"Pixel Wars": disseny i desenvolupament d'un videojoc d'estrategia multiplataforma

Xavier Collado Sànchez

Resum– PixelWars és un joc per a dispositius Android, IOS i PC del gènere Estratègia per torns, on el jugador haurà d'avançar per una sèrie de nivells en els quals haurà de vèncer l'exercit enemic. Per a vèncer, el jugador té a la seva disposició una sèrie d'unitats cadascuna amb habilitats i capacitats úniques, amb les quals podrà optar per eliminar totes les unitats enemigues o bé capturar el Quartell enemic. El jugador també haurà de tenir en compte les condicions del terreny al moment de planificar els seus atacs ja que cada terreny afecta d'una forma única a totes les unitats. El projecte ha sigut realitzat fent servir Java com a llenguatge de programació i LibGDX com a framework per a la visualització. El desenvolupament s'ha realitzat seguint els principis del Clean Code i la filosofia Test Driven Development per tal de permetre una modificació i un manteniment més senzill. Totes les funcionalitats han estat dissenyades pensant en els dispositius mòbils així com en els jugadors

Paraules clau– Java, LibGDX, CleanCode, 2D, Videojoc, mòvil, principis SOLID, patrons de disseny, TDD

Abstract– PixelWars is a game for Android, IOS and PC devices of the sort Strategy by turns, where the player will have to advance by a series of levels in which he will have to overcome the enemy army. To win, the player has at his disposal a series of units each with unique skills and abilities, with which he can choose to eliminate all enemy units or capture the enemy Barracks. The player must also take into account the terrain conditions when planning his attacks as each terrain affects all units in a unique way. The project has been carried out using Java as a programming language and LibGDX as a framework for visualization. The development has been carried out following the principles of the Clean Code and the Test Driven Development philosophy in order to allow a simpler modification and maintenance. All features have been designed with mobile devices in mind as well as gamers

Keywords– Java, LibGDX, CleanCode, 2D, videogame, phone, SOLID principles, design patterns, TDD

1 Introducció

Aquest projecte consisteix en el desenvolupament de un videojoc per a dispositius Android, IOS i PC. Aquest ha de complir també amb el requeriments recollits de 3 StakeHolders[9] que seràn els responsable finals per decidir si el projecte és la qualitat necesaria per afirmar que ha estat entregat amb exit.

- E-mail de contacte: xavier.collados@e-campus.uab.cat
- Menció realitzada: Enginyeria del Software
- Treball tutoritzat per: Ernest Valveny (departament)
- Curs 2019/20

1.1 Origen i Motivació

1.2 Objectius

- Desenvolupar un videojoc per a mòbil del gènere estrategia per torns.
- Implementar el joc amb totes les funcionalitats descrites en la proposta del TFG
- Entregar un producte final que sigui aprovat pels StakeHolders
- Implementar un software d'alta qualitat, seguint i aplicant els principis SOLID[8], bones pràctiques i patrons de disseny
- Desenvolupar el codi aplicant la filosofia TestDriven-Developement en tot moment, tenint així un producte final ben testejat amb diferents tipus de tests

1.3 Estat de l'art

Els jocs d'estratègia per torns tenen el seu origen en la transformació de jocs clàssics de taula al nou món dels videojocs. Alguns dels primers exemples de videojocs d'estratègia per torn pot ser Battle Chess per al dispositiu Atari-ST [1][2]. Aquest és un clar exemple de la idea de transportar els jocs de taula als videojocs on el seu reclam pels jugadors era la disponibilitat d'animacions específiques per cada unitat generant una experiència única pels jugadors.

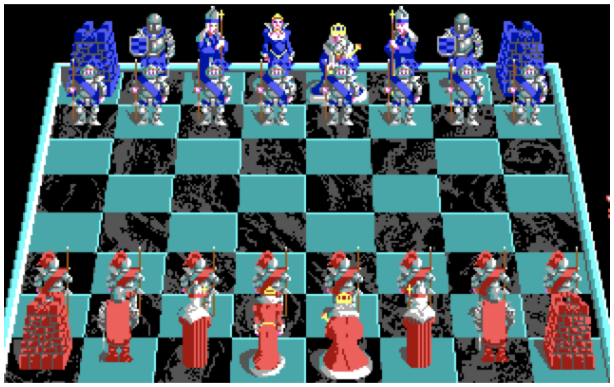

Fig. 1: Captura del videojoc Battle Chess (1988)

Amb el pas del temps els jocs d'estratègia per torns van anar evolucionant creant noves modalitats de joc més complexes. Un exemple de la seva evolució és la saga de videojocs Civilization[3]. Civilization requeria una gran planificació i estratègia, gestió de recursos, ciència, diplomàcia, dots militars... un aspecte realment destacable tot i les limitacions gràfiques de l'època. Els jocs han seguit evolucionant amb el llarg dels anys millorant sobretot en l'apartat gràfic tot i que la majoria de mecàniques de joc segueixen seguint els mateixos dissenys que temps enrere.

Fig. 2: Captura del videojoc Civilization (1991)

Respecte a l'oferta de jocs d'estratègia per torns per a dispositius mòbils la situació en el mercat és molt variada. A causa de la gran quantitat d'estudis desenvolupadors i d'eines que faciliten el desenvolupament de videojocs aquest mercat disposa d'una immensa oferta. Per contra, el fet que el mercat de desenvolupament per a mòbil sigui més obert a petits desenvolupadors genera que la qualitat d'aquests jocs oferits no sigui uniforme, tot i que existeixen exemples de gran qualitat com és el cas de Might and Magic[4].

Així doncs podem concloure que l'estat dels videojocs d'estratègia per torns segueix estant en actiu, disposant d'una gran oferta en quasi totes les plataformes del mercat. El

Fig. 3: Captura del videojoc CivilizationVI (2016)

Fig. 4: Captura del videojoc Might and Magic: Chess Royale (2019)

perfil del jugador sol ser un més especialitzat i familiaritzat en aquesta categoria de videojocs.

2 METODOLOGIA

La metodologia utilitzada pel desenvolupament del projecte és Kanban[5]. Kanban és un mètode Àgil[6] per gestionar el treball amb èmfasi en l'entrega just a temps, evitant la sobrecàrrega dels membres de l'equip. Amb aquesta metodologia el procés de desenvolupament, desde la definició d'una tasca fins la seva entrega final, es troba definit en un taulell (a partir d'ara BOARD). El BOARD del projecte es defineix de la següent manera:

- BACKLOG (tasques a realitzar): En aquesta columna es troben totes les tasques a realitzar. Cadascuna té una definició clara del problema a solucionar, exposant amb quins mòduls del projecte té relacions de dependència.
- DISCOVER (Quin problema estic resolent?): En cas de necesitar ampliar la descripció de la tasca per iniciar-la realitzo una investigació (SPIKE) amb límit màxim de 1,5h.
- DESIGN (Com resolem el problema per tenir èxit?): Presento una possible solució. Sempre intento tenir un mapa clar pel desenvolupament (HOW TO) definit abans de saltar al codi directament. Així tindré l'arquitectura present en tot moment del desenvolupament. Aquest HOW TO és afegit a la definició de la tasca a desenvolupar

- DEVELOP AND TEST (Crear test i itinerar fins aconseguir èxit): Fase en la qual desenvolupo el codi de la tasca agafada. Ho realitzo seguint la filosofía TDD. Això ho faig per assegurar-me que sols desenvolupo aquella funcionalitat en la qual estic treballant sense generar incidències o males pràctiques que puguin derivar en futurs BUGS o ISSUES. Pel testeig, realitzo un testeig seguint la metodologia TDD executant tots els test automatitzats

- RELEASE (Resolució pública de la solució proposada): Integrar la nova funcionalitat al BRANCH final.

- EVALUATE (Determinar si l'èxit ha estat assolit): Realitzo un testeig seguint la filosofia PLAY-WELL PLAY-BAD en cas de que sigui possible.

- DONE (Tasques realitzades): Recollirà totes les tasques finalitzades del projecte

2.1 Disseny

El disseny de les classes s'ha implementat seguint les bones pràctiques de Clean Code[7] i de SOLID principles[8]. Aquest bon disseny ha generat com a resultat un software estable, fàcil de mantenir i fàcilment extensible. Aquests factors són clau per a la vida del videojoc podent així crear nous nivells o unitats de qualsevol mena (Unitat, Terreny o Edifici) sense cap dificultat podent així entregar actualitzacions constantment a la comunitat de jugadors.

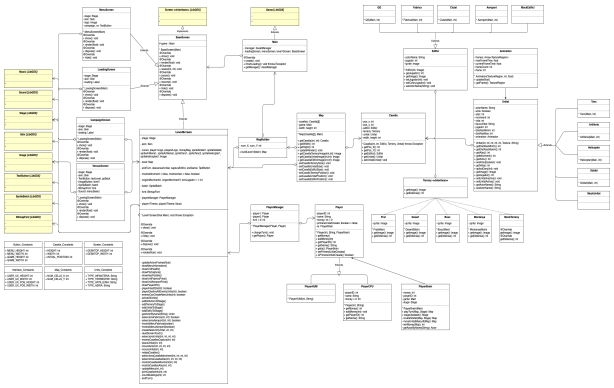

Fig. 5: Diagrama de clases

2.2 Implementació

Per a la implementació del codi he marcat com a referències les pautes de bones pràctiques de programació mostrades en el llibre de CleanCode[7] així com he intentat utilitzar els principis SOLID[8] en tot moment. Per últim, per tal de asegurar que el codi que estava sent implementat complia correctament amb les funcionalitats sol·licitades pels Stakeholders[9] i no afegia lògica de més he intentat seguir la filosofia de desenvolupament Test Driven Development[10]. Aquest filosofia consisteix en la implementació de tests en primer lloc abans de la implementació de la funcionalitat sol·licitada. Aquests tests fallaran fins que la funcionalitat desenvolupada estigui finalitzada correctament assegurant així que el codi queda cobert.

2.3 Validació

Com he comentat en l'apartat d'objectius, aquest projecte ha estat validat i seguit per un grup de 3 Stakeholders[9]. Han estat seleccionats intentant agrupar una variació de perfils de possibles usuaris finals. Els components de l'equip de Stakeholders han estat:

- Sergio: treballa com a enginyer informàtic. En el passat ha treballat en el desenvolupament de videojocs per a la distribuïdora de ElectronicArts (EA)[12] tot i que actualment ja no es dedica al desenvolupament de videojocs. També és un jugador de videojocs en quasi totes les plataformes disponibles al mercat. El seu perfil ha estat seleccionat com un perfil de Especialtista en videojocs.

- Fran: treballa com a professor d'història. No té coneixement sobre com és el desenvolupament de videojocs. També és un jugador usual en molts tipus de plataformes i estils de videojocs. El seu perfil ha estat seleccionat com un perfil de jugador Crític en videojocs.

- Irene: treballa com a directiva de comptes en l'agència de publicitat McCann. No té cap coneixement sobre el desenvolupament de videojocs. Tampoc té una gran experiència jugant, tan sols ha provat alguna plataforma, centrant-se sobretot en els jocs de mòbil. El seu perfil ha estat seleccionat com un perfil de jugadora Casual.

Durant el procés es van acordar 4 reunions (una inicial, 2 de seguiment i 1 de validació final) mitjançant la plataforma de videoconferències Zoom. En aquestes es presentava als Stakeholders l'estat del projecte demostrant que el desenvolupament seguia la planificació acordada en la reunió inicial així com es comentaven possibles incidències. També s'utilitzava aquestes reunions per a la discussió de possibles nous canvis proposats pels Stakeholders.

Fig. 6: Acció moure unitat del prototip

3 DESENVOLUPAMENT

3.1 Arquitectura

En aquesta secció em disposo a exposar més detalladament quin ha estat el disseny de classes i ressaltar els punts més importants.

3.1.1 Main

Aquesta és la classe inicial, en ella carreguem tots els assets (Sprites i àudios) necessaris pel correcte desenvolupament

del joc. Aquesta càrrega es realitza utilitzant un objecte proporcionat per LibGDX[13] anomenat AssetManager. Utilitzat l'AssetManager permet guardar tots els assets necessaris en memòria RAM en un inici reduint així considerablement els temps de càrrega durant una partida. Com que la primera vegada que l'AssetManager carrega tots els assets és més costosa, he desenvolupat una pantalla de càrrega que mostra en tot moment quin és l'estat donant així informació a l'usuari en tot moment.

Fig. 7: Captura durant la pantalla de carrega

3.1.2 LevelScreen

Aquesta classe és la responsable de la gestió de les partides. És la responsable de llegir els inputs de l'usuari i realitzar les accions pertinents. Consta d'una variable Map així com una variable PlayerManager (exposades a detall a continuació). En l'apartat 2.3 Visualització exposo amb més detall com es realitza la gestió de tot el front end en libGDX[13].

3.1.3 Map i MapBuilder

La classe Map és la responsable del coneixement i gestió de l'estat del taulell de joc. És qui sap com és l'estat de cadascuna de les caselles i és qui retorna o edita el Terreny, Edifici o Unitat demanat per la LevelScreen. Com totes les partides són gestionades per la LevelScreen he creat una classe MapBuilder. Com el seu nom indica aquesta classe és la responsable de passar-li el Map a utilitzar a la LevelScreen quan aquesta és creada. Així doncs en la classe MapBuilder definim tots els nivells disponibles dins del videojoc.

3.1.4 Casella

Aquesta és la unitat mínima del nostre taulell de joc. Consta de dues variables que defineixen la seva posició. Sempre consta d'una variable del tipus Terreny, aquesta mai pot ser nul·la. També pot disposar d'una variable Edifici i Unitat, limitades sempre a 1 per categoria, és a dir, en una mateixa casella mai podran existir dos Edificis o dues Unitats.

3.1.5 Terreny

És una interfície que disposa d'una sèrie d'implementacions, com són: Prat, Bosc, Desert i Muntanya. Cada Terreny disposa d'una variable Sprite (on guardem la imatge a mostrar) i una variable defensa. Aquesta última variable defensa s'utilitza al moment de realitzar un atac a una Unitat enemiga. Com major sigui la defensa del terreny menys efectiu serà aquest atac.

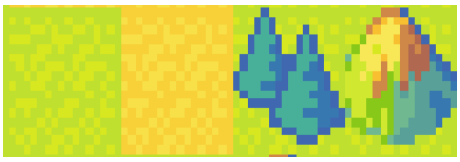

Fig. 8: Exemple del les 4 posibilitats de Terreny

3.1.6 Edifici

Aquesta és una classe del tipus abstracte. Consta d'una variable Jugador que s'utilitza per definir a qui pertany l'edifici, tant si és d'un jugador o encara es troba sense capturar. També té una variable PtsCaptura, que consisteix en 20 punts que es resten quan una Unitat comença l'acció de captura. Un cop han estat completament reduïts s'actualitza la variable Jugador i els PtsCaptura retornen al seu valor original de 20. Com la resta d'actors també consta d'una variable Sprite en la qual es guarda la imatge a mostrar. Per últim, com en Java no disposem d'apuntadors necessitem tenir una variable per poder guardar el nom de l'Actor que representa aquest Edifici dins l'escenari del Front end. És per aquest motiu que disposa també d'una variable actorName. Dintre dels edificis existeixen dos tipus que permeten

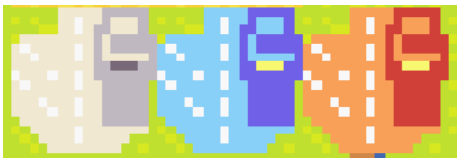

Fig. 9: Exemple del les 3 posibilitat per la clase Aeroport

la creació de noves unitats, aquest són la Fàbrica i l'Aeroport. El joc mostra un menu personalitzat de creació el qual té en compte l'economia del jugador per permetre o no la creació d'una unitat.

Fig. 10: Exemple del menu de creació de la Fabrica del jugador 1 i Aeroport del Jugador 2

3.1.7 Unitat

Aquesta és una classe del tipus abstracte. Consta de variables com són el moviment, l'atac, distancia màxima d'atac, distància mínima d'atac i el tipus d'unitat. Totes aquestes variables són utilitzades al moment de calcular les tres accions possibles d'una unitat: Moure, Atacar i Capturar (sempre que sigui Soldat). Les variables de distància màxima i mínima ens permet crear unitat d'atac indirecte, donant així encara més possibilitats per al jugador. Per últim també disposa d'un variable del tipus Animation, responsable de

la gestió dels Sprites així com una variable per guardar el nom del seu actor associat a l'escenari.

3.1.8 Player i PlayerManager

Per a la gestió dels jugadors he creat la classe PlayerManager. Aquesta és la responsable de retornar a la LevelScreen quin és el jugador segons el torn i també és la responsable de canviar de torn un cop la LevelScreen s'ho demani. Respecte a la classe Player disposem de dues extensions: PlayerHUM i PlayerCPU. En totes dues disposem d'un identificador, nom i economia (la qual anirà augmentant cada torn segons els edificis capturats). Per últim disposem d'una variable del tipus PlayerBrain què serà la nostra Intel·ligència Artificial utilitzada sempre que el jugador sigui del tipus PlayerCPU. Els jugadors del tipus PlayerHUM senzillament la té nul·la, ja que totes les dedicions són llegides dels inputs del jugador.

3.1.9 PlayerBrain

Com he exposat en el punt anterior, la classe PlayerBrain és on resideix la Intel·ligència Artificial del nostre joc. Aquesta classe rep el Map per part de la classe LevelScreen, realitzar tots els canvis que convinguin i retorna el nou Map, un cop retornat el nou Map passa el torn a l'altre jugador, donant així una sensació de jugador real. La Intel·ligència Artificial desenvolupada en aquest projecte és molt bàsica a causa del fet que no era l'objectiu principal d'aquest. Tot i això, gràcies al disseny modular la IA del joc pot seguir sent desenvolupada en un futur sense preocupació per trencar cap altre dels components del joc.

3.1.10 Guardar

Finalitzant l'exposició de l'apartat tècnic, també s'ha desenvolupat una lògica de guardat. Actualment un cop el jugador ha superat amb èxit un nivell de la modalitat CAMPAING queda registrat en la memòria del joc permetent així la selecció del pròxim nivell. Aquesta funcionalitat pot ser estesa en un futur permetent l'opció de guardar l'estat d'una partida sense finalitzar.

3.2 Visualització

La principal característica de LibGDX és la seva capacitat per a l'abstracció. Aquest framework és capaç de detectar en quin Hardware està executant-se el programa per derivar les diverses responsabilitats de l'execució del joc (Àudio, Imatge, Input....) a les parts del hardware responsable. Aquesta capacitat ens permet executar un únic projecte en plataformes com Android, IOS o PC sense haver d'escriure de nou cap línia de codi. Aquest és el motiu principal pel qual el desenvolupament s'ha realitzat utilitzant el framework LibGDX[13].

3.2.1 BaseScreen

L'arquitectura en la qual es basa libGDX[13] per mostrar i gestionar tots els assets en pantalla segueix una al·legoria de teatre. És a dir, libGDX[13] disposa de l'objecte interfície Screen el qual disposa de les funcions de show, render, resize, pause, resume, hide i dispose. Totes elles són

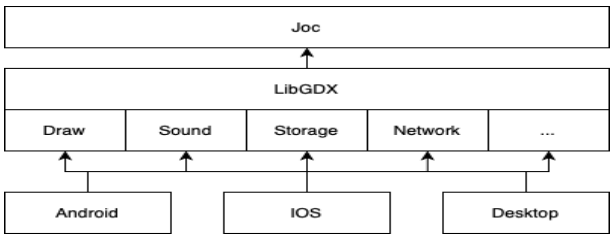

Fig. 11: Estructura general del framework LibGDX

bàsiques per a la gestió de tots els Actors que s'utilitzaran en cada escena. Respecte del projecte, tenim una classe BaseScreen que implementa a Screen. D'aquesta clase BaseScreen expandeixen totes les Screen utilitzades en el joc, com són: LoadingScreen, WelcomeScreen, CampaignScreen, VSScreen i LevelScreen. Com hem comentat anteriorment libGDX[13] una alegoria de teatre, en aquest cas el nostre escenari queda definit per la variable Stage dins de cada Screen. En aquesta Stage és on afegirem tots els nostres Actors, si un actor no és afegit a la Stage no apareixerà ni podrà ser interactuat per pantalla. Per últim disposem d'una gran varietat per als possibles actors, més específicament en el projecte hem utilitzat Actors del tipus Image i TextureButton.

3.2.2 Animation

Entre tots els possibles actors de les escenes del nostre joc, l'únic tipus que consta d'una animació són les classes Unitat. És per aquest motiu que cada extensió de la classe Unitat (Soldat, Tanc, Helicòpter...) consta d'una variable del tipus Animation. Aquesta classe Animation és la responsable de passar-li a la classe LevelScreen quin Sprite ha de ser mostrat a partir del temps d'execució del videojoc generant així una sensació de moviment per al jugador.

Fig. 12: Sprite d'animació del Soldat1

4 Resultats

A continuació exposo els resultats finals del desenvolupament. Podem veure el correcte desenvolupament de les dues modalitats de joc així com tota la funcionalitat relacionada amb les interfícies dels menús de selecció de nivells. Dins dels resultats finals també s'ha aconseguit la inclusió d'animacions així com efectes de so. Per últim també s'ha aconseguit una funcionalitat de guardar la partida.

5 Conclusions

A continuació exposo les conclusions extretes un cop finalitzat el projecte

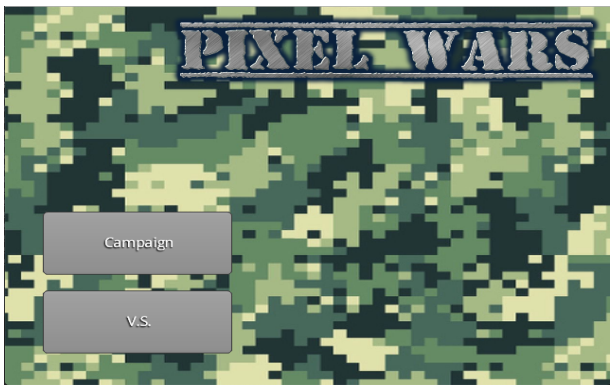

Fig. 13: Menu Inicial

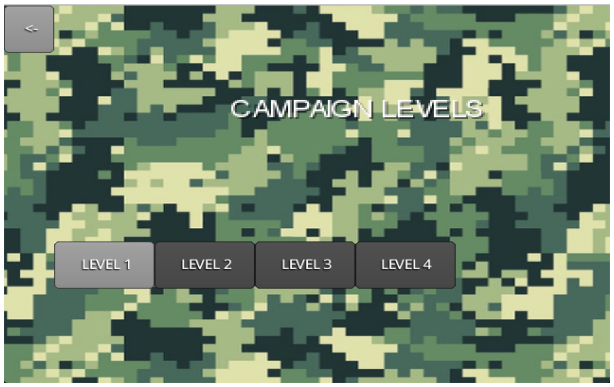

Fig. 14: Menu CAMPAIGN

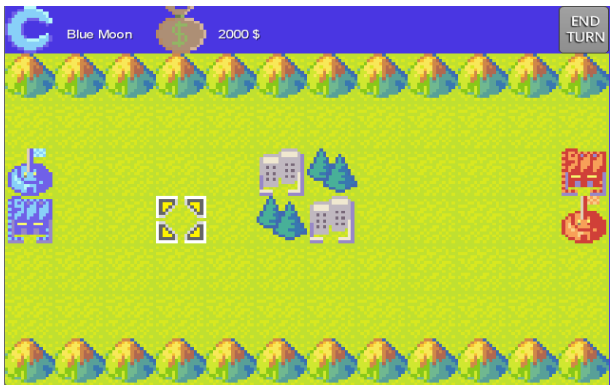

Fig. 15: Captura del nivell 1 de la modalitat CAMPAIGN

Fig. 16: Captura del nivell 1 de la modalitat V.S.

5.1 Obstacles i dificultats

La principal dificultat que he patit durant el desenvolupament ha estat el meu desconeixement respecte al framework libGDX[13]. Aquest fet ha generat un retard en el desenvolupament de totes les tasques planificades. També he patit un obstacle al moment de voler utilitzar la filosofia de TDD[10]. El fet d'utilitzar el framework de libGDX[13] no permet realitzar tests de forma senzilla a causa de totes les seves interrelacions de classes (Screen, Scene, Actor...), complicant així l'aplicació de la filosofia TDD[10]. Aquest bloqueig ha estat salvat utilitzant una sèrie de classes definides com a Mock (per exemple MockTerreny). Amb aquestes classes tipus Mock podia validar amb tests el seu correcte funcionament. Per últim també he hagut de replanificar les reunions amb els Stakeholders per culpa del confinament sent aquestes realitzades finalment mitjançant Zoom.

5.2 Resultat obtingut

Puc afirmar que el projecte ha tingut un resultat que compleixen tots els objectius plantejats a la concepció d'aquest. El joc és un joc per estratègia per torns compatible amb dispositius Android, IOS i PC a la vegada que compleix els estàndards de la indústria. L'aplicació consta d'una sèrie de test que permeten una major facilitat pel manteniment del codi. També consta de una estructura que segueix la filosofia de programació Clean Code, generant d'aquesta forma un codi estable, mantenible i extensible. Per últim també ha estat validat pels 3 StakeHolders que van definir i reorganitzar algunes funcionalitats al seu gust al llarg del desenvolupament del projecte.

5.3 Futur del projecte

Aquest projecte té una gran capacitat de futur. Encara es poden desenvolupar més tipus d'unitat o terreny. També es pot desenvolupar un possible mode de joc Online expandint així el joc al camp dels jocs competitius. Per últim es pot seguir treballant en l'art del joc, millorant l'estil afegint unes animacions de major qualitat facilitant així la immersió dels jugadors.

Respecte a la part tècnica del projecte, encara es pot treballar en la millora de la cobertura dels seus tests així com la seva Intel·ligència Artificial. Sobretot si es planteja obrir el joc al camp dels jocs competitius.

Agraïments

M'agradaria agrair la col·laboració de la Irene, el Fran i el Sergio en el rol de StakeHolders i trobar els espais per poder realitzar les entrevistes tot i els impediments patits pel confinament. En especial al Sergio, per tots els consells rebuts sobre bones pràctiques de programació i disseny aplicat als videojocs utilitzant LibGDX[13]. També vull agrair als usuaris Amyd i Zackie per haver alliberat els Sprites utilitzats en el desenvolupament d'aquest projecte al portal web de Sprite-resources [8]

Referències

[1] Atari ST https://es.wikipedia.org/wiki/Atari_ST

[2] Battle Chess https://es.wikipedia.org/wiki/Battle_Chess

[3] Civilization https://es.wikipedia.org/wiki/Civilization

[4] Might and Magic https://www.ubisoft.com/es-es/game/might-and-magic-chess-royale/?isSso=true&refreshStatus=noLoginData

[5] Kanban https://es.wikipedia.org/wiki/Kanban

[6] Àgil https://es.wikipedia.org/wiki/Desarrollo_\unhbox\voidb@x\bgroup\let\unhbox\voidb@x\setbox\@tempboxa\hbox{a\global\mathchardef\accent@spacefactor\spacefactor}\let\begingroup\endgroup\relax\let\ignorespaces\relax\accent19a\egroup\spacefactor\accent@spacefactorgil_de_software

[7] Clean Code https://es.wikipedia.org/wiki/Robert_C._Martin

[8] SOLID Principles https://es.wikipedia.org/wiki/SOLID

[9] StakeHolders https://es.wikipedia.org/wiki/Parte_interesada_(empresas)

[10] TestDriven Development https://es.wikipedia.org/wiki/Desarrollo_guiado_por_pruebas

[11] MarvelApp - Prototip https://marvelapp.com

[12] Electronics Arts (EA) https://es.wikipedia.org/wiki/Electronic_Arts

[13] LibGDX https://libgdx.badlogicgames.com

[14] www.spriters-resource.com

APÈNDIX

A.1 Diagrama de classes

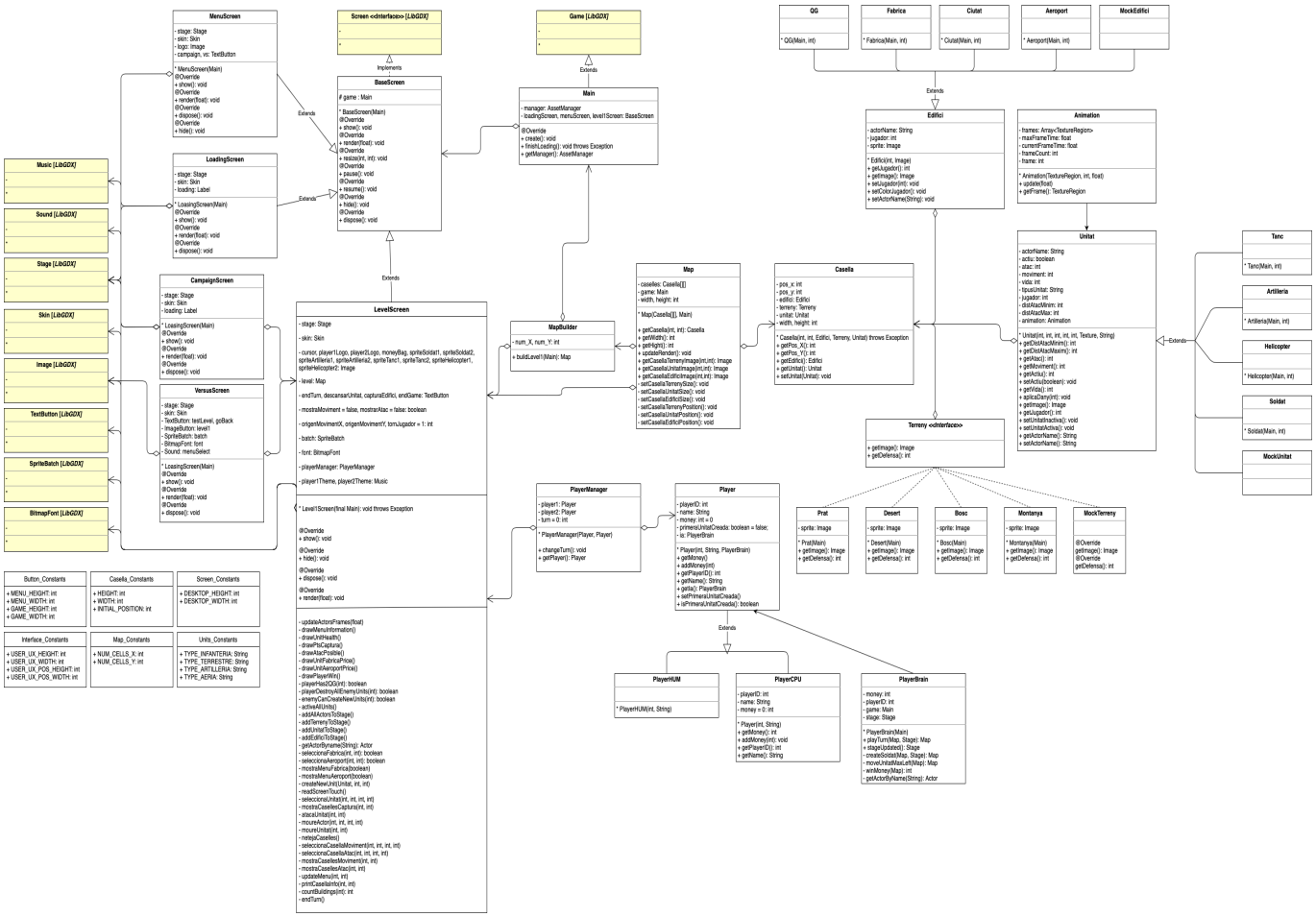

Fig. 17: Diagrama de classes del projecte

A.2 Diagrama de casos d'ús

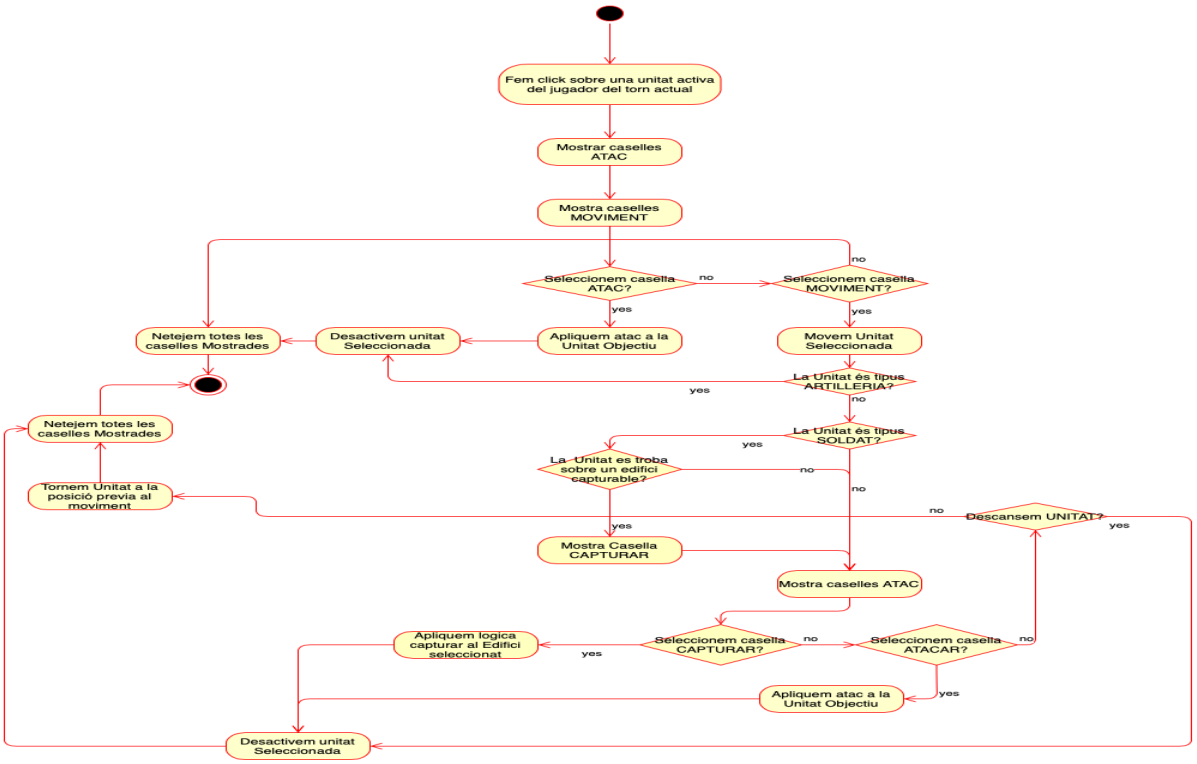

Fig. 18: Diagrama de casos d'ús de la lógica de selecció