AIC2019 Game Description

1 Resum del joc i objectius

El joc consisteix en recollir recursos, conquerir pobles i obtenir punts de victòria mentres es combat l'equip enemic. L'objectiu del joc és o bé destruir la base de l'equip enemic, capturar tots els pobles (més informació a la secció 4), o aconseguir 5000 punts de victòria. Si cap equip aconsegueix cap d'aquests objectius durant les 2000 rondes del joc, l'equip guanyador es determina amb el següent criteri, per ordre:

1. Equip que té més punts de victòria.
2. Equip que té més pobles conquerits al final de la partida.
3. Equip tal que la suma de recursos disponibles més el valor de cadascuna de les seves unitats és més gran (utilitzant l'equivalència inicial del mercat, mirar secció 2).
4. Aleatòriament.

2 Economia

Al joc hi ha tres tipus de recursos - la fusta, el ferro i el cristall - que són compartits per tot l'equip. Aquests recursos poden ser recolectats per unitats de tipus treballador als seus respectius depòsits. Cada equip comença amb 200 fusta i 50 ferro i 0 cristall. La fusta i el ferro s'aconsegueixen automàticament al començament de cada ronda a un ritme de 6 i 2 unitats per ronda respectivament.

Les unitats de tipus Base poden intercanviar recursos d'un tipus per un altre amb el banc (més informació a la Secció 9.

3 Mapa

El mapa consisteix en una quadrícula rectangular de dimensions entre 20×20 i 50×50. Totes les unitats ocupen exactament una casella i totes les caselles del mapa són accessibles per unitats excepte si hi ha aigua, muntanya, una altra unitat o un poble. Alhora, cada casella pot contenir un depòsit depòsit de ferro, cristall o fusta (bosc), que els treballadors poden recolectar si estan situats en aquesta casella (més informació a la secció 8.2). Al voltant dels pobles poden haver unitats d'una facció neutral, que es dedicaran a atacar a unitats de qualsevol dels dos equips de la partida si són prou a prop (més informacio a la Secció 4).

Cada unitat té un rang de visió finit, és a dir, només pot observar les caselles que són prou a prop de la seva posició.

Cada equip comença amb exactament una unitat de tipus Base. A més, es garanteix la simetria dels mapes per evitar afavorir un equip sobre l'altre. Aquesta simetria pot ser horitzontal, vertical o rotacional.

A diferència de les unitats standard, les estructures no poden moure's, però algunes d'elles poden atacar.

4 Pobles

En algunes caselles del mapa hi ha situats pobles, que originalment pertanyen a la facció neutral. Aquests pobles inicialment compten amb 50 de lleialtat, i cada cop que són atacats perden un punt de lleialtat per cada punt de mal. Si un equip ataca un poble, aquest pertany a un altre equip i baixa de 0 de lleialtat, el poble passa a ser de l'equip atacant amb un punt de lleialtat. Si el poble era del mateix equip que l'atacant, aquest passa a l'equip enemic. Cada poble recupera un punt de lleialtat per ronda i dona un punt de victòria, 1 fusta, 0.3 ferro i 0.1 cristall per cada 20 punts de lleialtat a l'equip al qual pertany. Si un equip té capturats tots els pobles amb més de 50 de lleialtat cadascun, aquest equip guanya la partida.

A una distància de fins a 25 unitats de cada poble, poden haver-hi unitats neutrals estacionades al comencçament de la partida. Aquestes unitats tenen un protocol molt bàsic per atacar i moure's. També es garanteix la simetria d'aquestes unitats en el mapa.

5 Unitats

Cada unitat rep un identificador (ID) únic escollit aleatòriament entre 1 i 10000. Quan una unitat nova es crea, hi ha un període de 10 torns durant el qual la unitat està en formació i no es pot controlar. Els torns restants de formació d'una unitat donada es poden consultar mitjançant les funcions del controlador.

Hi ha set tipus d'unitats: treballadors, exploradors, soldats, arquers, cavallers, mags i catapultes, els detalls de les quals es mostren en el proper requadre. Totes les distàncies es donen al quadrat. Per exemple, rang d'atac 12 vol dir que una unitat al $(0, 0)$ pot atacar una unitat al $(2, 1)$ ja que $2^2 + 1^2 \leq 12$, però no una al $(2, 3)$. També, un $(*)$ en alguna casella indica que aquesta unitat segueix una mecànica especial en aquell apartat, que serà explicat en detall a la Secció 8.

	Treballador	Explorador	Soldat	Arquer
Cost Fusta	50	50	70	100
Cost Ferro	10	10	30	20
Cost Cristall	0	0	0	0
Vida màxima	20	5	40	10
Atac	0	1	8	3
Rang d'atac	–	5	5	18
Rang mínim d'atac	–	0	0	9
Rang de visió	20	50	25	34
Cooldown al moure's	2	1	2	2.5
Cooldown d'atac	1	1	2	2

	Cavaller	Mag	Catapulta
Cost Fusta	60	100	150
Cost Ferro	50	50	40
Cost Cristall	0	10	0
Vida màxima	100	30	25
Atac	2	9	20*
Rang d'atac	2	5*	50
Rang mínim d'atac	0	0	25
Rang de visió	32	25	20
Cooldown al moure's	1	2	4
Cooldown d'atac	2	3	4

6 Estructures

Hi ha tres tipus d'estructures: la base, els quarters i les torres. La base és una estructura que no pot ser construïda per cap unitat, i cada equip comença exactament amb una. Aquesta estructura pot crear tots els tipus d'unitats. Els quarters són una estructura que pot ser construïda pels treballadors i que pot crear soldats, arquers, cavallers i mags. Finalment les torres són una estructura defensiva que pot ser construïda pels treballadors. Els detalls d'aquestes estructures es mostren en el proper requadre:

	Base	Torre	Quarter
Cost Fusta	–	200	300
Cost Ferro	–	70	100
Cost Cristall	–	0	0
Vida màxima	500	40	200
Atac	7	10	0
Rang d'atac	36*	32	–
Rang mínim d'atac	0	0	–
Rang de visió	50	32	32
Cooldown al moure's	–	–	–
Cooldown d'atac	1	2	–

7 Moviment, atac i cooldowns

Cada unitat té un cooldown de moviment i un cooldown d'atac assignats. La unitat només podrà realitzar l'acció desitjada en cas que el cooldown assignat a aquella acció sigui inferior a 1. En particular, si tots dos cooldowns són inferiors a 1, la unitat pot atacar i moure's al mateix torn.

Cada cop que la unitat es mou o ataca, se li suma certa quantitat al cooldown de l'acció efectuada. La quantitat que es suma està indicada a la taula de la Secció 5. Si la unitat es vol moure en una direcció diagonal, el cooldown de moviment sumat es multiplica per 1.4142 (aproximadament $\sqrt{2}$). Al principi de cada torn tots dos cooldowns baixen en una unitat.

Les unitats, a excepció de bases i catapultes, no poden atacar a través de muntanyes, és a dir, una unitat no pot atacar una casella si el segment que uneix el centre de la unitat al centre de la casella objectiu passa per l'interior d'una casella amb muntanya (es permet, però, que el segment només passi per la cantonada d'una casella amb muntanya). En particular, si un explorador està a sobre d'una muntanya, l'explorador no pot atacar.

8 Mecàniques especials

8.1 Base

És la unitat principal de cada equip, si mor es perd la partida. Els seus atacs fan mal en una regió 3×3 al voltant de la casella atacada. Pot crear totes les unitats de la partida excepte bases, torres i quarters. És l'única unitat que pot intercanviar amb el banc.

8.2 Treballadors

Els treballadors poden construir torres i quarters, i recolectar recursos dels depòsits de fusta, ferro o cristall quan està a sobre d'ells. Més concretament, al recolectar, el pagés emmagatzema 3 unitats de fusta, 1 de ferro o 0.3 unitats de cristall depenent del recurs on estigui situat.

Cada treballador perd un 2% percent dels seus recursos cada ronda. Si els recursos es depositen en la base o en un dels pobles conquerits per l'equip del treballador[1], aquests passen a ser compartits per tot l'equip (de manera que són accessibles per totes les unitats). A diferència dels recursos en possessió dels treballadors, els recursos compartits per tot l'equip no decreixen cada torn.

8.3 Explorador

Els exploradors poden atravessar qualsevol tipus de terreny que no tingui una unitat o un poble a sobre.

8.4 Mag

De forma similar a les bases, els seus atacs fan mal en una regió 3×3 al voltant de la casella atacada.

8.5 Catapulta

Els seus atacs impacten cinc torns més tard sobre la casella atacada. Els atacs de catapulta compten com a atacs de la facció neutral quan s'ataca un poble.

8.6 Quarter

Els quarters poden crear qualsevol unitat del joc excepte bases, torres, quarters i treballadors.

9 Banc

Les bases poden intercanviar recursos amb el banc. La idea general és que mentresmés es compri d'un cert recurs aquest es torna més valuós, i en conseqüència és mes car de comprar però també es ven més car. El valor del recurs es va estabilitzant amb el temps fins i tot si cap dels dos equips

[1] Per fer aquesta acció, cal que el treballador sigui adjacent a l'estructura.

intercanvia aquell recurs. El que s'obté al intercanviar X unitats d'un recurs per un altre en un torn donat es pot cosultar en tot moment mitjançant les funcions del controlador.

A continuació presentem el model exacte que es farà servir en el joc i la intuició que hi ha al darrere. Tanmateix, notem que no és necessari saber tots els detalls per implementar un bot que utilitzi el mercat correctament. El model és el següent: Cada recurs i té un valor v_i i pes w_i associat. El valor v_i és constant durant tota la partida i val 1 per la fusta, 3 pel ferro i 10 pel cristall. El pes w_i inicialment val 0, i indica quantes unitats s'han comprat d'aquell recurs fins el moment. Al intercanviar X unitats del recurs i per un cert recurs j, la fórmula que dona la quantitat del recurs j obtinguda és la següent:

$$T_i^j(X) = v_i v_j^{-1} \int_{w_i}^{w_i+X} e^{C(v_j w_j - v_i t)}\ dt$$

on C és la constant d'inflació, que actualment val 0.01. També, al fer l'intercanvi, w_i augmenta en X unitats. Aquesta fórmula té la següent interpretació: Suposem que existeix un quart recurs R que està disponible només pel banc amb un valor $v_R = 1$. Al intercanviar X unitats d'un recurs i per un recurs j, el banc primer canvia les X unitats d'i per R, i després canvia el que hagi obtingut de R a j. La quantitat de R que el banc dona per cada unitat d'i decreix amb el pes seguint la fórmula $p_i^R(w_i) = v_i e^{-Cv_i w_i}$, però alhora el pes augmenta en Cv_i unitats per cada unitat d'i comprada, amb la qual cosa s'obté que $T_i^R(X) = v_i \int_{w_i}^{w_i+X} e^{-Cv_i t}\ dt$. Alhora, el banc intercanvia R per j a un preu constant: per cada unitat de R el banc dona $v_j e^{-Cv_j w_j}$ unitats de j, resultant en la fórmula per T_i^j descrita anteriorment.

Finalment, abans del torn de cadascuna de les bases, cada pes w_i baixa en $2/v_i$ unitats (fins a un mínim de 0). També, per evitar problemes de precisió, els pesos poden pujar com a molt fins a 300, a partir d'aquest valor es mantenen constants tot i comprar recursos.

10 Comunicació i visió

Cada unitat pot veure només una secció del mapa, que es correspon a les caselles que estiguin dins el seu radi de visió (en unitats quadrades). Si una casella és dins el seu radi de visió, aleshores la unitat pot demanar informació sobre aquesta casella mitjançant el controlador (més informacio a la documentació i a la Secció 12).

Les unitats s'executen independentment i no comparteixen memòria. Per exemple, els objectes visibles per una unitat no són visibles per la resta d'unitats. Tot i així, existeix una única manera de comunicar-se entre unitats: per a cada equip hi ha un array compartit d'enters de mida 300000 on tots els seus components estan inicialitzats a zero. Al seu torn, cada unitat pot llegir una o vàries posicions d'aquest array i/o sobreescriure una o váries posicions mitjan¸cant les funcions *read()* i *write()* del controlador.

11 Energia

L'energia és una mesura aproximada de la quantitat d'instruccions que executa una unitat al llarg d'un torn. Més concretament, cada instrucció de bytecode que executa una unitat costa una unitat d'energia, a excepció de les instruccions de les classes pròpies del joc, dels quals el cost és constant i està disponible a la documentació. Per als usuaris no familiaritzats amb Java, no és necessari saber exactament el que són les instruccions de bytecode, tan sols cal tenir en compte que l'energia consu-

mida creix a mesura que la unitat executa més instruccions, i que es pot comprovar experimentalment quina quantitat queda i quina quantitat es porta gastada amb les funcions del controlador.

Quan una unitat sobrepassa el límit d'energia permesa (actualment posat en 15000 unitats d'energia), aquesta unitat es pausa i continua el seu torn a la següent ronda. Es recomanable procurar mai passar-se de l'energia máxima permesa, ja que perdre un torn pot ser crucial en moltes situacions.

12 Instruccions per l'usuari

Els jugadors han d'omplir la funció *run()* de la classe UnitPlayer. D'entrada es passa un controlador (UnitController) per la unitat, que permet a l'usuari donar ordres a l'unitat i relacionar-se amb l'entorn. Per exemple, la classe UnitController té funcions per detectar el que hi ha a certa casella, per donar ordres d'atacar/moure's/crear unitats/dividir-se/absorbir/etc, o per llegir/escriure a l'array compartit. Per més detalls, a la documentació hi ha disponible tota la informació necessària sobre les classes pr'opies del joc.

La funció run() funciona de la següent manera: quan es genera una unitat nova i acaba el seu període de construcció s'executa la funció *run()* d'aquesta unitat. La funció *run()* s'executa fins cridar la funcio *yield()* del controlador o fins que es supera el límit d'energia permesa (més informació a la Secció 11). Per aquest motiu, un cop una unitat acaba de fer totes les tasques desitjades (atacar, moure's, comunicar-se, etc.) s'ha de cridar la funció *yield()*. Així s'indica que es vol acabar el torn d'aquesta unitat. Si no s'indica, la funcio *run()* continuarà executant-se fins superar l'energia permesa i acabarà el torn automàticament aleshores. S'ha de tenir en compte que no acabar el torn cridant *yield()* pot portar a comportaments de la unitat no desitjats en el futur, ja que és difícil predir des d'on tornarà a executar-se al següent torn.

Si en algun moment la funcio *run()* retorna (acaba), la unitat mor. Per aquest motiu és recomanable assegurar que la funció *run()* mai retorni, per exemple amb un bucle *while(true)*. En general, és recomanable utilitzar l'esquema disponible als exemples *nullplayer* i *demoplayer*.

13 Informació sobre la implementació

Aquesta secció es pot ignorar si l'usuari no est'a prou familiaritzat amb Java.

Cada unitat s'executa en un thread independent que es pausa al finalitzar el seu torn. Aquest thread es reactiva a cada ronda de la partida mantenint l'ordre d'execució relatiu entre unitats. Per seguretat, es prohibeix l'accés a totes les classes de Java fora de *lang*, *math*, i *util*, i fins i tot a algunes sub-classes i funcions d'aquestes classes. De totes maneres, aquestes classes són totalment prescindibles pel joc actual.

També, per motius de seguretat, es prohibeix l'ús de variables estàtiques[2].

[2]Som conscients que hi ha maneres sofisticades de compartir memòria entre threads sense fer servir variables estàtiques. Els codis pujats seran revisats manualment per evitar aquests casos, i en cas que un equip intenti violar aquesta restricció, aquest equip serà desqualificat.