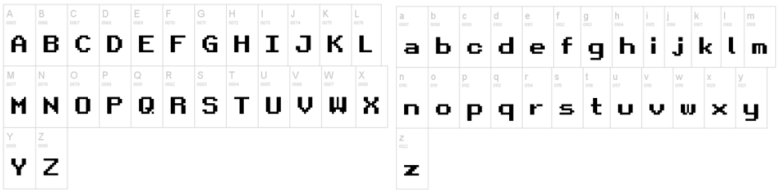
Il·lustració 33 - Panell d'informació petit

Imatges de tercers

S'han fet servir dos icones de tercers, ambdós per la funcionalitat de la interfície principal d'accelerar la partida o tornar-la a la velocitat normal. Aquests assets tenen la llicència creative commons [8][9]

Fonts

S'ha incorporat una font de tercers, Retro Gaming[10], ubicada a la carpeta fonts com a Retro Gaming.ttf. Aquest asset es lliure per us personal i comercial.



Il·lustració 34 - Font retro gaming

4.5 Arquitectura del joc

- Singleton: En primer lloc s'ha de mencionar la classe Singleton. Singleton es una implementació comú en moltes aplicacions programades amb orientació a objectes. Aquesta patró de disseny limita a una **instancia** a l'objecte que hereti directament d'ell. Això es útil per a casos on només hi haurà una instancia d'aquest objecte i assegura que quan s'accedeixen als mètodes o propietats d'aquest sigui la mateixa instancia sempre. A més d'això, a Unity es pot fer tediós accedir a les instancies, mentre que si son singletons es molt més fàcil i directe (i per tant menys propens a errors).

Per exemple, si es fa un videojoc on un objecte controla la música de fons (no hi haurà mai dos fent això a la vegada), lo òptim seria que hereti de Singleton. Si es volgues accedir a un mètode de la instancia que estigui reproduint la música es faria simplement amb

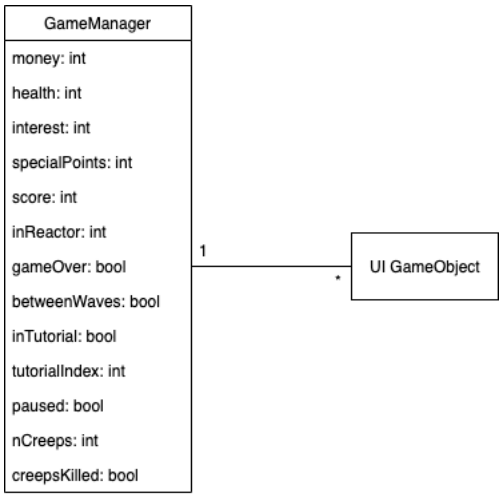
NomDeLaClasseControladoraDeMusica.Instance.Funcio();

En canvi, per una classe normal això seria:

GameObject.find("nomObjecteAmbScript").GetComponent<NomDeLaClasseControladoraDeMusica>().Funcio();

També es podria emmagatzemant en una variable amb un **camp serialitzat** ([SerializeField]) i arrastrant l'objecte amb el script de la classe, amb això estalviaria la part de GameObject.find("nomObjecteAmbScript") però no la resta. Com es pot veure les altres opcions son innecessàriament complicades i més propenses a errors, sobretot si es una instància a la que s'ha d'accedir constantment.

- GameManager: Controla la informació de la partida, com la puntuació, guanyar diners, pausar la partida, les funcionalitats de la interfície d'usuari, les pantalles del tutorial, etc. Es un Singleton i la classe més utilitzada de tot el projecte. Una al·legoria que encaixaria amb el seu paper seria dir que fa de "director d'orquestra" amb la resta de classes. Aqui es pot veure una representació de la classe GameManager amb les seves propietats principals.



Il·lustració 35 - UML simplificat de la classe GameManager

- LevelManager: Un altre Singleton, però aquest te una tasca més concreta. S'encarrega de les operacions relacionades amb les caselles del nivell, des de registar-les fins a portar el control sobre elles o calcular distàncies. També gestiona la posició de l'inici i final del camí que fan els enemics.

- Pointer: Aquesta classe es fa servir per mesurar coordenades en el mapa d'una casella d'una forma més senzilla. Les coordenades son simplificades a números enters. També es fa servir per determinar en quina casella està una torre o un enemic.

- TileScript: Gestiona la informació d'una casella, com si es pot construir o no, si es pot caminar o si te una torre construïda. Gestiona també la construcció de torres sobre la casella i les instancia.

- Ax (Algorisme A*): Aquest només existeix per el mètode implementat de l'algorisme A* adaptat al joc, però millor entraré en detall en el proper punt quan parli de la IA dels enemics (4.4).

- Node: Feta servir per representar les caselles a l'algorisme A*. Els camins a seguir pels enemics es guarden en una cua de Nodes creada per l'algorisme.

- Spawn: Classe molt sencilla amb una propietat del tipus Pointer que marca un punt del mapa, principalment feta servir per l'inici però finalment també pel final del recorregut a fer pels enemics.

- Reactor: Gestiona la part visual del reactor on volen arribar els enemics, la informació dels minerals que guarda i conté el GameObject del mineral que serà instanciat pels enemics. També s'encarrega del sistema de col·lisions quan un enemic arriba al reactor i si aquest rep un mineral o no. No s'encarrega de la vida del jugador ni si perd o no la partida ja que això ho fa el GameManager, però es comunica amb ell per fer això.

- TowerBtn: Tot i que la resta de les funcionalitats de la interfície d'usuari la gestiona la classe GameManager, la part del botons de compra de torres ho fa aquesta classe, ja que es una mica més complexa. Aquesta classe principalment controla la informació de la torre que es construirà, si el jugador la selecciona i les imatges de la torre i rang que mostren la posició on es construirà.

- Hover: Complementaria a la classe anterior, aquesta classe gestiona la funcionalitat del mouse que mostra la imatge de la torre a construir, la funcionalitat del input per part del usuari i si es cancel·la la selecció. Aquesta classe també e un Singleton ja que només existeix una instancia a la vegada fent això.

- Tower: Conté tota la informació de la torre, les seves propietats com atac o velocitat, quin tipus de torre es, els seus sprites o el seus preus. També controla coses com la rotació del canó o el so. La torre no infligeix dany a l'enemic, instancia un objecte amb la classe Projectile que descriuré a continuació. Tampoc detecta als enemics, d'això s'encarrega una altre classe, Range.

- Projectile: Aquesta classe gestiona l'objecte que colisiona amb l'enemic i infligeix el dany. La seva informació ve determinada per la torre que instancia l'objecte.

- Range: Gestiona les col·lisions dels enemics en un determinat rang al voltant de la torre. Aquests els guarda en una cua i indica l'objectiu per la torre. Quan un enemic surt de l'àrea de col·lisió també s'encarrega de eliminar-lo de la cua

- Creep: Es la classe dels objectes enemics. Conte la informació d'aquests com vida, velocitat, premi monetari per matar-lo o si te alguna característica especial. També gestiona el camí que ha de seguir, que l'obté de l'algorisme A* i el guarda com una pila de Nodes. Entre d'altres coses també guarda l'objecte del mineral, en cas de ser portador d'un. Quan la seva vida arriba a 0 es destruït i si porta el mineral l'instancia a la casella on es trobava, marcat per un Pointer.

- WaveManager: Un altre Singleton. Es l'encarregat de gestionar les onades, que es guarden en un vector del tipus Wave, que comentaré a continuació. Es l'encarregat també de determinar si s'han acabat les onades i d'incrementar la dificultat d'aquestes.

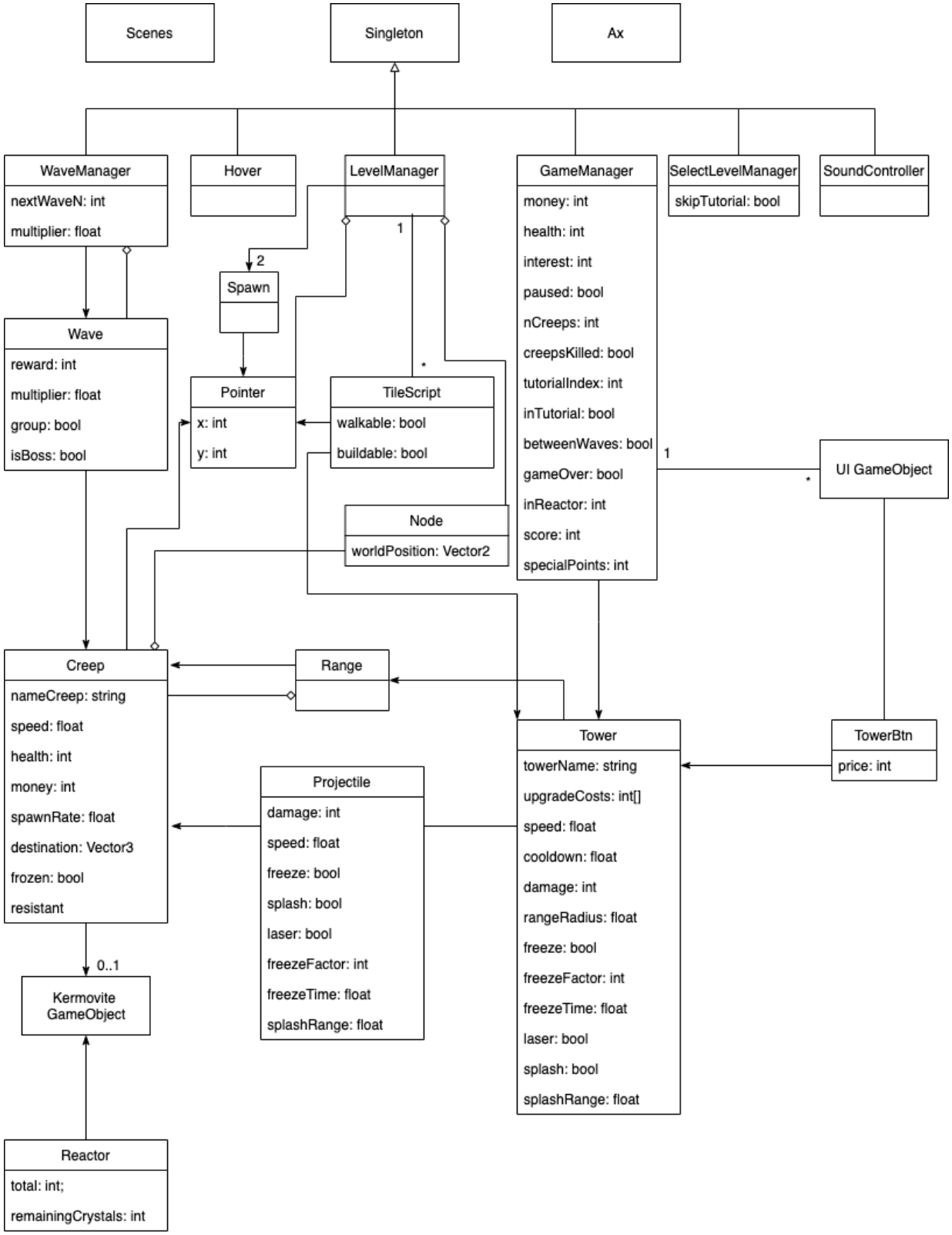
- Wave: Aquesta classe es un Scriptable Object de Unity. Aquesta es una forma eficient de emmagatzemar diferents dades en scripts quan la classe només conté propietats i hi hauran diferents variants de valors (en aquest cas hi hauran moltes onades amb valors diferents). Aquesta classe conte informació com el preu monetari, la **prefab** de l'enemic, el premi monetari per cadascun o si es un enemic especial o no.

- SoundController: Controla la música de fons. Com és el mateix cas de l'exemple exposat per Singleton, aquest òbviament també es un Singleton.

- Scenes: Controla el canvi entre escenes de Unity, es a dir el canvi de pantalla al nivell, menú de selecció de nivell o menú principal.

- SelectLevelManager: Està creat per futures implementacions però actualment només guarda la informació de si es selecciona saltarse el tutorial quan comenci una partida. Aquesta classe conté la funcionalitat DontDestroyOnLoad(), que fa que l'objecte que conté el script no es destrueixi entre escenes, en aquest cas guarda les opcions de la selecció del nivell per ser usat a la partida.

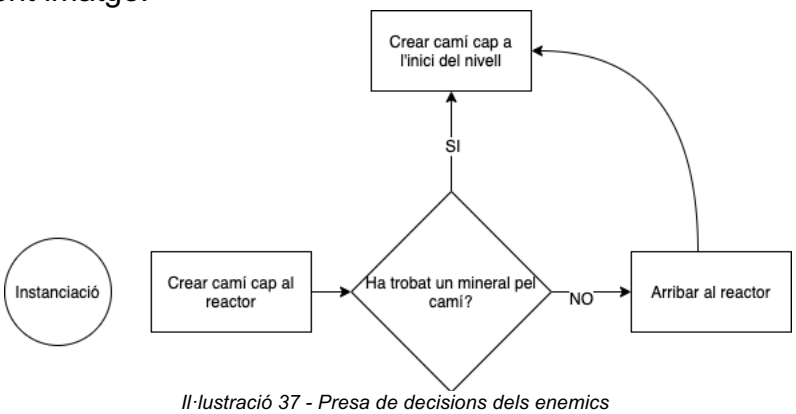
A la següent pàgina es mostra un esquema UML resumit de les diferentes classes i les seves relacions:



Il·lustració 36 - Esquema de relacions a alt nivell de les classes del projecte. No s'han incorporat atributs que serveixen simplement com a utilitat per alguns métodes.

4.6 IA dels enemics

Els enemics fan servir l'algorisme A* per trobar el camí més ràpid al reactor o a l'inici del nivell, depenent del cas que es necessiti com indica la següent imatge:



Il·lustració 37 - Presa de decisions dels enemics

Cada casella està guardada al LevelManager i porta un script de Node amb les seves coordenades. L'algorisme va comprovant els nodes i els seus veïns. Els nodes comprovats formen part d'una llista, la llista tancada, i els no comprovats es van afegint a una llista oberta. Els nodes es comproben amb la funció

$$f(n) = g(n) + h(n)$$

On:

- n es el node a comprovar
- g es la distància des de el node inicial fins a n (els adjacents tenen distància 10 i els que estan en diagonal 14)
- h es un càlcul heurístic de la distància al destí, (la distància en línia recta)
- f(n) es el resultat de la funció, i es el càlcul que es pren com a determinant amb quin node seguir

Aquest càlcul per node i veïns es repeteix fins trobar el destí.
Apart de la forma bàsica de A* a l'algorisme se li ha afegit el factor de les caselles per les quals es pot caminar i per les quals no, el valor booleà *walkable* de la classe TileScript.

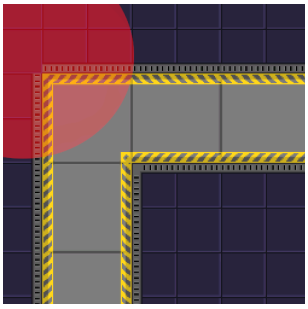
5. Disseny dels Nivells

5.1 Distribució dels mapes i la seva relació amb la dificultat

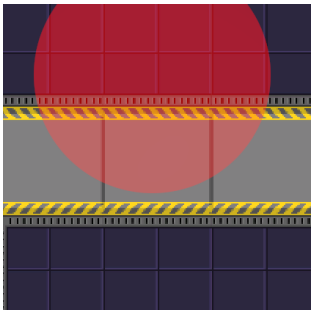
Tots els nivells tenen una funcionalitat casi idèntica, l'únic que varia es el camí que segueixen el camí, depenent d'aquest la partida serà més fàcil o més difícil.

En concret, n'hi ha dos factors que determinaran la dificultat. En primer lloc la quantitat de caselles que ha de recórrer un enemic. Això te una explicació simple, quan més estona estigui un enemic caminant per el nivell més oportunitat es te de matar-lo. Per una altre banda el número de cantonades obertes també influeix molt. En una cantonada oberta el rang de les torres tenen accés a més caselles. Aquí amb vermell es marca amb un rang a quantes caselles accedeix el rang:

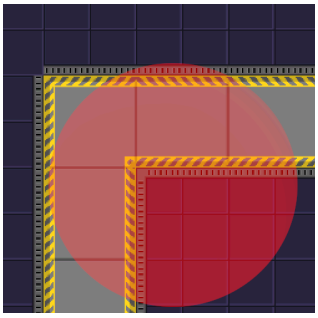
Cantonada tancada: 1



Recta: 3

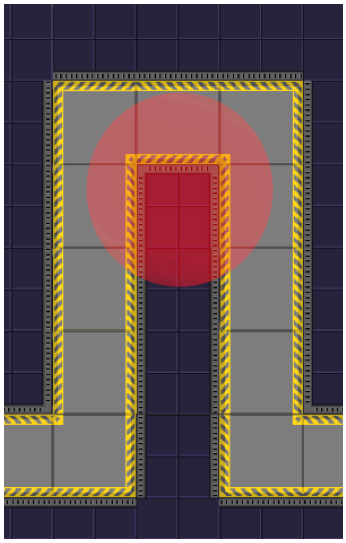


Cantonada Oberta: 5



N'hi ha un tipus més que no apareix als nivells fàcil i normals, és aquest tipus de forma, on s'aprofita encara més el rang:

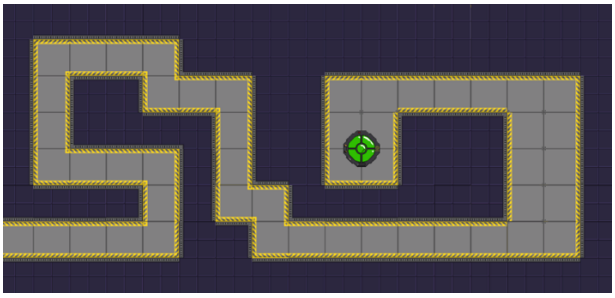
Número de caselles: 7



5.2 Guia dels nivells

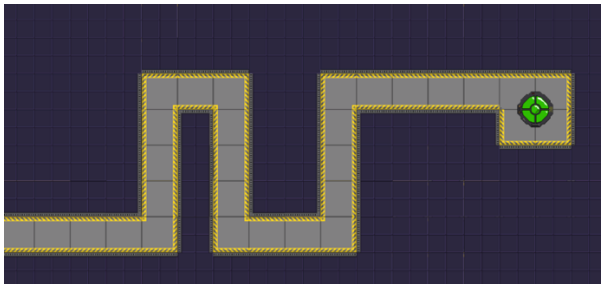
Nivell Fàcil

Recorregut dels enemics: 40 caselles
Cantonades obertes: 12
Aquest nivell es fàcil ja que te un recorregut molt llarg i moltes cantonades obertes.



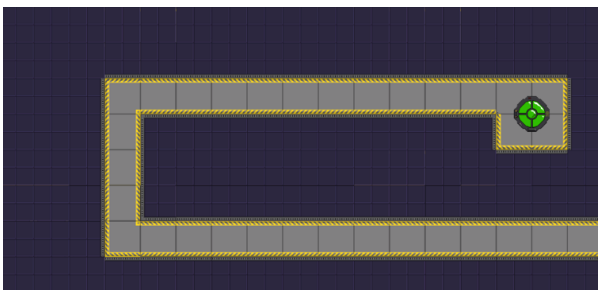
Nivell Normal

Recorregut dels enemics: 27 caselles
Cantonades obertes: 6
El nivell mes equilibrat, te 6 cantonades obertes i en dos parts d'elles es pot construir molt per aprofitar els dos costats



Nivell Difícil

Recorregut dels enemics: 29 caselles
Cantonades obertes: 3
La dificultat d'aquest es la manca de moltes cantonades obertes, només te dos i una que quasi no conta ja que està al costat del reactor. Aquí s'ha d'aprofitar el centre del mapa per construir torres amb molt rang i aprofitar els dos costats.



6. Manual d'usuari

Gracies per seleccionar Kermovite Defenders per PC i MAC

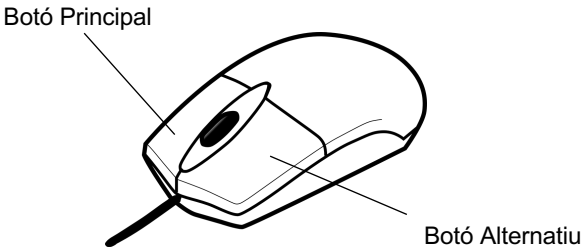
INTRODUCCIÓ

La nau que funciona amb un reactor del preciat mineral Kermovite ha sigut invaida per insectes de l'espai exterior, i venen a emportar-se els minerals! La teva missió es defensar el reactor i no deixar que se l'emportin o morirem tots. Construeix torres per defensar-lo. El destí de la nau està a les teves mans.

Si us plau, llegir aquest manual d'instruccions per assegurar una experiència adequada d'aquest joc, guardal per futures referències.

1

1. Controls



Botó Principal: Selecciona un element del menú o una torre
Botó alternatiu: Des selecciona qualsevol element seleccionat

2

3. Objectius

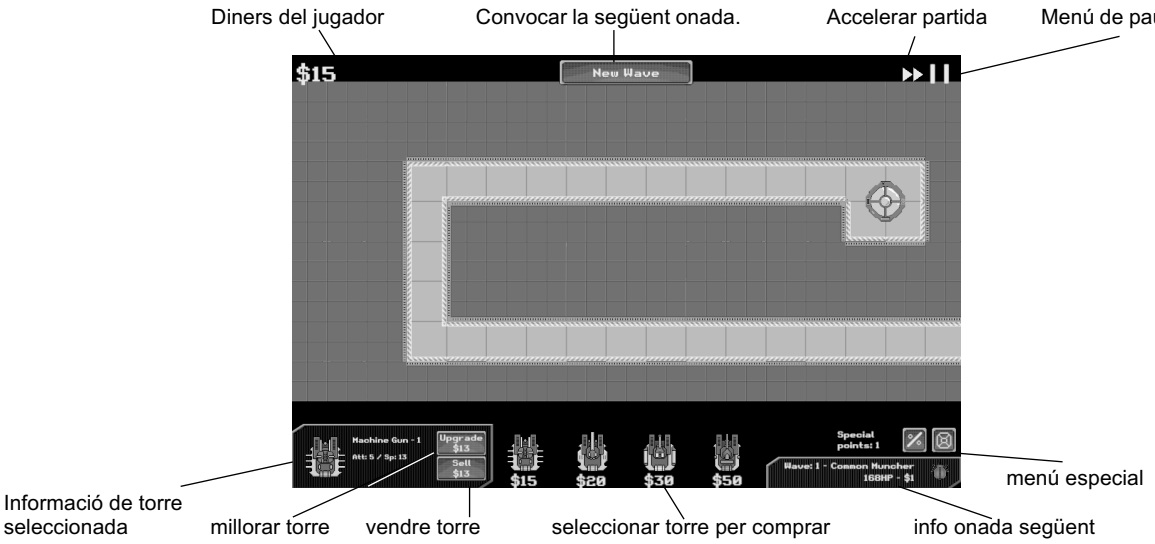
Els enemics sortiran des de una obertura en onades i intentaran arribar al reactor  . Si arriben a ell robaran un mineral  . Si surten amb els 5 perdràs la partida. Si moren pel camí el deixaran caure i un altre el podrà agafar. L'objectiu es sobreviure a totes les onades d'enemics. Això es farà construint torres als llocs disponibles del mapa, les caselles obscures.

4. Elements del joc

Torres: Cada una es construeix a una casella i ataca automàticament als enemics, n'hi ha de tres tipus, normals, amb dany d'àrea, congeladores i làser.
Diners: El jugador guanya diners matant enemics o al final un % d'interès al final de cada ronda, les torres valen diners
Enemics: Els enemics aniran a buscar el reactor o el primer mineral que trobin i fugiran amb ell. N'hi ha de 4 tipus, normals, grupals, ràpids i amb armadura

3

7. Pantalla de joc



4

9. Menú especial

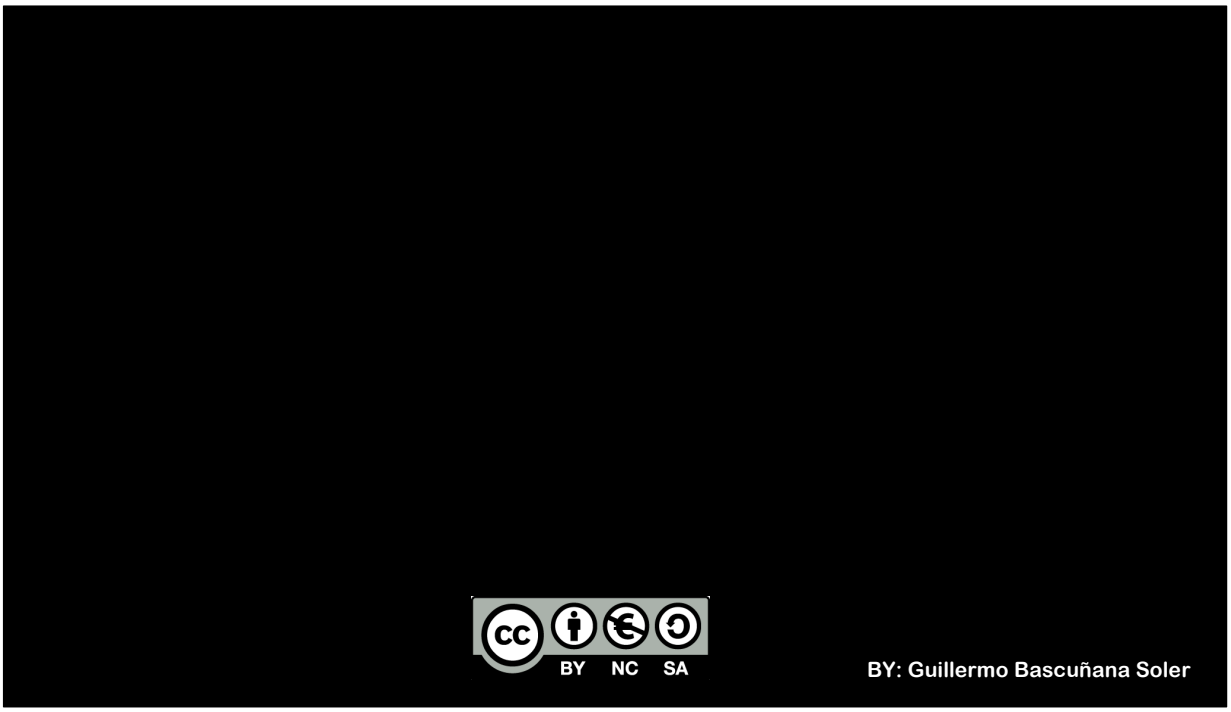
Cada vegada que es derroti a un enemic especial (Lord) s'obtindran punts especials, aquests es poden gastar en dues coses al menú especial

 Augmentar l'interés rebut al final del es onades un 1%

 Retornar els minerals disponibles al reactor

5

MEMO.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....



7. Conclusions

Aquest treball a servit per aprendre a com es el desenvolupament d'un videojoc. No només el funcionament de les eines, els factors i tots el processos, sinó també a fer-ho amb dades límit. Com he mencionat al principi sempre he volgut aprendre això i les dades límit era poder el que necessitava. Per poder portar-ho a terme he aplicat part de l'experiència que tenia en desenvolupament de software, però aquí hi ha agut casos particular i altres coses que s'han de tenir en compte. La organització del projecte i la seva planificació inicial es molt important per aconseguir un bon resultat.

També he aprés bastant sobre els tipus abstractes de dades o també anomenats estructures de dades. No només a fer servir coses com una cua o una pila en un entorn pràctic diferent al dels exercicis del grau sinó també la importància de quin escollir, perquè i les situacions en les quals un es millor opció que un altre.

En quant als objectius inicials, des de una ignorància de la magnitud que suposaria aquest projecte, ja que era la meva primera experiència, vaig voler fer el joc per portar-ho a tablet. Això ho vaig descartar les primeres setmanes. Però això no vol dir que quedi descartat, passat aquesta entrega penso atacar aquest problema.

Com ha crítica del resultat tinc de bones i dolentes. En primer lloc les bones. He aconseguit acabar el joc que tenia en ment, amb tota la seva funcionalitat planejada. El joc funciona i es jugable fins al final. També he de mencionar els punts negatius. A nivell de codi n'hi ha alguns problemes. En primer lloc, en alguns casos no he respectat algunes bones pràctiques de la programació orientada a objectes, principalment l'encapsulació, tot i que al mitjà i llarg termini si que ho he respectat al haver aprés una mica més. Per una altre banda no he respectat molt la modularitat entre classes. Moltes depenen d'altres innecessàriament, i això no es una bona pràctica. La part bona d'això es que per futurs projectes e aprés a evitar-ho. Des de el punt de vista de jugabilitat, encara van sortint alguns bugs i no he acabat de aconseguir un equilibri òptim de dificultat.

En quant a la planificació, crec que he complert amb tot lo planificat, més o menys en els temps establerts. Han anat sorgint altres tasques però excepte el port de tablet, totes les altres han estat completades.

Com a conclusió final extrec unes bones bases per crear un altre tipus de producte de software, i una programació des de un altre punt de vista al que estic acostumat. Se m'ha obert un nou mon conceptual de la mateixa manera que quan vaig aprendre a desenvolupar web. No només això, aquest projecte m'ha fet fixar-me en molts altres conceptes, que tenia com teòrics però realment no havia fet servir de forma pràctica.

8. Glossari

- **NES, Super Nintendo, GameBoy i la Nintendo 64:** Consoles de nintendo dels anys 90
- **Indie**: Independent, referint-se a tipus de joc es un joc desenvolupat per una persona o un grup petit sense el suport econòmic d'una gran empresa.
- **AAA**: Joc desenvolupat amb un pressupost molt gran
- **Jocs Flash**: Petits jocs executats al navegador web que feien servir la tecnologia flash d'Adobe
- **SCRUM**: Disciplina de l'organització de projectes, normalment aplicada a software, on es divideixen tasques en tasques petites i es gestionen amb un panell
- **OSX**: Sistema operatiu dels ordinador Apple
- **Game Loop**: Mecánica principal del joc que es va repetint o conjunt de rutines en les que es bassa la funcionalitat del joc
- **Mod**: Modificació del codi del joc per part de tercers per crear noves funcionalitats o aconseguir un joc diferent
- **Ensamblador**: Llenguatge de programació de baix nivell on es dona instruccions directament al processador
- **Engine**: Entorn de software desenvolupat per crear videojocs
- **Bug**: Error en el comportament del codi
- **Package**: Paquet de codi o recursos de tercers que es pot incorporar a un projecte de Unity
- **Sprites**: Gràfic d'ordinador pensat per ser mogut a la pantalla o manipulat
- **Instància**: En el mon del softwareopia d'un objecte o programa sent executat
- **Camp Serialitzat**: Funcionalitat de Unity que permet fer visible el contingut d'una variable per poder ser modificada
- **Prefab**: Objecte de Unity creat per l'usuari pensat per ser re-utilitzat

9. Bibliografia

- [1] https://metro.co.uk/2019/01/03/video-games-now-popular-music-movies-combined-8304980/ (28/12/19)
- [2] Revolución Indie, Jonathan León i Manu Delgado, ed. Heroes de Papel (pag 35-39)
- [3] https://en.wikipedia.org/wiki/List_of_best-selling_video_games (02/01/20)
- [4] https://en.wikipedia.org/wiki/Tower_defense (02/01/20)
- [5] https://en.wikipedia.org/wiki/Plants_vs._Zombies (03/01/20)
- [6] https://docs.unity3d.com/Manual/system-requirements.html (27/12/19)
- [7] https://www.pcgamebenchmark.com/aseprite-system-requirements (27/12/19)
- [8] https://en.wikipedia.org/wiki/File:Regular_polygon_3.svg (04/01/20)
- [9] https://en.wikipedia.org/wiki/Fast_forward (04/01/20)
- [10] https://www.dafont.com/retro-gaming.font (16/12/19)

- Música del joc: https://soundcloud.com/royaltyfreebackgroundmusic/creative-commons-music-4313?in=hugga-mein/sets/dark-ambient-eerie