# AIC2023: Home Run

## 1 Lore

Estava sent una partida molt intensa. Hi havia milers d'espectadors mirant la final entre els dos millors equips de la MLB (Major League Baseball). Hagués pogut guanyar qualsevol dels dos, però de sobte va començar a ploure. Plovia com no havia plogut des de feia anys, i semblava que continuaria així durant uns quants dies. La final es va acabar cancel·lant, la partida determinada com a empat, i tothom va haver d'anar cap a casa.

Però els equips finalistes de beisbol no estaven disposats a deixar córrer la final només per quatre gotes. Com que els àrbitres no els venia gens de gust mullar-se, els equips van acabar creant unes normes noves per determinar el guanyador.

## 2 Resum del joc i Objectius

En aquest joc, cada equip pot reclutar nous jugadors, aconseguir punts i reputació, comprar pilotes de beisbol i eliminar oponents, tirant-los a l'aigua o fent-los col·lisionar. Si un equip aconsegueix 2000 punts, guanya. Si cap equip ha guanyat al cap de 1500 rondes, el guanyador es determina seguint el següent criteri, per ordre:

1. L'equip amb més punts.
2. L'equip tal que la suma de recursos + el valor de totes les seves unitats és més gran.
3. Aleatòriament.

## 3 Economia

Hi ha dos tipus de recursos en aquest joc: **punts** i **reputació**. Els punts són necessaris per guanyar, i són la primera raó de desempat si una partida arriba al nombre màxim de rondes. La reputació s'utilitza per reclutar noves unitats i per comprar pilotes de beisbol.

Al principi de cada ronda, tots els equips reben 2 de reputació. A més, es poden aconseguir punts o reputació adicional quan un pitcher comença el seu torn sobre una **Base** o **Estadi**, respectivament. Les bases donen 1 punt, i els estadis donen 1 de reputació. Una base no pot donar puntuació més d'una vegada per ronda, per equip. Cada equip comença amb 0 punts i 100 de reputació.

Com que com més dura el partit, més audiència arreplega, hi ha una mica d'inflació tant en punts com en reputació. Això fa que els punts i reputació que s'obtenen, ja sigui de la manera que sigui, incrementi en 0.001 cada ronda. Per exemple, si un pitcher comença el seu torn sobre una base a la ronda 1000, obtindrà 2 punts en comptes d'1 (el creixement no és exponencial).

# 4 Mapa

El mapa consisteix en una quadrícula de dimensions entre $20 \times 20$ i $60 \times 60$. Cada unitat ocupa exactament una casella i tota casella pot ser ocupada per una unitat excepte si hi ha aigua, una altra unitat o una pilota de beisbol. Cada casella que no sigui aigua pot tenir una Base o un Estadi a sobre.

Cada unitat només pot detectar els objectes al mapa que estiguin dintre del seu radi de visió. La visió entre unitats no és compartida. És a dir, una unitat no pot interactuar (veure si hi ha una unitat, de quin tipus és...) amb un objecte que té fora del rang de visió.

Cada equip comença amb exactament un HQ. A més, es garanteix la simetria dels mapes per evitar afavorir un equip sobre l'altre. Aquesta simetria pot ser horitzontal, vertical o rotacional. Les caselles d'un mapa donat tindran un offset fix entre 0 i 1000 respecte la seva posició "real" (per exemple, la casella que seria la $(0, 0)$ serà la $(offset_x, offset_y)$), de tal manera que les unitats no puguin deduir les fronteres del mapa sense explorar primer.

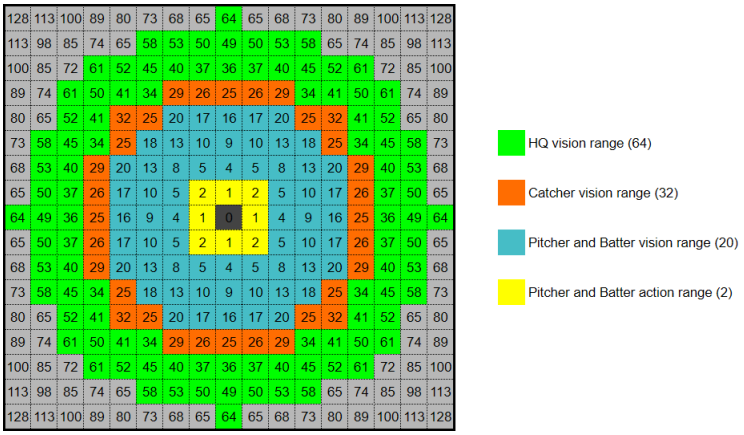
# 5 Unitats i Estructures

Cada unitat té un identificador únic (ID), escollit aleatòriament entre 1 i 10.000.

Hi ha quatre tipus d'unitats: HQ, Pitchers, Batters i Catchers, les característiques dels quals 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 $(2, 3)$. També, un $(*)$ en alguna casella indica que aquesta unitat segueix una mecànica especial en aquell apartat, que serà explicat en detall a la Secció 7.

| | HQ | Pitcher | Batter | Catcher |
|---|---|---|---|---|
| Cost Reputació | – | 50 | 75 | 35 |
| Rang de Moviment | 0 | 2 | 2 | 2 |
| Rang de Visió | 64 | 20 | 20 | 32 |
| Cooldown al moure's | – | 1.5 | 1.5 | 1 |
| Cooldown després d'actuar | $0^{(*)}$ | $2^{(*)}$ | $1^{(*)}$ | $-^{(*)}$ |

La següent imatge mostra tots els rangs d'unitat presents al joc:



# 6 Moviment, accions i cooldowns

Cada unitat té un cooldown de moviment i un cooldown d'acció. Només es poden moure, o actuar, quan el seu cooldown és estrictament menor que 1. En particular, si ambdós cooldowns són menors que 1, la unitat podrà actuar i moure's al mateix torn.

Cada vegada que una unitat actua o es mou, s'afegeix un cooldown. La quantitat afegida està especificada a la taula de la secció 5. Si una unitat es mou en diagonal, el cooldown de moviment que s'afegeix es multiplica per 1.4142, que és aproximadament $\sqrt{2}$. Al principi de cada torn, ambdós cooldowns es redueixen en 1.

# 7 Mecàniques especials

En aquesta edició, totes les unitats tenen mecàniques especials:

**HQ:** Pot reclutar noves unitats i construir pilotes de beisbol en posicions adjacents[1]. Construir una pilota costa 1 de reputació.

**Pitcher:** Si comença el seu torn sobre una base o estadi, l'equip al qual pertany aconsegueix un punt o un de reputació, respectivament. A més, els pitchers poden agafar pilotes adjacents, portar-les (a sobre) i deixar-les a una casella adjacent buida.

**Batter:** Pot batejar pilotes de beisbol i altres unitats (que no siguin HQ) fins a una distància de 3 caselles. Quan un objecte $A$ és batejat, es mou en la direcció que ha sigut batejat (és a dir, la direcció del batter a $A$) fins que col·lisiona amb un altre objecte $B$ (si és que col·lisiona). El resultat que provoca una col·lisió entre $A$ i $B$ depèn dels dos objectes, i segueix aquest comportament:

- **Pilota col·lisiona amb Pitcher o Batter:** El Pitcher o Batter és eliminat, i la pilota continua el seu camí.
- **Pilota col·lisiona amb Catcher:** La pilota és detinguda pel Catcher. El Catcher no s'elimina.
- **Pilota col·lisiona amb una altra Pilota:** Les dues pilotes es destrueixen.
- **Pilota col·lisiona amb HQ:** L'equip contrari del HQ aftectat és recompensat amb 100 punts, i la pilota es destrueix.
- **Pitcher, Batter o Catcher col·lisiona amb Pilota o HQ:** El Pitcher, Batter o Catcher s'elimina.
- **Pitcher, Batter o Catcher col·lisiona amb Pitcher o Batter:** Ambdues unitats són eliminades.
- **Pitcher, Batter o Catcher col·lisiona amb Catcher:** El Catcher para la unitat. No s'elimina cap unitat.

Cal remarcar que l'equip el qual pertany una unitat no afecta el resultat d'una col·lisió.

A més, si un objecte es mou (o és mogut, perquè ha sigut batejat) a una casella d'aigua o fora del mapa, queda destruït.

[1] Posicions a disància al quadrat menor o igual a 2.

# 8 Comunicació i visió

Cada unitat només pot veure els objectes (unitats, pilotes, etc.) que estan dintre el seu radi de visió. La visió no es comparteix entre unitats. Per tant, que una unitat vegi un objecte no implica que una altra unitat pugui també.

Les unitats s'executen independentment i no comparteixen memòria. Tot i així, es poden comunicar a través d'un array compartit de 1000000 enters (int). Els valors de l'array estan inicialitzats a 0. Tant llegir com escriure sobre l'array es pot fer utilitzant els mètodes presents al *UnitController*.

# 9 Ordre d'Unitats i Scheduling

Les unitats executen el seu torn cada ronda. Primer, totes les unitats del primer equip estan "scheduled"(en una llista, per ordre), i tot seguit les del segon equip. Quan un HQ crea una altra unitat, aquesta unitat s'insereix a la llista immediatament després del HQ a la cua de torns. Tanmateix, una unitat poden modificar l'ordre de torns fent "scheduling"(és a dir, especificant una unitat que s'executarà just després d'acabar el torn d'aquesta unitat) a una unitat que encara no s'ha executat en aquesta ronda. Quan una unitat $A$ fa "schedule"a una unitat $B$, el torn de $B$ es posa a la cua just després del torn de $A$, i es manté d'aquesta manera tret que una altra unitat faci "schedule"de $A$ o de $B$, o si $A$ fa schedule a una altra unitat (en aquest cas, aquesta nova unitat, en comptes de $B$, s'executarà just després de $A$).

Per norma, el HQ és sempre la primera unitat que s'executa al principi de la ronda d'un equip.

# 10 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 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. Es recomanable procurar mai passar-se de l'energia máxima permesa, ja que perdre un torn pot ser crucial en moltes situacions.

# 11 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 què hi ha a certa casella, per donar ordres d'atacar/moure's/crear unitats/etc. 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 perídode 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 10). 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*.

# 12 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], i això inclou també el "switch", ja que utilitza variables estàtiques internament.

[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.