Títol: Implementació de jocs de taula

Volum: 1 de 1

Alumne: Carles Boada Prats

Director/Ponent: Marc Vigo Anglada

Departament: LSI

Data: 19 de Gener de 2009

Índex

1. Introducció **5**
   - 1.1 Sator 5
   - 1.2 Arepo 6
   - 1.3 Tenet 7
   - 1.4 Motivació del projecte 8
   - 1.5 Objectius del projecte 9
     - 1.5.1 Progrés dels objectius 9
2. Planificació i Eines de desenvolupament **11**
   - 2.1 Decisions inicials 11
   - 2.2 APIs i llibreries 11
   - 2.3 Eines de desenvolupament 12
     - 2.3.1 Xcode 2.0 12
     - 2.3.2 Terminal 12
   - 2.4 Llenguatge C++ 12
   - 2.5 Interfície gràfica 13
   - 2.6 Fase del projecte: especificació de l'aplicació 14
   - 2.7 Fase del projecte: disseny 14
   - 2.8 Fase del projecte: implementació 14
   - 2.9 Fase del projecte: IA 14
3. Especificació de l'aplicació **15**
   - 3.1 Programa principal 15
   - 3.2 Tauler i les peces 15
   - 3.3 Els jocs 15
     - 3.3.1 Sator 16
     - 3.3.2 Arepo 16
     - 3.3.3 Tenet 16
   - 3.4 IA 16
4. Disseny **17**
   - 4.1 Arquitectura de l'aplicació 17
   - 4.2 Diagrama de disseny 17
   - 4.3 L'objecte aplicació 18

4.3.1 La classe escena 18
4.3.2 L'aplicació principal 19
4.3.3 Les peces 19
4.3.4 El tauler 20
4.3.5 Els jocs 20
4.3.5.1 La classe Sator 20
4.3.5.2 La classe Arepo 22
4.3.5.3 La classe Tenet 22
4.3.5.3.1 La classe cartes 22
5. Implementació **25**
5.1 L'estat 25
5.2 El bucle de joc 27
5.2.1 Funcions Init 28
5.2.2 Funció hum 29
5.2.3 Funció IA 29
5.2.4 Funcions final 29
6. IA: Expansió de moviments **31**
6.1 Expansió de moviments 31
6.2 La funció Alfabeta 31
6.3 La funció eval 33
6.3.1 Sator 33
6.3.2 Arepo 34
6.3.3 Tenet 36
7. IA: Estratègies i perfils **37**
7.1 Estratègia 37
7.1.1 La funció escull_atac_múltiple 37
7.1.2 Tipus d'estratègies 39
7.1.2.1 Aleatòria 39
7.1.2.2 Millor 39
7.1.2.3 Agressiva 40
7.1.2.4 Defensiva 41
7.1.2.5 Bloquejant 41
7.1.2.6 Corredora 42

7.1.2.7 Trampera 42
7.1.2.8 Protectora 43
7.2 Perfil 44
7.2.1 Automàtic 44
7.2.2 De joc ràpid 45
7.2.3 Agressiu 45
7.2.4 Defensiu 45
7.2.5 Aleatori 45
8. Joc de proves **47**
8.1 Entorn de proves 47
8.2 Resultats obtinguts 47
8.2.1 Perfil automàtic 47
8.2.2 Perfil de joc ràpid 48
8.2.3 Perfil agressiu 48
8.2.4 Perfil defensiu 48
8.2.5 Perfil aleatori 49
9. Planificació **51**
9.1 Planificació inicial 51
9.2 Planificació real 52
10. Conclusions **53**
10.1 Valoració de l'aplicació 53
10.2 Perspectives de futur 53
11. Bibliografia **55**

1. Introducció

En aquest apartat s'exposen les normes dels tres jocs creats per Oriol Comas i que s'han implementat: el Sator, l'Arepo i el Tenet. Per últim veurem cap a on s'enfocarà el projecte i perquè.

1.1 Sator

El Sator és un joc de tauler de 2 jugadors. Per jugar-hi cal un tauler de cinc per cinc caselles i cinc peces per cada jugador, de les quals tres tenen a la part de sota un punt blanc i dues un punt vermell.

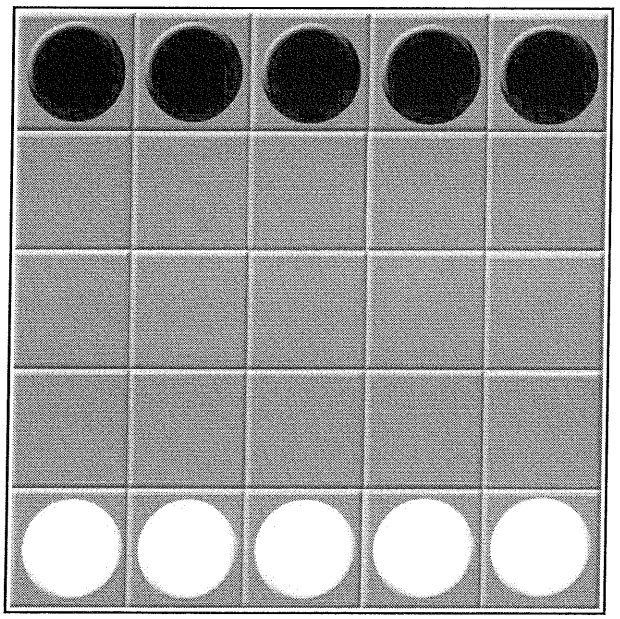

Fig. 1 Tauler del Sator

Per començar a jugar, el tauler s'ha de posar entre els 2 jugadors. Cada jugador col·loca a les peces del seu color amb els punts cara avall sense que l'altre jugador sàpiga si una peça té un punt blanc o un vermell.

Les peces es mouen i capturen les peces del jugador rival com les dames. La única diferència és que si una peça mata una altra però aquesta té un punt vermell a sota, es retiren del joc les dues peces. És obligatori moure peça, però no capturar-ne una altra. En el cas que no es pogués moure, l'altre jugador seguiria movent fins que ho pogués fer el primer jugador.

Comença el jugador amb les peces negres i es va jugant per torns fins que una peça amb un punt blanc a sota arribi a l'última casella. Si no es dóna el cas i ja no es pot moure cap altra peça, guanya el jugador té les peces més avançades.

1.2 Arepo

L'Arepo és un joc per a dos o quatre jugadors. Per jugar-hi, cal un tauler de set per set caselles i 20 fitxes de color, cinc per cada jugador (del mateix color).

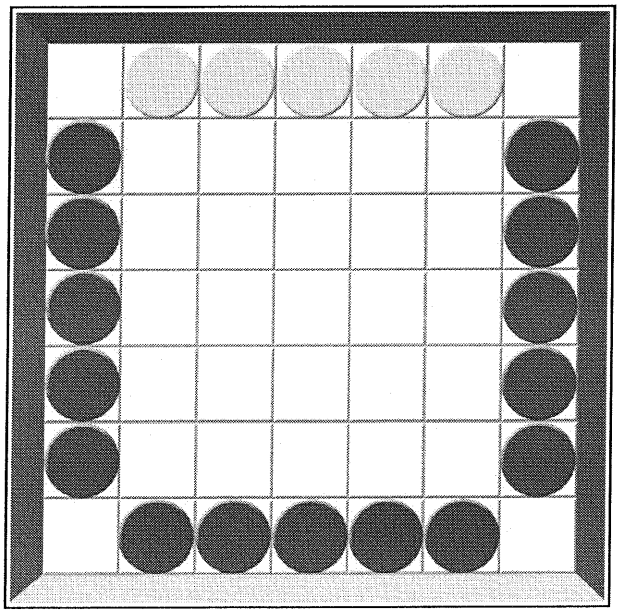

Fig. 2 Tauler de l'Arepo

El tauler es posa al mig de la taula, de manera que cada jugador tingui com a costat oposat el color de les seves fitxes. Cada jugador agafa les cinc fitxes d'un color i les col·loca a les cinc caselles centrals de la línia que té davant seu, una a cada casella.

A l'Arepo i poden jugar dos jugadors controlant dos grups de peces encarats cadascun o bé quatre jugadors controlant un grup de peces cadascun. Siguin dos o quatre jugadors, les peces es mouen i capturen com les dames. Un peça pot capturar-ne una altra del mateix equip però no del mateix color.

Es tria a l'atzar quin jugador comença i després es juga per torns en el sentit de les agulles del rellotge. Moure peça es obligatori, si no es pot se salta el jugador. Es segueix jugant fins que una peça arriba a l'última casella en el cas de quatre jugadors o quan ja no es pot moure cap altra peça.

El jugador que guanya la partida és aquell que fa arribar una peça a l'última casella (en el cas de quatre jugadors) o el que fa arribar més peces a les últimes caselles en el cas de dos jugadors.

1.3 Tenet

El Tenet és un joc per dos jugadors. Per jugar-hi cal un tauler de cinc per cinc i cinc peces per a cada jugador: el rei, el cavall, l'alfil, la torre i la dama. Calen també quinze cartes diferents per a cada jugador: una pel rei, dues pel cavall, tres per l'alfil, quatre per la torre i cinc per la dama.

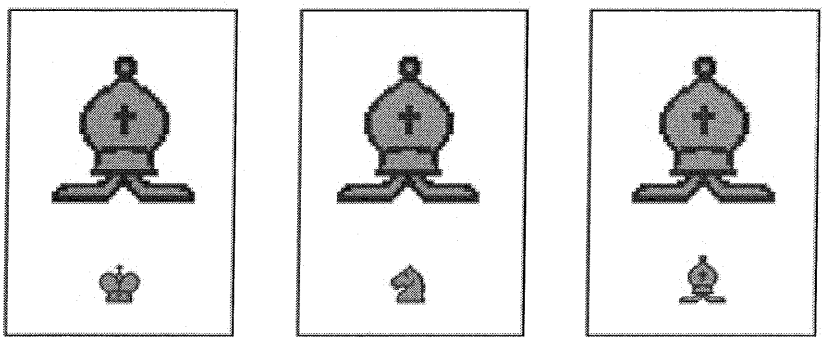

Fig. 3 Cartes del Tenet

Les cartes indiquen quina peça es pot moure i quin moviment pot fer. Cada peça pot fer el seu moviment als escacs i els de les peces anteriors (per exemple, les tres cartes Alfil negre permeten moure l'alfil negre una vegada com a rei, una com a cavall i una com a alfil).

El tauler es posa entre els dos jugadors. Cada jugador col·loca les cinc peces del seu color com s'indica a la Fig. 4. A més, cada jugador agafa les quinze cartes del seu color i les estén cara amunt a un costat del tauler, de manera que els dos jugadors les vegin bé.

Comença el jugador que té les fitxes negres i després es juga per torns. En cada un, el jugador juga la carta del seu color que vol, és a dir, la gira i mou la peça corresponent el moviment que indica la carta, cap a una casella buida o cap a una casella ocupada per una peça de l'adversari. Tots

els moviments són els de les peces d'escacs. Cada carta es pot jugar una única vegada i si es captura una peça i el jugador de la peça capturada encara en tenia cartes, les deixa al costat del tauler, com si ja les hagués fet servir.

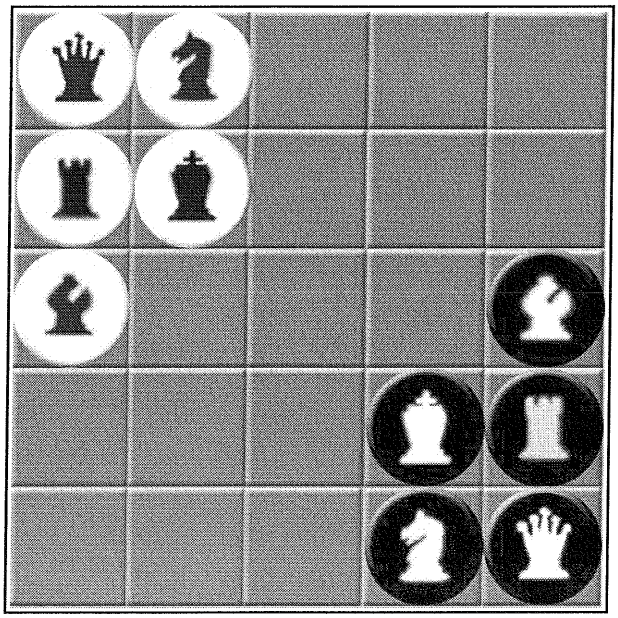

Fig. 4 Tauler del Tenet

Jugar és obligatori, però si un jugador no pot jugar perquè no li queden cartes vistes, l'altre segueix jugant mentre pugui. Si un jugador no pot jugar perquè les peces que pot moure estan bloquejades per altres peces, les podrà tornar a jugar quan es desbloquegin.

Si un jugador captura les peces de l'altre guanya directament la partida. En el cas que cap dels dos jugadors pugui moure cap peça, es sumen els valors de les peces (rei = 1, cavall = 2, alfil = 3, torre = 4 i dama = 5) i guanya el que te les peces amb més valor.

1.4 Perquè aquest projecte?

Els jocs de tauler sempre han estat molt populars. Això passa perquè son jocs amb normes molt senzilles i que no requereixen gaires materials per jugar, sempre podem substituir una peça que hem perdut per un tros de paper o aprofitar el tauler dels escacs per les dames per exemple.

El problema apareix quan un jugador no té amb qui jugar o bé, vol un repte major i jugar contra una màquina, com tantes vegades s'ha vist en els escacs. Aquesta necessitat és la que motiva aquest projecte.

1.5 Objectius del projecte

Els objectius del PFC són 2:

1. Implementar els 3 jocs per jugar-hi a l'ordinador.
2. Crear 3 IA[14], una per a cada joc, que permeti jugar a un jugador contra la màquina o la màquina contra si mateixa.

1.5.1 Progrés dels objectius

- Objectiu 1:
  - Crear un programa principal que s'encarregui de preguntar a l'usuari quin joc (i quina variant del joc si en té) vol executar.
  - Per a cada joc (i les seves variants), implementar un algorisme[15] que controli la inicialització del tauler i les peces, el desenvolupament del joc per torns i el final del joc.

- Objectiu 2:
  - Per a cada IA, conèixer el joc i saber quines jugades son millors en cada moment.
  - Quantificar cada moviment possible per cada joc i situació.
  - Crear estratègies que triïn el millor resultat d'entre un conjunt de moviments
  - Agrupar estratègies per definir perfils de jugadors.

2. Planificació i Eines de desenvolupament

En aquest apartat es veuran les primeres decisions que s'han pres per començar el projecte, així com les eines usades. Un cop presentat l'escenari inicial, s'expliquen les diferents fases per que passa el projecte.

2.1 Decisions inicials

Segons els objectius, s'ha d'implementar un programa que permeti jugar a un usuari al Sator, l'Arepo i el Tenet contra un altre usuari o contra si mateix. Per aconseguir això, es decideix considerar cada joc com un mòdul independent i usar un tauler i unes peces com un element bàsic i comú als tres jocs. Com que tenim mòduls independents, ens cal accedir-hi d'alguna manera, així que considerarem el programa principal un simple gestor de menús, on l'usuari escollirà a què vol jugar i com vol fer-ho.

Aquest plantejament està concebut per ampliar el nombre de jocs que usin el mateix tauler i peces a voluntat, convertint el programa en un recull de jocs i oferint així més possibilitats de joc.

2.2 APIs i llibreries

El projecte està enfocat a crear des de zero el projecte, així doncs no utilitza cap mena de llibreria externa per facilitar els càlculs. Tot i això sí que s'usen llibreries de caire general per gestionar la sortida del programa per pantalla i la interacció amb l'usuari.

Una de les ampliacions del programa és la possibilitat de jugar en línia contra altres jugadors. La connexió dels dos programes en línia es faria mitjançant una API que permet l'establiment d'un canal i la fàcil transmissió de dades entre les màquines. Aquesta API està adjuntada en el codi del projecte.

2.3 Eines de desenvolupament

Les eines que s'han fet servir per crear i provar el codi font programa han estat dues: l'Xcode i la Terminal del sistema.

2.3.1 Xcode 2.0

L'Xcode 2.0 és el programa de desenvolupament d'aplicacions d'Apple per als seus sistemes operatius Mac OS. Aquesta eina disposa tant d'un compilador, com manuals de referència per als diferents llenguatges que el programa suporti. Aquest programa també disposa d'una terminal pròpia per provar les aplicacions de línia de comandes.

2.3.2 Terminal

Tot i que l'Xcode 2.0 disposa d'una terminal, aquesta no és una del sistema sinó que és un complement del programa. Tot i que la majoria de crides a sistema estiguin implementades a dins de l'Xcode 2.0, n'hi ha d'altres que no ho estan.

Tot i que el projecte només disposa d'una crida al sistema, aquesta no es pot realitzar des de la terminal de l'Xcode 2.0 i per això s'usa directament la terminal del sistema.

2.4 Llenguatge C++

S'ha escollit el llenguatge C++ per dos motius: es tracta d'un llenguatge orientat a objectes i permet l'herència.

La necessitat d'un llenguatge orientat a objectes s'explica perquè tot joc de tauler guarda la informació de la posició de les peces, i amb aquesta informació es pot anar avançant. El plantejament és tractar el conjunt de peces i la seva posició com a objectes i el moviment d'aquestes peces pel tauler com els procediments que composarien una classe.

Segons l'apartat 2.1, considerem un tauler únic i els jocs com a mòduls independents. Seguint aquest plantejament, direm que els jocs hereten l'estructura del tauler i hi afegeixen les seves normes per desenvolupar el joc i arribar a un final.

2.5 Interfície gràfica

Crear una interfície gràfica és un objectiu indispensable, ja que tot programa s'ha d'executar i comprovar que els resultats obtinguts son els mateixos que s'esperen. Pels primers objectius es va usar la sortida de la terminal per observar el progrés del programa.

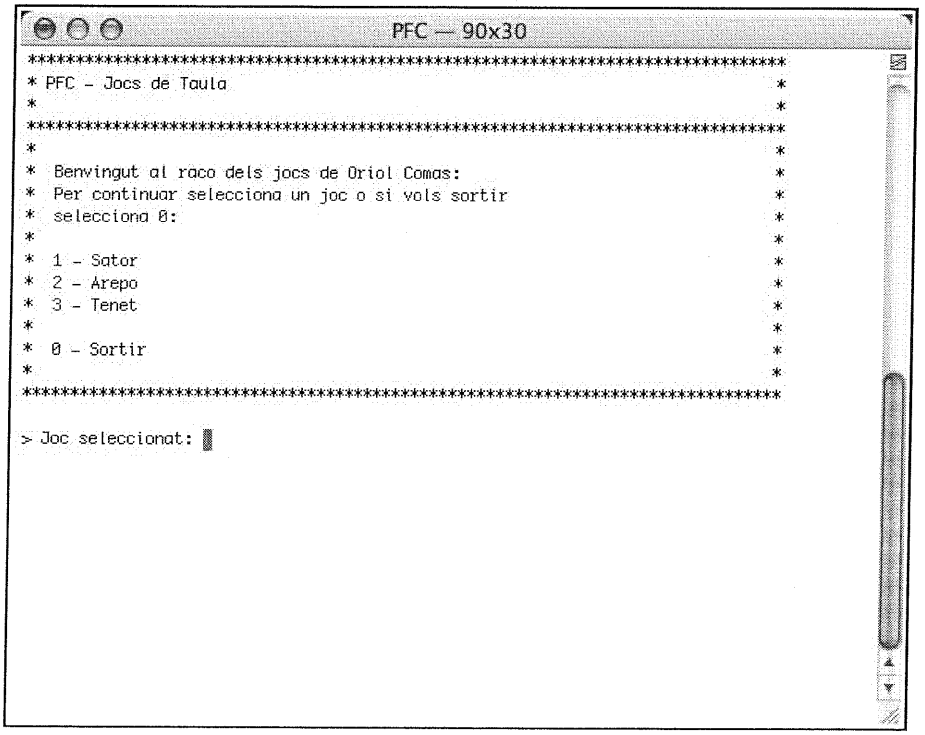

Fig. 5 Exemple de la interfície ASCII

Un cop es van assolir i els 3 jocs es van poder jugar com indiquen les seves especificacions, es va usar una interfície mitjançant ASCII i preparada per una terminal de 80x25. Aquesta interfície disposa de menús que permeten executar qualsevol dels jocs com es mostra a la Fig. 5.

2.6 Fase del projecte: especificació de l'aplicació

En aquests apartat enumerarem les característiques que ha de tenir el nostre projecte. Diferenciarem el programa principal, el tauler i les peces, cadascun dels jocs i com es mostrarà per pantalla. Tot això es detallarà en el següent apartat (cap. 3).

2.7 Fase del projecte: disseny

La fase de disseny és una de les més importants del projecte, ja que ha de complir amb l'especificació del projecte juntament amb la decisió inicial. Això implicarà una organització de classes molt clara i una homogeneïtat pel que fa als mòduls dels jocs. La fase de disseny s'explica detalladament en l'apartat 4.

2.8 Fase del projecte: implementació

Les solucions d'implementació, com son el control dels moviments, la interacció amb l'usuari o els bucles principals de joc s'exposen a l'apartat 5.

2.9 Fase del projecte: IA

Les fases de la creació de les IA, les seves parts més rellevants i el seu funcionament intern es detallen en l'apartat 6 i 7.

3. Especificació de l'aplicació

En l'especificació s'exposen les funcionalitats que ha de complir cadascuna de les parts del projecte.

3.1 Programa principal

El programa principal ha de guiar a l'usuari pels menús. La navegació ha de ser fàcil d'usar i les opcions clares. Ha de permetre anar endavant i enrere pels menús fins a arribar a l'execució d'un joc o bé sortir del programa. En el cas que els jocs tinguin variants, ha de permetre seleccionar la variant just després del menú i l'execució del joc.

El programa principal ha d'estar organitzat per permetre la posterior addició de nous jocs si es donés el cas. També ha de permetre la supressió d'aquests jocs.

3.2 Tauler i les peces

Les peces son l'element principal de tots els jocs. Han de tenir tots els elements que necessiti cada joc per a permetre el seu normal desenvolupament. Les peces sempre han de contenir la informació del jugador a qui pertanyen. També ha de ser fàcilment ampliable si es dóna el cas que s'afegeix un joc.

El tauler ha de ser únic pels tres jocs. Ha de ser versàtil per permetre el seu ús en una gran quantitat de jocs possibles. El tauler ha de contenir la posició de les peces en cara moment.

3.3 Els jocs

Els jocs han de ser modulars, així afegir-ne o treure'n un ha de ser una operació senzilla. Han de compartir una estructura i tenir les mateixes crides principals.

3.3.1 Sator

El Sator té tres modalitats: un jugador contra la màquina, la màquina contra la màquina i la versió de màquina contra màquina però per executar més d'una partida.

El Sator ha de permetre veure les peces amb un punt vermell o amb un punt blanc quan pertanyen a un jugador, però no facilitar aquesta informació al contrincant, ja sigui màquina o usuari. També ha de guardar la informació de les peces del contrari quan les captura.

3.3.2 Arepo

L'Arepo té dues modalitats: la modalitat de 2 jugadors contra 2 jugadors i el tots contra tots. Tant en una modalitat com en l'altra les peces no poden atacar les peces que es troben a les caselles més externes del tauler. El moviment de les peces és el mateix que el de les dames.

3.3.3 Tenet

El Tenet funciona com uns escacs. El Tenet depèn de les 15 cartes que se li assignen inicialment i ha de poder gestionar-les, sigui eliminant una carta, com totes les cartes si l'adversari captura una peça del jugador.

3.4 IA

Les IA s'executen sobre un jugador i al final del càlcul s'executa un moviment. En les IA es poden influir i fins i tot forçar un comportament o bé deixar que la IA trii quin és el millor moviment a realitzar.

La IA ha de ser fàcil de manejar i ràpida a l'hora de calcular el seu moviment, ja que una gran espera pot impacientar a un jugador humà.

4. Disseny

En aquest apartat veurem com es resolen les especificacions i l'organització interna del projecte. Degut a que les classes són molt extenses només es veuran per sobre.

4.1 Arquitectura de l'aplicació

L'aplicació ha estat creada de tal manera que cadascuna de les seves parts pugui funcionar per separat. Així cada joc es pot jugar de manera independent separant-ne el codi. Per ajuntar els tres jocs, es fa servir una classe que s'encarrega de gestionar els bucles de joc i la impressió per pantalla. Aquesta classe realitza totes les crides a les classes dels jocs, i a més a més, crea i modifica els taulers a plaer.

Quan s'executa l'aplicació s'entra al programa principal, d'aquí a través dels menús s'arriba a cadascuna de les variants dels jocs i es poden executar dins d'aquesta classe intermèdia. Quan el joc acaba es retorna al programa principal després d'haver mostrat per pantalla el guanyador del joc.

4.2 Diagrama de disseny

En la següent figura es mostra l'organització de les classes més bàsiques del programa. Aquest diagrama de disseny conté les classes tauler i peça, i els jocs. Aquí s'hauria d'afegir la classe escena i el programa principal.

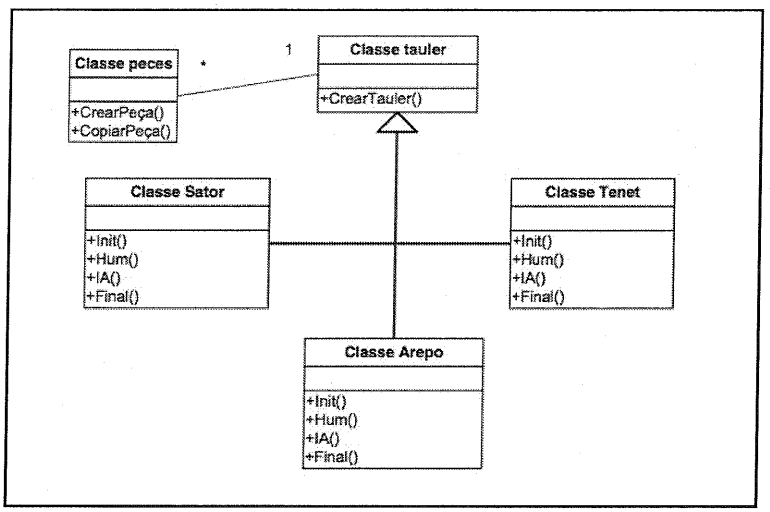

Fig. 6 Diagrama de disseny

4.3 L'objecte aplicació

L'aplicació funciona sobre una terminal del sistema de 90x30 (tot i estar preparada per una terminal de 80x25). S'interactua amb l'usuari mitjançant el teclat alfanumèric i sempre s'ha de confirmar amb el salt de linea. L'aplicació en cada moment demana la informació que necessita i en cas que l'usuari no introdueixi la informació que se li demana o en el cas que es tracti d'un joc, una moviment que no està permès, l'aplicació torna a demanar aquesta informació.

Per la gestió de textos per pantalla juntament amb la interfície ASCII, es treballa molt amb la llibreria string i el programa està preparat per ometre tots els espais intermedis a fi d'evitar errors d'entrada de dades.

4.3.1 La classe escena

La classe escena té tres classes funcions marcades: crear les pantalles dels menús i recollir la informació introduïda per l'usuari, executar els bucles de joc, mostrar les pantalles intermèdies del joc i quan aquest acaba mostra la pantalla final.

Les pantalles dels menús es divideixen en dues parts: les primeres són les que usa el programa principal i les segones són les que usen els bucles de joc. Tenen la mateixa funcionalitat, recollir la informació que introdueix l'usuari i la retornen al programa principal o al bucle de joc. Tot i fer exactament el mateix, les primeres serveixen per guiar a través del programa i les altres serveixen per inicialitzar els paràmetres dels jocs.

Els bucles de joc són el que realment s'utilitza per jugar. Comença inicialitzant els paràmetres del joc mitjançant les pantalles de menú. Quan té tota la informació usa els paràmetres per ajustar el tauler (si els paràmetres influeixen en la inicialització normal del joc es modifica el tauler segons els paràmetres). Un cop està la partida preparada dóna pas a la primera jugada segons les normes del joc i tot seguit comença a donar pas per torns als jugadors. A cada jugada utilitza les pantalles intermèdies per mostrar el tauler de joc, les peces i informació addicional del joc. A cada torn mira si s'ha arribat al final del joc, si és així usa la pantalla final del joc per mostrar el guanyador de la partida.

4.3.2 El programa principal

El programa principal s'encarrega de guiar l'usuari a través de dos nivells de menús fins al joc o la variant d'aquest que es vulgui executar. Aquests menús els obté de la classe escena (pantalles de menú) i indiquen al programa principal quin és el següent menú a mostrar. El comportament dels menús es força lineal ja que a cada pantalla del menú sempre hi apareix la mateixa informació i sempre et porta al mateix següent menú.

El primer menú és la pantalla de benvinguda. En la primera pantalla ens dóna la opció de triar quin dels tres jocs o sortir. Si escollim sortir ens mostra una pantalla final que ens indica d'on procedeixen els jocs i dona les gràcies per haver fet servir l'aplicació. Si per contra escollim un joc, el programa principal ens mostrara un altre menú on apareixen les variant del joc que havia escollit l'usuari. El programa principal sempre respon a l'estructura de dos menús, un pel joc i l'altre per la variant. Un cop l'usuari ha triat la variant l'execució passa a ser a la classe escena i el seu bucle de joc corresponent. Quan l'execució del joc arriba a la seva fi, el programa torna a l'ultim menú de la variant que havia seleccionat l'usuari.

4.3.3 La classe peces

La classe peces compleix dues premisses: ser la classe més simple possible i que un objecte de la classe peces sempre ens indicarà a quin jugador pertany.

La necessitat de ser una classe senzilla pren més importància quan parlem de les IA. Tot i no haver gaires peces en joc, la senzillesa de la classe permet copiar-la en cost lineal O(n), cosa que redueix el cost computacional i cost espacial del càlcul de les IA.

La segona premissa que compleix la classe peça es que cadascun dels seus objectes pertany a un jugador. Això pot semblar força obvi però no ho és tant: ser o no d'un jugador condiciona l'atac d'una peça i també la seva direcció de moviment com passa en el Sator o l'Arepo. En el Tenet condiciona també l'atac però en comptes de la direcció, condiciona la seva possibilitat de moviment ja que una peça està vinculada a un conjunt de cartes que limiten els seus possibles moviments.

La classe peces passa només per dues etapes, la primera és quan s'inicialitza el joc també s'inicialitzen els valors de les peces segons al joc a que es vagi a jugar. La segona etapa és la sobre escriptura dels valors de la peça avança o en captura una altra.

4.3.4 La classe tauler

La classe tauler requereix a l'igual que la classe peces d'una estructura molt senzilla. Tot i que la classe tauler es presenta com a senzilla, es tracta d'una matriu quadrada de caselles on col·locar-hi les fitxes. La cerca i la manipulació de peces dins el tauler s'incrementa fins a O(n^2) si no es disposa de la informació d'on es troba la peça.

La mida del tauler és bàsica pel seu bon funcionament en els 3 jocs: tant el Sator com el Tenet usen taulers de 5x5 però l'Arepo usa taulers de 7x7. A fi de minimitzar el cost espacial de la classe s'ha decidit que el tauler tingui una mida màxima de 7x7 i cada joc aprofiti les caselles que necessiti. En el cas de voler ampliar el tauler per incloure jocs com ara les dames o els escacs només caldria canviar-ne els paràmetres inicials.

4.3.5 Els jocs

Cadascun dels jocs és una classe, així tenim la classe Sator, la classe Arepo i la classe Tenet. Cadascuna d'aquestes classes disposa de les seves característiques pròpies marcades per les normes del joc.

4.3.5.1 La classe Sator

La classe Sator hereta de la classe tauler. Així doncs, disposa d'una matriu de caselles 7x7 per guardar-hi les peces i desenvolupar-hi el joc. Però no només disposa del tauler, sinó que fa servir altres variables per mantenir una informació actualitzada a cada torn del joc: el torn, les peces que coneix un jugador de l'altre, la mida del tauler que farà servir el joc, tipus de jugador i quin jugador comença.

El torn es va incrementant a cada moviment que es fa, però no es considera torn si el jugador no pot moure cap peça. Si es donés el cas que un jugador quedés bloquejat o sense peces, el torn seguiria augmentant amb els moviments del seu oponent fins que s'arribés al final del joc.

Les peces que es coneixen de l'adversari és una informació molt útil, ja que aquest joc es basa en la informació oculta per les peces (punt blanc o punt vermell). Conèixer aquesta informació pot donar la partida o condicionar un moviment o un altre. Aquesta informació només s'actualitza quan es captura una peça i segons el tipus de captura s'actualitzen les dades d'un jugador o dels dos.

En el primer cas, si una peça qualsevol captura una peça amb punt blanc es resta una peça blanca del que sabia el jugador que captura del que ha perdut la peça. L'altre cas es dóna si es captura una peça amb punt vermell, si passa això, s'actualitza la quantitat de peces vermelles que coneixia l'atacant i es posa una peça en dubte del jugador que ha perdut la peça a l'atacar.

Les peces en dubte apareixen com a incertesa i no actualitzen les peces amb punt blanc o vermell dins la informació de les peces del contrincant ja que segons la normativa no es permet mostrar els punts de les peces que s'han tret del joc per capturar una peça amb punt vermell. Per tant, si un jugador sap que té tres peces amb punt blanc i dos amb punt vermell però té un dubte vol dir que el seu adversari ha perdut una peça al capturar una peça pròpia amb punt vermell i en realitat no té cinc peces sinó quatre, però no sap si té dues de cada o tres amb punt blanc i una amb punt vermell.

La mida del tauler és important ja que la distància que ha de recórrer una peça per arribar a la meta en el Sator és cinc, i no set com és la mida del tauler genèric. Aquests paràmetres del joc permeten delimitar el joc dins la primera àrea de 5x5 del tauler, i poder jugar amb normalitat al Sator.

El tipus de jugador és una dada que fa servir els bucle de joc per demanar que executi el següent moviment l'usuari o bé la màquina. Per exemple, una de les modalitats de joc del Sator és la de màquina contra màquina, en aquest cas el tipus de jugador marcaria màquina pel primer jugador i pel segon, i cada torn demanaria que mogués la màquina el jugador corresponent (el primer o el segon segons correspongui).

El que ens indica si un jugador comença la partida és que controli les peces negres. Aquesta dada no només dóna només qui comença la partida, sinó que pel bucle de joc ens dóna quin jugador comença cada vegada en el bucle. Així si el primer jugador té negres, cada bucle de joc s'executarà: demanar moviment jugador 1, demanar moviment jugador 2, fins al final de la partida.

4.3.5.2 La classe Arepo

La classe Arepo hereta , com ho feia la classe Sator, de la classe tauler. Les cinc caselles centrals de cada costat del tauler es fan servir de punt de partida de les peces d'un color i les 4 caselles restants es consideren no útils, i així l'Arepo usa una matriu de 7x7.

Al ser un joc més senzill, no requereix guardar gaires dades pel seu desenvolupament, tot i així en guarda dues: el torn i el tipus de jugador. Tant el torn i el tipus de jugador segueixen el mateix comportament que en la classe Sator.

4.3.5.3 La classe Tenet

La classe Tenet hereta de tauler com el Sator, i com el Sator usa un tauler de 5x5. La classe utilitza algunes dades per gestionar el joc correctament: quin jugador té les negres, el torn, el tipus de jugadors i la mida del tauler. Aquestes dades segueixen el mateix comportament que les exposades en el punt 4.3.5.1.

La classe tenet té un número límit de torns. Aquest límit ve donat pel nombre de cartes donat a cada jugador. Aquestes cartes s'usen per permetre un moviments i es retiren del joc quan una peça mor o quan es realitza un moviment. Però aquestes cartes també son part de la informació de la partida, ja que si sabem els moviments que pot fer el nostre rival ens condiciona alguna de les nostres jugades. L'ús de les cartes requereix la creació d'una nova classe.

4.3.5.3.1 La classe cartes

La classe cartes apareix per gestionar els possibles moviments d'un jugador. Cada objecte de la classe cartes només pertany a un jugador, per tant, la classe Tenet requereix l'us de dos objectes cartes.

Les principals funcions de la classe cartes són: informar de quins moviments estan disponibles i quins no, actualitzar les dades cada cop que es realitza un moviment i eliminar totes les cartes d'una peça que ha estat capturada. Una funcionalitat que apareix després en la classe cartes és la de mostrar tots els possibles moviments d'una peça per a cada carta (es veurà en la secció de la IA).

5. Implementació

En aquest capítol es veurà les parts més importants de la implementació del projecte. S'explicaran les parts relacionades amb el joc Sator, ja que la resta son adaptacions d'aquesta.

5.1 L'estat

Anomenem estat a la estructura de dades que fa servir el Sator per guardar els possibles moviments d'un jugador. Si un estat està vuit vol dir que el jugador no té jugades disponibles, això es pot donar en 2 casos possibles: el jugador no té cap peça, o bé, el jugador té peces però estan bloquejades.

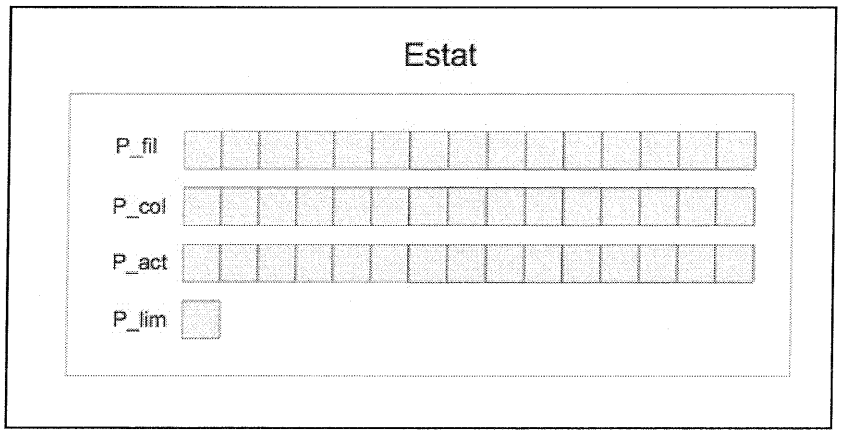

Fig. 7 Estructura de l'estat

L'estat es composa de 3 vectors d'integers: p_fil, p_col i p_act, i un integer anomenat p_lim que ens indica el número de moviments disponibles guardats a l'estat. Els tres vectors tenen un màxim de 15 posicions. Aquestes 15 posicions son el màxim teòric, ja que cada jugador disposa de 5 peces inicialment, cada peça pot realitzar 3 moviments: avançar, atacar a la dreta o atacar a l'esquerra.

5 peces * 3 mov / peça = 15 moviments

Aquest màxim no s'hi arriba mai ja que perquè totes les peces puguin matar a dreta i esquerra, implica que hi hagi 2 peces per cada peça en disposició de matar. Si disposéssim un taulell prou ample i col·loquéssim totes les peces en posició òptima veuríem que necessitaríem 6 peces en el equip contrari per poder arribar als 15 moviments.

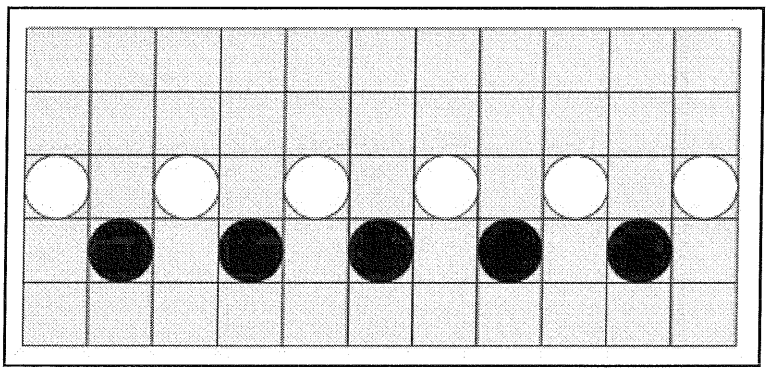

Fig. 8 Tauler teòric per 15 moviments

Cada element que es guarda a l'estat, es guarda com si fos una llista, el primer element guarda un integer a la primera posició de p_fil, a la primera de p_col i la primera de p_act, tanmateix augmenta en una unitat el p_lim. Cada element que s'afegeix es posa a continuació de l'anterior.

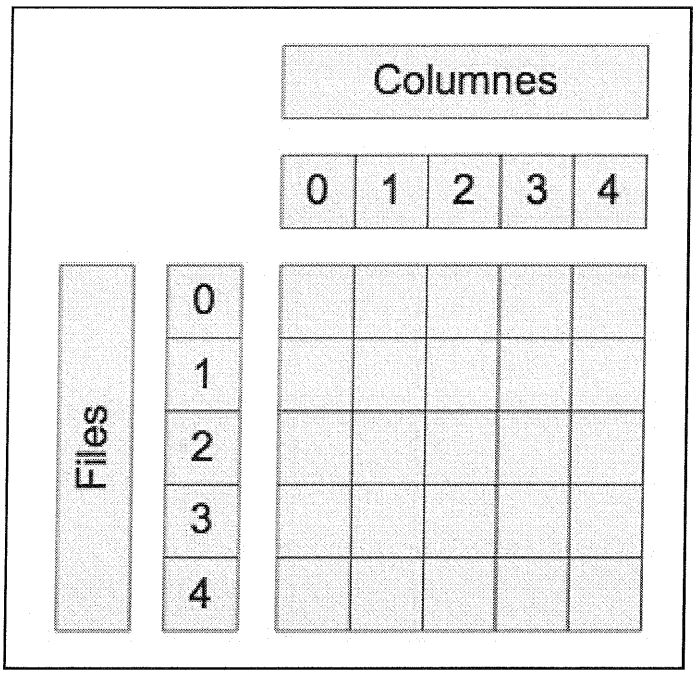

Fig. 9 Numeració del tauler

La funció encarregada d'omplir un estat s'anomena "extreu_estat". Cada vegada que s'omple un estat, només serveix per aquell moment del joc, així cada vegada que es realitza un moviment s'ha de renovar l'estat de tots dos jugadors.

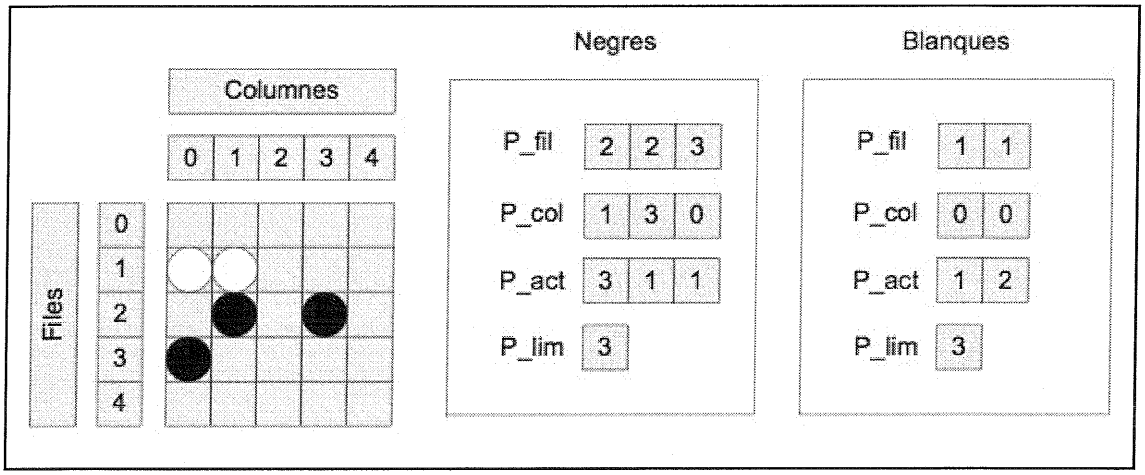

Fig. 10 Exemple d'estat

Aquesta funció funciona de manera iterativa, buscant peces al taulell des de la posició (0,0) fins a la posició (número de files -1, número de columnes -1), recorrent així tota la matriu. Per a cada cel·la del taulell mira si hi ha la peça que es del jugador que estem extraient l'estat, i si es així mirem si pot: atacar a l'esquerra, moure endavant i atacar a la dreta.

Si pot fer alguna de les accions, per a cada moviment guardem la fila de la peça a la posició p_lim del vector p_fil (p_fil[p_lim]), la columna de la peça a la posició p_lim del vector p_col (p_col[p_lim]), i a la posició p_lim del vector p_act (p_act[p_lim]) guardem: **1** si pot moure endavant, **2** si pot atacar a la dreta i **3** si pot atacar a l'esquerra. Quan hem guardat la posició augmentem en un p_lim.

5.2 El bucle de joc

Els bucles de joc es troben emmagatzemats a la classe escena. Aquests bucles són únics per a cada variant del joc. Els bucles segueixen l'algorisme que es mostra a la Fig 11.

El Sator està pensat per poder-se executar en 3 modes de joc: 1 jugador contra la CPU, la CPU contra la CPU (amb monitorització) i la CPU contra la CPU (sense monitorització). Així doncs, per cada torn del jugador es llença la funció hum, que gestionarà tots els passos necessaris perquè el jugador acabi movent la seva peça o passarà el torn si veu que no pot moure peça. En el cas que sigui la CPU qui hagi de moure, també té la seva pròpia funció: la funció IA.

```
procediment bucle_de_joc ()
    Inicialització (mostrar_menú_inicial ( ) )
    mentre ! final() fes
        si comença("jugador 1") llavors
            // Ordre de jugada: jugador 1 -> jugador 2
            si es_humà("jugador 1") llavors
                hum("jugador 1")
            sinó
                IA("jugador 1")
            fi si
            // Torn del jugador 2
            si es_humà("jugador 2") llavors
                hum("jugador 2")
            sinó
                // Ordre de jugada: jugador 2 -> jugador 1
                si es_humà("jugador 2") llavors
                    hum("jugador 2")
                sinó
                    IA("jugador 2")
                fi si
                // Torn del jugador 1
                si es_humà("jugador 1") llavors
                    hum("jugador 1")
                sinó
                    IA("jugador 1")
                fi si
            fi si
        fi mentre
            mostrar_menú_final()
    fi procediment
```

Fig. 11 Algorisme dels bucles de joc

Aquests algorismes funcionen sempre de la mateixa manera: el bucle de joc executa les pantalles de menús. Com a resultat obté les els paràmetres inicials del joc.

5.2.1 Funcions Init

La funció init recull les dades dels menús i les passa a la inicialització del tauler. En cadascuna de les tres variants la recollida de dades es diferent.

En la primera variant, la de jugador contra màquina només es recull quin jugador duu negres. En la segona variant, la de màquina contra màquina es recullen les dades que farà servir la IA, com són el patró de comportament. En la tercera variant a part de recollir els comportaments, recull el nombre d'execucions que ha de realitzar el programa.

5.2.2 Funció hum

La funció hum llegeix, quan es el torn del jugador, la peça que s'ha de moure. Si la posició indicada no es la correcta, torna a demanar una nova posició. Si es donés el cas que la peça té més d'un moviment possible, demanaria introduir: 1 per moure endavant, 2 per atacar a la dreta i 3 per atacar a l'esquerra. Un cop el moviment es vàlid, s'executa sobre el taulell el moviment de la peça i s'incrementa el torn actual.

5.2.3 Funció IA

La funció IA s'encarrega de moure una peça de la màquina. Aquest moviment es calcula segons dos processos que s'executen un darrera l'altre, el primer és l'expansió de moviments i el segon és l'aplicació de l'estratègia.

L'expansió de moviments és detalla al capítol 6 i l'aplicació d'estratègies en el capítol 7.

5.2.4 Funcions final

Les funcions final tenen dues funcionalitats: detectar quan el joc ha arribat a una situació de final i indicar en cas d'arribar al final, quin jugador és el guanyador.

Aquesta sortida la té codificada de la següent manera: retorna 0 si el joc encara no ha acabat, retorna 1 si el jugador 1 ha fet arribar una peça al final, i si retorna 2 és el jugador 2 que ha arribat una peça al final. En el cas que cap peça arribi al final i ja no es pugui moure més, retornara el jugador que hagi avançat més les peces, però si es dóna el cas que els dos jugadors tenen el mateix nombre de peces avançades retorna 3.

6. IA: Expansió de moviments

En aquest capítol s'explica el funcionament de la primera fase de la IA: l'expansió de moviments.

6.1 Expansió de moviments

El primer que fa la funció IA és extreure l'estat del jugador que ha de moure peça. Acte seguit es generen còpies idèntiques del taulell actual, tantes com elements te l'estat que ha extret la IA. Un cop estan totes les còpies, es realitza de manera ordenada un moviment de l'estat per cada còpia del taulell.

A continuació es crea un vector anomenat alfes amb el mateix nombre de posicions que les còpies dels taulells. Un cop està tot preparat, s'executa la funció alfabeta a cadascun dels taulells, i el resultat de la funció es guarda en la posició corresponent del vector d'alfes.

6.2 La funció Alfabeta

La funció alfabeta s'utilitza en molts jocs de dos jugadors organitzats per torns. L'alfabeta s'encarrega de crear tots els moviments possibles, alternant cada jugador, per tal de cobrir tots els possibles moviments i veure quina es la millor jugada.

El primer que fa la funció alfabeta es comprovar si ja ha mirat totes les jugades per endavant que havia de mirar o si ja s'ha arribat a una situació de final (un jugador ha arribat una peça blanca a l'inici de del jugador contrari). Quan arriba a una situació de final, s'executa la funció eval que ens dóna un valor del taulell actual i la funció alfabeta retorna aquest valor.

Si el cas que tracta l'alfabeta es el cas que màxim (cas inicial o intermedi però que tracta el jugador que ens interessa) mirem si la crida d'alfabeta llençada des de l'actual dona un valor mes gran que l'alfa actual i si es així guarda l'alfa actual. Si es dóna el cas que l'alfa actual és més gran que la beta no seguim explorant totes les possibilitats.

Si ens trobem en el cas que és un cas mínim (cas del contrari jugador contrari al que ens interessa) guardem el valor del mínim entre la beta i la crida de l'alfabeta que llancem a cada moviment possible. Tal i com passa en el cas màxim, l'alfabeta pot deixar de les crides a alfabeta si la beta és mes gran o igual que l'alfa.

El nombre de jugades que s'avança l'alfabeta depèn d'un paràmetre de la funció que es va reduint a cada crida d'alfabeta. Cal recordar que cada crida a alfabeta crea una expansió de moviments del jugador que esta tractant en aquella crida de l'alfabeta, així doncs, cada jugada que s'avança genera un arbre de taulells i això multiplica els càlculs a fer, alentint la resolució de la funció alfabeta.

```
Procediment alfabeta ( J )
    a := - ∞
    b := ∞
    si J es node terminal llavors
        V(J) := eval(J)
    sinó
        siguin J1, J2, .. Jn successors de J
        k := 1
        si J es max llavors
            a := max(a, V(J;a,b))
            si a ≥ b llavors
                retorna b
            sinó
                continua
            fi si
            si k = n llavors
                retorna a
            sinó
                k := k + 1
                continua
            fi si
        sino
            b := min(b,V(J;a,b))
            si b ≤ a llavors
                retorna a
            sinó
                continua
            fi si
            si k = n llavors
                retorna b
            sinó
                k := k + 1
                continua
            fi si
        fi si
    fi si
fi procediment
```

Fig. 12 Algorisme Alfabeta

L'alfabeta "juga" amb els valors que va obtenint d'eval i els va distribuint entre alfa i beta segons l'algoritme de la funció. Quan retorna l'últim valor la funció alfabeta és el valor que tindrà aquell moviment.

Quan es llença l'alfabeta per cada taulell en l'expansió de moviments de la IA, el que se n'obté és un vector de resultats, i per tant, un valor per cada possible moviment. Tenint un valor assignat a cada moviment és possible traçar una estratègia.

6.3 La funció eval

La funció eval de l'alfabeta només dóna com a resultat 1, 0 o -1. Aquests resultats signifiquen que l'últim moviment ha provocat la victòria del jugador que interessa, que l'última jugada no genera un final de partida o que l'última jugada ha provocat la victòria de l'adversari respectivament.

En aquest projecte no s'usa la funció eval pròpia de l'alfabeta, sinó que s'usa una adaptació per ajudar el càlcul de les estratègies que s'aplicaran a continuació. Aquesta adaptació és diferent a cada joc, ja que les situacions son diferents, com també ho són els objectius.

6.3.1 Sator

Els principals objectius del Sator són avançar peces blanques per fer-les arribar a la última línia del tauler, o bé, evitar que les peces blanques de l'adversari arribin a la nostre línia inicial.

	Peça blanca	Peça amb un punt vermell
Peça sota amenaça	-10	10
Peça sota doble amenaça	-20	20
Peça bloquejada	-40	10
Peça bloquejada, sota amenaça	-50	20
Peça bloquejada, sota doble amenaça	-60	30
Punts / casella avançada	20	5
Peça arribada a l'ultima casella	0	20
Peça amb via lliure fins a l'ultima casella	100	0

Fig. 12 Taula de l'eval del Sator

Tot i la importància de les peces amb punt blanc, les peces amb un punt vermell són bàsiques pel desenvolupament del joc: permeten parar esquers, atacar sense por a perdre una peça important, i en el cas que les capturin, treuen una peça de l'adversari del joc.

Podríem pensar així que les peces vermelles tenen molta importància dins del joc, i així és, però no tanta com les peces amb punt blanc com reflexa la Fig. 12.

En la Fig. 12 veiem que apareixen les situacions de risc potencial o més desfavorables que pot tenir una peça: estar amenaçada per una peça rival o bloquejada. Aquests fets afecten de manera diferent les peces amb punt blanc, que les peces amb punt vermell. Si aquestes situacions desfavorables afecten a peces amb punt blanc, el valor de eval decreix, al ser perjudicial pel jugador. Per contra, si afecten a peces amb el punt vermell, existeix la possibilitat d'estar bloquejant o essent víctima de la captura d'una peça amb punt blanc de l'adversari, fer que beneficia al jugador.

No només es valoren les situacions desfavorables, sinó també les favorables, tot i que aquests moviments favorables només prenen valor si la peça a que afecten té un punt blanc. Aquestes situacions són: cada casella avançada per una peça (o distància a l'origen) i la possibilitat de guanyar el joc en el següent torn.

6.3.2 Arepo

L'Arepo, a diferència del Sator, és més simple a l'hora de jugar ja que no té informació oculta. Al comptar amb aquesta simplificació, l'Arepo obté més informació valorant les accions immediates de realitzar un moviment.

El càlcul doncs, es realitza segons la premissa de mirar la situació actual de cada peça, i la situació que ocuparia quan es realitzés un moviment sobre la peça.

	Valor
Peça sota amenaça	-3
Peça amb defensa	8
Peça bloquejada	-1
Peça creant doblet	-4

Existència de la peça	3
Punts / casella avançada	2

Fig. 13 Taula de l'eval de l'Arepo (situació actual)

En l'Arepo existeix igualtat entre les peces, això fa que es valorin per la seva situació i relació amb les altres peces, més que si és d'una classe o d'una altra.

Els casos favorables que comprova l'eval són: l'existència de la peça, la distància a l'origen i que una peça estigui defensada. Per contra, si una peça es troba en una situació desavantatjosa com seria estar sota amenaça o bloquejada, això restaria valor al resultat de la funció eval. També existeix un cas en el que una peça bloqueja una altra lateralment, impedint que avanci, però sense perdre la seva pròpia mobilitat cap a la meta, aquests cas també es considera desavantatjós ja que si es mou la peça endavant, queda exposada a l'ataca de la peça que bloquejava.

	Valor
Peça sota amenaça	-5
Peça bloquejada	-2
Peça creant doblet	-2
Peça creant bloqueig sobre l'enemic	2
Està prop de la meta	15
Peça defensada	7
Pot atacar	4

Fig. 13 Taula de l'eval de l'Arepo (situació futura)

L'altre punt que mira l'eval de l'Arepo és en quina situació es trobarà la peça realitzant cadascun dels moviments dels que disposa. Aquests fet vol dir que aquests segon càlcul no es realitzaria en el cas que les peces estiguessin bloquejades, o es podria donar el cas que hi haguessin més moviments que peces.

Com passava en l'eval de la situació actual, que una peça avanci i es trobi sota amenaça no és un bon moviment. Així també existeix una similitud entre les dues situacions: tenir una peça sota amenaça o bloquejada segueix considerant-se un cas desfavorable i restant punts al valor total. Però

també hi ha una diferència: crear un bloqueig lateral al moure implica que la peça estava sota amenaça en la seva situació actual, realitzant aquest moviment doncs, millorem el valor del jugador.

6.3.3 Tenet

El valor del tauler en el Tenet canvia respecte del Sator i l'Arepo. En el Tenet, la importància no recau sobre la posició actual i la futura, sinó amb la existència de peces sobre el tauler, ja que és aquest fet el que decideix la victòria o la derrota d'un jugador.

Els valors de les peces són els mateixos que es comproven al final dels 30 torns del joc o el fi de la partida degut a la impossibilitat de moure més peces de cada jugador.

Peça	Valor
Rei	1
Cavall	2
Alfil	3
Torre	4
Dama	5

Fig. 14 Taula de l'eval del Tenet

7. IA: Estratègies i perfils

En aquest capítol s'explica el funcionament de la segona part de les IA, la encarregada de valorar els resultats obtinguts de la primera part i executar un moviment. Només es mostraran les estratègies i perfils del Sator, ja que per la resta de jocs son adaptacions d'aquests.

7.1 Estratègia

Entenem estratègia com a subconjunt dels moviments possibles d'un estat que compleixen un cert nombre de condicions. Aquestes estratègies poden concatenar-se de tal manera que si el primer conjunt d'estratègies està buit, es pugui aplicar la següent estratègia.

Totes les estratègies poden concatenar-se i formar una cadena d'estratègies, que anomenarem perfil o perfil d'estratègies. Pel Sator hi ha un total de vuit estratègies que es descriuran en els següents punts: Random, Millor, Agressiva, Defensiva, Bloquejant, Corredora, Trampera i Protectora.

Per aplicar una estratègia, abans s'ha hagut de comprovar que es pugui aplicar, acte seguit s'ha de trobar el millor moviment dins els subconjunt de moviments que forma l'estratègia. D'això se n'encarrega una funció anomenada "escull_atac_múltiple".

7.1.1 La funció escull_atac_múltiple

La funció escull_atac_múltiple rep tres vectors: un vector d'enters i dos de booleans. Cadascun dels moviments que formen part de la estratègia envien un enter i dos booleans a aquesta funció, i segons aquests paràmetres seran comparats.

El primer que es mira és el vector d'enters, si només hi ha un màxim, es retorna el moviment amb enter màxim (o mínim segons sigui necessari). Si hagués més d'un màxim, miraria el primer vector de booleans i retornaria la posició, que d'entre els màxims, dels valors que tinguin com a cert (o fals segons convingui a l'estratègia). Però això només passarà si només existeix un valor (sigui cert, o fals) que coincideixi amb el valor que a l'estratègia li es favorable. Si es donés el cas que hi

hagués més d'un valor amb coincidència es miraria el 2n vector de booleans i es procediria igual que amb el primer vector d'aquests.

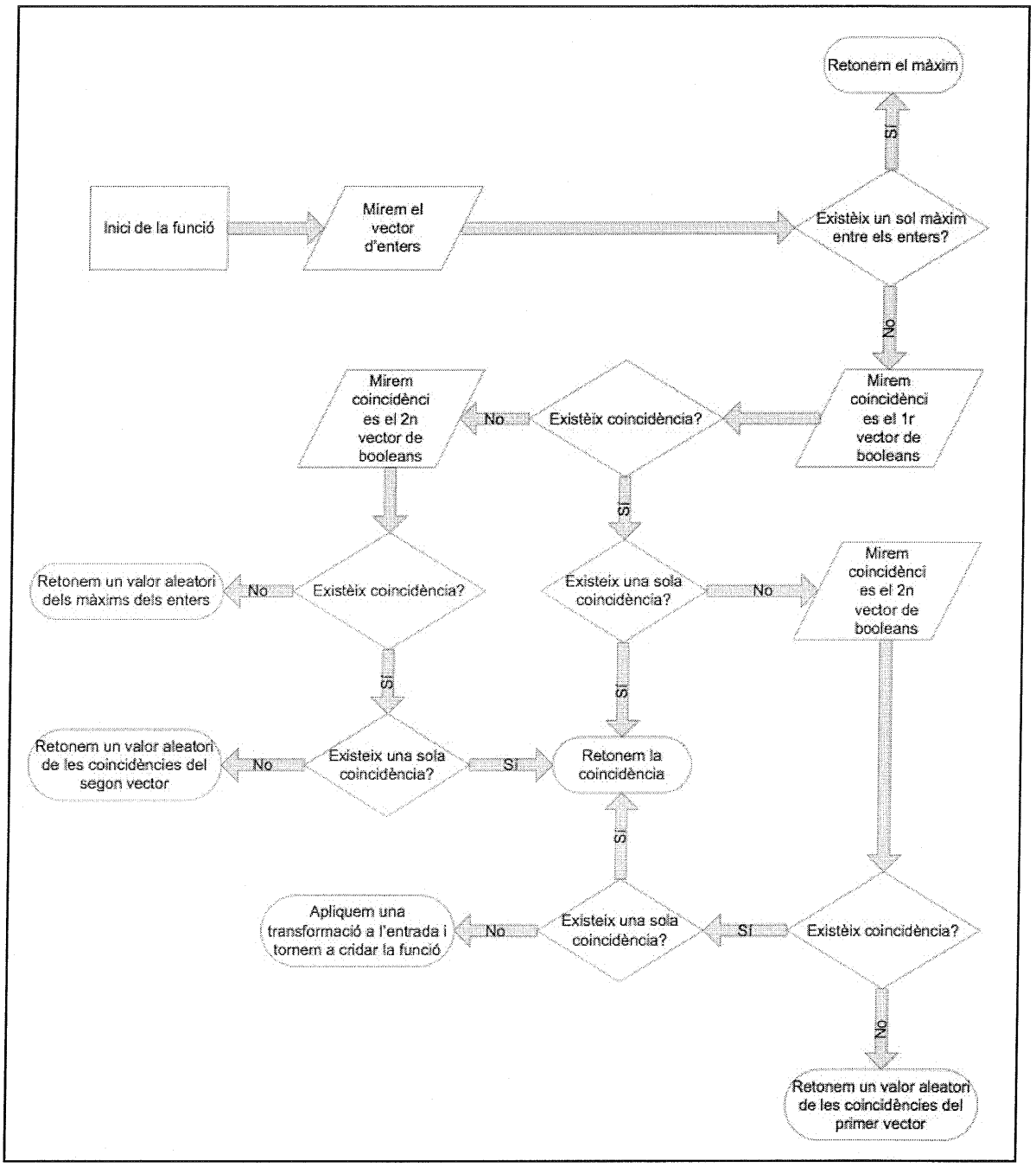

Fig. 15 Diagrama del flux de la funció escull_atac_múltiple

Si un cop passat tot això encara hi ha més d'un valor que coincideixi amb el valor necessari del segon vector de booleans seguiríem els següents passos: Si no coincidís cap dels valors dels vectors de booleans, triaríem un valor a l'atzar dels màxims del vector dels enters. Si només hi havia coincidències amb el valor en el primer vector de booleans es triaria a l'atzar entre els moviments

que coincideixen en el valor d'aquest primer vector. Si per contra, no hi hagués coincidència en el primer vector però si en el segon, es realitzaria la mateixa operació que en el primer vector de booleans, però al segon.

L'últim cas que tractaria seria el que hi hauria valors amb coincidència en el primer i segon vector de booleans, en aquest cas es tornaria a llençar la funció escull_atac_múltiple amb els mateixos valors en el vector d'enters, un valor cert els que coincideixin els valors del primer i segon vector en el primer vector de booleans i un valor fals a tots els valors del segon vector de booleans.

7.1.2 Tipus d'estratègies

La divisió de les estratègies es basa en les condicions que han de complir cadascuna. Algunes de les estratègies son complementàries, i d'altres comprenen tots els moviments, tot i així, la idea de separar-les pren sentit quan s'ordenen per crear un perfil.

7.1.2.1 Aleatòria

L'estratègia aleatòria (o Random) pren com a condició el ser un moviment disponible de l'estat. Així doncs parlem del conjunt de tots els moviments de l'estat. Per aplicar aquesta estratègia no tenim en compte els valors de les alfes calculades per la funció alfabeta, sinó que triem un dels moviments a l'atzar.

7.1.2.2 Millor

L'estratègia Millor pren com a condició tenir una alfa màxima. Si sabem que es disposa d'un o més d'un moviment sempre hi haurà un màxim disponible. Si només hi ha un màxim s'aplica el moviment associat a aquesta alfa, si n'existeix més d'un, s'aplica el primer que es troba.

7.1.2.3 Agressiva

L'estratègia Agressiva pren com a condició que el moviment sigui de caire agressiu (atacar a la dreta o atacar a l'esquerra). Tot i existir moviments en l'estat, si cap d'ells es d'atac aquesta estratègia no podria executar-se.

Aquesta estratègia sempre intentarà atacar preferiblement amb una peça vermella ja que amb una peça vermella no es pot guanyar directament el joc. Tot i que existeix una petita component aleatòria que, donat el cas que existís la possibilitat d'atacar amb una peça blanca, evités atacar amb la peça vermella i ataqués amb la blanca.

En el cas que es pogués atacar amb una peça vermella i la component aleatòria no fes atacar amb una peça blanca, es procediria de la següent manera: si només hi hagués un sol moviment ofensiu amb la peça vermella, atacaríem amb aquest. En el cas d'haver-n'hi més d'un procediríem a usar la funció escull_atac_múltiple amb els següents paràmetres:

	Paràmetre	Valor desitjat
Vector d'enters	Distància a l'origen	MÀXIM
Vector de booleans 1	Fa doblet amb una altre peça del mateix jugador a l'atacar?	FALS
Vector de booleans 2	Fa doblet amb una peça blanca del mateix jugador a l'atacar?	FALS

Fig. 16 Taula de peces amb punt vermell

En el cas que s'hagués d'atacar amb una peça blanca, es procediria de la següent manera: si només hi ha un moviment ofensiu, s'executaria aquest, si n'hi hagués més d'un s'usaria la funció escull_atac_múltiple amb els següents paràmetres:

	Paràmetre	Valor desitjat
Vector d'enters	Distància a l'origen	MÀXIM
Vector de booleans 1	Té alguna peça del mateix equip que també amenaci la peça objectiu?	CERT
Vector de booleans 2	Fa doblet amb una altre peça del mateix jugador a l'atacar?	FALS

Fig. 17 Taula de peces amb punt blanc

7.1.2.4 Defensiva

L'estratègia defensiva té com a condició realitzar un moviment d'avanç i que en realitzar-lo estiguin cobrint una altre peça del mateix jugador. Aquesta estratègia no funciona amb els moviment d'atac ja que es pot donar el cas que a l'atacar, la peça mati una peça enemiga amb un punt vermell, cosa que faria que no arribés a defensar una altre peça.

Es poden donar 3 casos: el cas que no hi hagués cap peça que pogués avançar per defensar-ne una altre. El cas que només hi hagués un moviment, aquest s'executaria directament, o bé, el cas que hi hagués més d'un moviment usaríem la la funció escull_atac_múltiple amb els següents paràmetres:

	Paràmetre	Valor desitjat
Vector d'enters	Nivell d'amenaça de que te la peça actualment.	MÍNIM
Vector de booleans 1	La peça té un punt vermell a sota?	CERT
Vector de booleans 2	La peça estarà sota amenaça un cop avanci?	FALS

Fig. 18 Taula de l'estratègia defensiva

7.1.2.5 Bloquejant

L'estratègia bloquejant té com a condició bloquejar una peça del jugador contrari amb un moviment d'avanç del jugador actual. Com passa amb l'estratègia defensiva, no es poden contemplar com a moviments de l'estratègia bloquejant els moviments d'atac, ja que podrien atacar una peça amb un punt vermell del jugador contrari i no bloquejarien cap peça, ja que segons les normes del Sator al matar una peça amb punt vermell la peça que l'ha atacat també es retira del joc.

Es poden donar 3 casos: el cas que no hi hagués cap peça que pogués avançar per bloquejar-ne una altre. El cas que només hi hagués un moviment, aquest s'executaria directament, o bé, el cas que hi hagués més d'un moviment usaríem la la funció escull_atac_múltiple amb els següents paràmetres:

	Paràmetre	Valor desitjat
Vector d'enters	Distància de la peça a l'origen	MÍNIM
Vector de booleans 1	La peça té un punt vermell a sota?	CERT
Vector de booleans 2	La peça està sota amenaça?	CERT

Fig. 19 Taula de l'estratègia bloquejant

7.1.2.6 Corredora

L'estratègia corredora engloba tots els moviments d'avanç. Aquesta estratègia consisteix en el fet d'intentar guanyar la partida quan abans millor, així doncs donarà preferència a les peces més avançades i les que tinguin pas fins a guanyar la partida, sempre posant per davant les peces blanques ja que les peces amb punt vermell no poden guanyar directament la partida.

Es poden donar 3 casos: el cas que no hi hagués cap peça que pogués avançar. El cas que només hi hagués un moviment, aquest s'executaria directament, o bé, el cas que hi hagués més d'un moviment usaríem la la funció escull_atac_múltiple amb els següents paràmetres:

	Paràmetre	Valor desitjat
Vector d'enters	Nivell d'amenaça de que te la peça actualment.	MÍNIM
Vector de booleans 1	La peça té un punt vermell a sota?	CERT
Vector de booleans 2	La peça estarà sota amenaça un cop avanci?	FALS

Fig. 20 Taula de l'estratègia corredora

7.1.2.7 Trampera

L'estratègia trampera recull tots els moviments d'avanç que un cop avançada la peça estigui defensada o tingui un punt vermell, i sota amenaça al mateix temps (en qualsevol dels 2 casos). La idea d'aquesta estratègia es provocar que l'altre jugador mati una peça ja defensada o una vermella i així provocar la retirada del joc d'aquella peça. Aquesta estratègia pot ser útil per desgastar les peces o bé per obrir pas a una peça perquè avanci i pugui arribar a la fila contrària, guanyant la partida.

Es poden donar 3 casos: el cas que no hi hagués cap peça que pogués avançar per trampejar la situació. El cas que només hi hagués un moviment, aquest s'executaria directament, o bé, el cas que hi hagués més d'un moviment usaríem la la funció escull_atac_múltiple amb els paràmetres que es mostren a la figura 21.

	Paràmetre	Valor desitjat
Vector d'enters	Nivell d'amenaça de que tindrà la peça quan avanci.	MÀXIM
Vector de booleans 1	La peça té un punt vermell a sota?	CERT
Vector de booleans 2	La peça defensarà alguna altra peça quan avanci?	CERT

Fig. 21 Taula de l'estratègia trampera

7.1.2.8 Protectora

L'estratègia protectora conté tots els moviments de les peces en la seva posició inicial i que puguin atacar peces contràries. Aquesta és la estratègia que evita que l'altre jugador pugui guanyar arribant una peça blanca a l'altre extrem del taulell.

Els dos vectors de booleans, anomenats prot1 i prot2 en l'estratègia protectora, són els que afinen el millor moviment quan el jugador contrari està a punt d'arribar amb una peça a l'última fila de peces.

El prot1 retorna cert si un moviment es aconsellable de realitzar-lo. Això es dóna quan una peça amb un punt vermella pot atacar estant a la posició inicial, o bé, si es blanca i es dóna que sabem que l'altre jugador no te peces vermelles o el jugador contrari té menys peces blanques.

El **prot2** retorna cert quan al atacar no deixa pas a una altre peça del jugador contrari.

Es poden donar 3 casos: el cas que no hi hagués cap peça que pogués protegir la fila inicial. El cas que només hi hagués un moviment, aquest s'executaria directament, o bé, el cas que hi hagués més d'un moviment usaríem la la funció escull_atac_múltiple amb els següents paràmetres:

	Paràmetre	Valor desitjat
Vector d'enters	Nivell d'amenaça de que tindrà la peça quan avanci.	MÀXIM
Vector de booleans 1	prot1	CERT
Vector de booleans 2	prot2	CERT

Fig. 22 Taula de l'estratègia protectora

7.2 Perfil

Entenem perfil com una llista ordenada d'estratègies que s'apliquen seqüèncialment fins que és possible executar-ne alguna. Aquests perfils estan codificats de manera que formen un enter, i cada perfil esmentat en el punt anterior té la següent codificació:

1. Aleatòria
2. Millor
3. Agressiva
4. Defensiva
5. Bloquejant
6. Corredora
7. Trampera
8. Protectora

Aquest enter que formen la unió d'enters sempre comença per 1 i es comencen a executar des de la xifra de les unitats i van avançant cap a l'1. Dins de l'enter del perfil no hi ha cap xifra repetida ja que si una estratègia no es pot executar, tampoc es podrà tornar a executar si no es canvia l'estat actual. Per tal de garantir que es realitzarà un moviment, sempre hi ha d'haver un 1 a la xifra de més pes de l'enter, ja que l'estratègia aleatòria garanteix sempre un moviment.

7.2.1 Automàtic

El Sator té una funció que selecciona un perfil dels que es definiran a continuació: El joc ràpid funciona els primers torns i cap al final de la partida, el perfil agressiu s'activa quan el jugador té mes peces o iguals que el seu rival, el perfil defensiu s'activa quan el jugador te menys peces que el

seu adversari. Totes aquestes estratègies tenen una petita component aleatòria que fa que si es compleix, s'activi l'últim perfil, el perfil aleatori.

7.2.2 De joc ràpid

El perfil de joc ràpid es basa en la idea de realitzar un desenvolupament ràpid del joc, o una conclusió del joc el més ràpid possible. Així doncs, sempre intentarà avançar les peces blanques per guanyar quan abans millor i aplicar estratègies amb millors resultats alfes, i deixar per últimes estratègies la trampera i l'agressiva.

7.2.3 Agressiu

El perfil agressiu té com a objectiu el desgast de peces i l'atac directe com a principals objectius. Aquest perfil és més recomanable quan es disposa de més peces que l'altre o es disposa de certa informació de les peces que encara li queden a l'adversari.

7.2.4 Defensiu

El perfil defensiu té com a objectiu l'estancament del joc i la defensa de les peces clau, deixant de banda les estratègies agressives com a última opció. Aquest perfil és més òptim si el jugador es troba en desavantatge numèrica.

7.2.5 Aleatori

El perfil aleatori (o random) és completament aleatori. Així doncs, només fa servir l'estratègia random. Aquest perfil només s'activa de tant en tant i en contades ocasions a cada joc, i té la seva raó de ser per evitar que cada joc sigui idèntic a l'anterior, tot i que a vegades això impliqui realitzar un moviment que no seria el més òptim.

8. Joc de proves

En aquest capítol es comentaran els resultats de l'us d'una variant del Sator a fi de provar-ne els perfils.

8.1 Entorn de proves

Les proves s'han realitzat des de una terminal del sistema en un Powerbook 12' amb un processador PowerPC G4 a 1,33 Ghz i 768 Mbytes de memòria ram. El SO que corre sobre la màquina és un Mac OS X 10.4.11 (Tiger).

8.2 Resultats obtinguts

S'han realitzar proves entre els cinc perfils, tenint en compte qui comença la partida. Cada resultat s'obté d'executar el programa 100 vegades. El resultat que es mostra és el tant per cent de les partides guanyades pel perfil que es troba a les files de la taula sobre el de les columnes. Els perfils que es troben a les files són els que comencen el joc.

	Automàtic	Joc ràpid	Agressiu	Defensiu	Aleatori
Automàtic	53,00%	50,00%	49,00%	60,00%	57,00%
Joc ràpid	55,00%	62,00%	45,00%	65,00%	56,00%
Agressiu	47,00%	52,00%	22,00%	67,00%	54,00%
Defensiu	40,00%	63,00%	42,00%	76,00%	54,00%
Aleatori	58,00%	58,00%	34,00%	54,00%	47,00%

Fig. 23 Taula de resultats del joc de proves

8.2.1. Perfil automàtic

Quan comença un perfil automàtic el joc, té vora el 50% de probabilitats de guanyar pels perfils automàtic, de joc ràpid i agressiu, però resulta més efectiu contra els perfils defensiu i aleatori (60% victòries).

En el cas que el perfil automàtic dugui les peces blanques, es manté el seu marge de victòries. La única variació és que perd efectivitat contra el perfil aleatori (del 57% de victòries al 42%).

8.2.2 Perfil de joc ràpid

Observem que el perfil de joc ràpid no comença la partida dóna sobre el 50% de victòries contra un perfil automàtic o un perfil agressiu, en enfrontar-se als altres perfils obté sobre el 40% de les victòries.

En el cas de començar aquest perfil, dóna bons resultats si s'enfronta als perfils automàtic i aleatori (sobre el 55% de victòries), molt bons resultats contra un altre perfil de joc ràpid o contra un perfil defensiu (65% victòries), però perd efectivitat contra un perfil agressiu (45% victòries). Aquest últim cas passa perquè el perfil de joc ràpid intenta avançar quan abans millor les peces amb un punt blanc, i el perfil agressiu intenta capturar peces tant bon punt estan en disposició de fer-ho.

8.2.3 Perfil agressiu

Quan el perfil agressiu obre la partida, obté sobre la meitat de victòries contra els perfils automàtic, de joc ràpid i aleatori (~50% victòries). Guanya molta avantatge contra un perfil defensiu (67% de victòries) i esdevé força inútil contra el seu mateix perfil (22% de victòries).

En el cas que no comenci la partida, s'assegura més de la meitat de les victòries contra tots els perfils.

8.2.4 Perfil defensiu

En el perfil defensiu existeix una simetria quan juga contra el perfil aleatori: si comença el defensiu guanya el 54% de les vegades, però si és l'aleatori qui comença, guanye ell el 54% de les vegades.

Per altre banda, el perfil defensiu només guanya per sobre del 50% de les vegades contra el perfil de joc ràpid, ja que el perfil de joc ràpid no pot fer arribar una peça ja que el desenvolupament del perfil defensiu fa que es bloquegin totes les peces.

8.2.5 Perfil aleatori

El perfil aleatori es troba al voltant del 50% de victòries exceptuant el cas quan el perfil aleatori comença i l'altre perfil és un perfil agressiu. En aquest cas el perfil aleatori no funciona perquè quan aquest avança una peça i l'altre perfil pot capturar-la ho fa. Però el perfil aleatori no dóna més pes a un moviment o altre, així si una peça en defensava una altre no té perquè atacar si capturen la primera.

9. Planificació

Al començar el projecte, es va calcular de presentar-lo en 5 mesos. Passades unes setmanes és va veure que es requeria més temps per implementar l'apartat de les IA, i per tant es va allargar el període a 11 mesos.

9.1 Planificació inicial

- Implementar els 3 jocs (7 setmanes)
  - Programa principal (1 setmana)
  - Joc i les variants (6 setmanes)
    - Sator (2 setmanes)
    - Arepo (2 setmanes)
    - Tenet (2 setmanes)

- Creació de les IA (10 setmanes)
  - Conèixer els jocs (1 setmana)
  - Quantificar els moviments (3 setmanes)
  - Crear les estratègies (3 setmanes)
    - Sator (1 setmana)
    - Arepo (1 setmana)
    - Tenet (1 setmana)
  - Crear els perfils (3 setmanes)
    - Sator (1 setmana)
    - Arepo (1 setmana)
    - Tenet (1 setmana)

- Creació de la memòria i la presentació (3 setmanes)

9.2 Planificació real

Del càlcul de 20 setmanes de l'estimació, uns 130 dies aproximadament, passem a 46 setmanes (uns 275 dies). La distribució de càrrega no és uniforme com es va preveure inicialment, ja que el desenvolupament i implementació de la primera IA suposa la major carrega de treball del projecte. Això no passa amb la IA del Arepo ja que es tracta d'una adaptació de la IA del Sator, i el Tenet no disposa d'estratègies, cosa que encara en simplifica més la implementació.

Id.	Nombre de tarea	Comienzo	Fin	Duración
1	Disseny dels jocs	03/03/2008	24/03/2008	16d
2	Implementació Sator	17/03/2008	31/03/2008	11d
3	Implementació Arepo	31/03/2008	14/04/2008	11d
4	Implementació Tenet	14/04/2008	28/04/2008	11d
5	Disseny IA (Sator)	28/04/2008	30/05/2008	25d
6	Implementació IA (Sator)	05/05/2008	08/09/2008	91d
7	Disseny IA (Arepo)	08/09/2008	29/09/2008	16d
8	Implementació IA (Arepo)	22/09/2008	01/12/2008	51d
9	Disseny IA (Tenet)	01/12/2008	15/12/2008	11d
10	Implementació IA (Tenet)	08/12/2008	29/12/2008	16d
11	Memòria del Projecte	29/12/2008	19/01/2009	16d

Fig. 24 Total de dies emprats al projecte

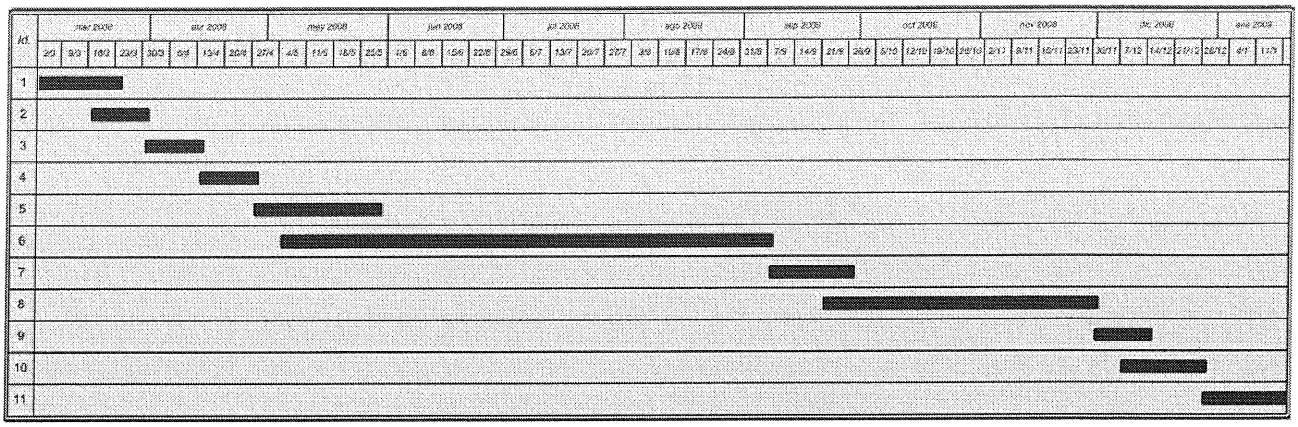

Fig. 25 Diagrama de Gantt

10. Conclusions

En aquest capítol s'exposaran les valoracions del projecte, així com les possibles ampliacions que es podrien realitzar sobre aquest.

10.1 Valoració de l'aplicació

Un dels objectius finals del joc era que provant la IA d'un joc, es notés la dificultat. Aquest objectiu podem dir que s'ha complert en els tres jocs ja que cadascun dels jocs té les seves estratègies i perfils que s'adapten a les normes i millors jugades.

Si l'aplicació hagués de sortir al mercat, o utilitzar-se de manera més pública, podríem dir que es troba en fase beta. Això passa perquè les IA sempre poden millorar: afegint casos a comprovar, estratègies, perfils o canviant els valors de la funció eval.

10.2 Perspectives de futur

El projecte té diverses vies d'ampliació: l'addició de nous jocs, una interfície gràfica o la possibilitat de jugar en línia.

- Tant el Sator, l'Arepo i el Tenet formen part del que s'anomena un quadrat màgic de 5x5, això vol dir que encara hi ha dos jocs de la família que encara no estan implementats.

- Una interfície gràfica milloraria quantitativament la interacció amb l'usuari, fent la navegació i el joc més còmode. També podria ampliar-se per crear una aplicació multi plataforma, ja que l'aplicació no fa servir cap llibreria exclusiva sinó que s'implementa les seves pròpies funcions.

- Una altra modalitat de joc que podria ser interessant seria la de multi jugador en línia. Aquesta opció és molt pràctica si la IA del joc no li presentés cap dificultat al qui juga, o bé, l'usuari volgués jugar amb un amic.

11. Bibliografia

- API de sockets [En línia] <http://www.pcs.cnu.edu/~dgame/sockets/socketsC++> [Consulta: 28 Mar. de 2008]

- Algorisme de joc per 2 jugadors [En línia] <http://www.exaunicen.edu.ar/catedras/aydalgo2/docs/TFca06aCompleto> [Consulta: 9 Abr. 2008]

- Un Algoritme de cerca del 3 en ratlla [En línia] <http://www.iiia.csic.es/~pedro/busqueda6-juegos.pdf> [Consulta: 9 Abr. 2008]

- Funció alfabeta [En línia] <http://www.ia.uned.es/~seve/docencia/intro-ia/p+f/Busqueda-AlfaBetaA.html> [Consulta: 31 Mar. 2008]

- Funció alfabeta [En línia] <http://www.ia.uned.es/~seve/docencia/intro-ia/p+f/Busqueda-AlfaBetaA.html> [Consulta: 30 Mar. 2008]

- Manual de referència C++ [En línia] <http://www.sisoft.ucm.es/Manuales/C++.pdf> [Consulta: 30 Mar. 2008]

- Normativa dels jocs [En línia] <http://www.comascoma.com/cat/index.htm> [Consulta: 3 Mar. 2008]