Bug Wars: Descripció del joc

1 Lore

Dues colònies d'insectes s'han establert molt a prop l'una de l'altra i els recursos naturals no donen prou abast per les dues. Qui guanyarà la guerra que està a punt d'esclatar?

2 Resum del joc i objectius

El joc consisteix a destruir totes les reines de l'equip enemic. L'equip que aconsegueixi aquest objectiu guanya la partida. En cas que al cap de 3000 torns cap equip hagi aconseguit aquest objectiu, l'equip guanyador es determina amb el següent criteri, per ordre:

1. Equip que té més reines vives.
2. Equip tal que la suma total de la vida de totes les seves reines és més gran.
3. Equip tal que la suma de recursos disponibles més el valor de cadascuna de les seves unitats és més gran.
4. Aleatòriament.

3 Economia

L'únic recurs del joc és el menjar, que es consumeix al crear noves unitats. Aquest recurs és compartit per tot l'equip i s'aconsegueix automàticament al començament de cada ronda a un ritme de 10 unitats de menjar per ronda. Alhora, les formigues poden minar menjar adicional a certes caselles especials.

4 Mapa

El mapa consisteix en una quadrícula rectangular de dimensions entre 20×20 i 80×80. Totes les unitats ocupen exactament una casella i totes les caselles del mapa són accessibles per unitats excepte si hi ha una pedra o una altra unitat. Les pedres, tot i ser inaccessibles, es poden destruir si són atacades prou cops.

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 casella del mapa, independentment de si és accessible o no, pot contenir un dipòsit de menjar que pot ser minat per les formigues a un ritme de tres unitats de menjar per torn. Aquests dipòsits regeneren un punt de menjar cada ronda, però mai superen el valor inicial.

Cada equip comença amb entre una i cinc reines sobre el mapa, a més es garanteix la simetria dels mapes per evitar afavorir un equip sobre l'altre. Aquesta simetria pot ser horitzontal, vertical o rotacional.

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 és un capoll i no es pot controlar. Els torns restants de capoll d'una unitat donada es poden consultar mitjançant les funcions del controlador.

Hi ha cinc tipus d'unitats: reines, formigues, escarabats, aranyes i abelles. Les reines són la unitat principal del joc. Tot i no poder atacar, són les úniques que poden crear noves unitats i són el primer criteri de desempat. A més, les reines poden curar una unitat cada ronda. Les formigues tenen un poder de combat molt limitat, però són les úniques unitats del joc que poden minar menjar. Els escarabats, les aranyes i les abelles són unitats de combat amb diferents característiques, que es detallen al 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 $(2, 3)$.

	Reina	Formiga	Escarabat	Aranya	Abella
Cost	-	150	200	280	300
Vida Màxima	250	15	45	10	100
Atac	-	1	4	3	1
Rang d'atac	-	5	5	18	5
Mínim rang d'atac	-	-	-	9	-
Rang de visió	36	24	24	32	36
Cooldown de moviment	3	2	2	2	1
Cooldown d'atac	-	2	2	2	1
Mecàniques especials	Si	Si	No	No	No

Les mecàniques especials estan detallades a la secció 7.

6 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 se suma està indicada a la taula de la secció 5. Tanmateix, si la unitat es vol moure en una diracció diagonal, el cooldown de moviment sumat es multiplica per 1.4142, que és aproximadament $\sqrt{2}$. Al principi de cada torn tots dos cooldowns baixen en una unitat.

Les unitats no poden atacar a través de pedres, és a dir, una unitat no pot atacar una casella si el segment que uneix el centre d'una unitat al centre de la casella objectiu passa per l'interior d'una

casella amb una pedra (es permet, però, que el segment només passi per la cantonada d'una casella amb pedra). A més, les unitat poden atacar pedres no obstruïdes per altres pedres.

7 Mecàniques especials

Les reines i les formigues poden rebre altres ordres a més d'atacar o moure's.

Reina:

- Crear unitats: Cada reina pot crear una unitat cada torn en una posició adjacent[1] a la seva mentres hi hagi prou recursos.
- Curar unitats: Cada reina pot curar un punt de vida per torn a una unitat que no sigui de tipus reina en un radi de cinc unitats quadrades, és a dir, tal que la diferència (x, y) entre les posicions satisfaci $x^2 + y^2 \leq 5$.

Formiga:

- Minar menjar: Cada formiga pot minar fins a tres unitats de menjar per torn d'una casella adjacent que contingui menjar.

8 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 10).

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 100000 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 vàries posicions mitjançant les funcions *read()* i *write()* del controlador.

9 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

[1] Que es pot accedir mitjançant una de les vuit direccions que no siguin zero.

exactament el que són les instruccions de bytecode, tan sols cal tenir en compte que l'energia consumida 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. És recomanable procurar mai passar-se de l'energia máxima permesa, ja que perdre un torn pot ser crucial en moltes situacions.

10 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 la 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/curar/minar, o per llegir/escriure a l'array compartit. Per més detalls, a la documentació hi ha disponible tota la informació necessària sobre les classes pròpies del joc.

La funció *run()* funciona de la següent manera: quan es genera una unitat nova i surt del capoll s'executa la funció *run()* d'aquesta unitat. La funcio *run()* s'executa fins cridar la funcio *yield()* del controlador o fins que se supera el límit d'energia permesa (més informació a la secció 9). 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*.

11 Informació sobre la implementació

Aquesta secció es pot ignorar si l'usuari no està 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]. Més endavant cada unitat s'executarà en un procés independent amb la seva propia JVM i es podran tornar a utilitzar.

[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.