

Bola de Drac 2015

Jordi Petit Enric Rodríguez Salvador Roura

15 de maig de 2015

1 Regles del joc

Aquest joc s'inspira en la mítica sèrie de dibuixos animats Bola de Drac (vegeu per exemple http://ca.wikipedia.org/wiki/Bola_de_Drac).

Cada jugador controla un personatge anomenat *goku*, el protagonista de la sèrie. Els gokus es mouen en un tauler rectangular que conté un laberint. L'objectiu d'un goku és agafar *boles de drac* i seguidament dipositar-les en alguna de les càpsules *Hoi Poi* que hi ha repartides per tauler. Cada goku només pot dur a sobre una bola de drac com a molt en cada moment. El nombre de boles de drac que hi ha al tauler és sempre el mateix, comptant tant les que duu algun goku com les que encara estan lliures. Quan una bola de drac és dipositada en alguna càpsula Hoi Poi, immediatament n'apareix una altra en una casella aleatòria del tauler.

Els gokus poden combatre entre ells, i d'aquesta manera prendre la bola de drac que duu un altre. Els combats tenen lloc quan dos gokus coincideixen en una casella, i el seu resultat depèn de la *força* de cadascun dels combatents. Una altra manera d'impedir que un altre goku dipositi la bola de drac que duu és llançant un raig *Kame Hame* que l'elimini.

A part de boles de drac i càpsules Hoi Poi, a les caselles del tauler poden haver-hi també *mongetes màgiques*, que permeten recuperar la força perduda, i *núvols Kinton*, sobre els quals els gokus es mouen el doble de ràpid i sense cansar-se.

Quan el joc acaba, és a dir, s'han realitzat totes les rondes, el jugador que ha dipositat més boles de drac guanya la partida. En cas d'empat, guanya aquell que conservi més força.

A continuació es donen les regles del joc amb més detall:

- Els següents paràmetres configuren una partida:
 - *nb_players*(): Nombre de jugadors.
 - *nb_rounds*(): Nombre de rondes de la partida.
 - *rows*(): Nombre de files del tauler.
 - *cols*(): Nombre de columnes del tauler.
 - *nb_capsules*(): Nombre de càpsules Hoi Poi.
 - *nb_balls*(): Nombre de boles de drac.

- *nb_beans()*: Nombre de mongetes màgiques.
- *nb_kintons()*: Nombre de núvols Kínton.
- *max_strength()*: Força màxima dels gokus.
- *res_strength()*: Força dels gokus després de ressuscitar.
- *moving_penalty()*: Penalització de força per moure's.
- *kamehame_penalty()*: Penalització de força per llançar un raig Kame Hame.
- *combat_penalty()*: Penalització de força per combatre.
- *goku_regen_time()*: Temps de regeneració per als gokus.
- *bean_regen_time()*: Temps de regeneració per a les mongetes.
- *kinton_regen_time()*: Temps de regeneració per als núvols Kínton.
- *kinton_life_time()*: Temps de vida dels núvols Kínton.

- Els diferents tipus de casella són:

- *Empty*: Casella buida.
- *Rock*: Una roca.
- *Capsule*: Una càpsula Hoi Poi.
- *Ball*: Una bola de drac.
- *Kinton*: Un núvol Kínton.
- *Bean*: Una mongeta màgica.

El tauler sempre estarà tot rodejat de roques.

- Cada casella pot rebre la visita d'un (i només un) goku. Els gokus poden trobar-se en un d'aquests estats:

- *Normal*: Un goku normal.
- *On_Kinton*: Un goku muntat sobre un núvol Kínton.
- *With_Ball*: Un goku amb una bola de drac.
- *On_Kinton_With_Ball*: Un goku muntat sobre un núvol Kínton amb una bola de drac.

Cada goku pertany a un jugador i pot estar viu o eliminat temporalment.

- Inicialment tots els gokus es troben al màxim nivell de força *max_strength()* i els marcadors de boles dipositades estan a zero.
- A cada ronda, cada jugador pot demanar una acció, és a dir, pot triar —independentment dels altres jugadors— què vol que faci el seu goku: moure's cap a una casella adjacent seguint una direcció, o llançar un raig Kame Hame cap a una direcció (però no les dues coses alhora!). Els gokus sense núvol Kínton només poden realitzar accions en les rondes parells. Per contra, un goku muntat sobre un núvol Kínton pot moure's o llançar rajos Kame Hame en totes les rondes. La primera ronda és la ronda 0.

Les direccions possibles són cap amunt, cap avall, cap a la dreta o cap a l'esquerra. Els gokus no es poden moure a les caselles on hi ha roques.

En cas de demanar diverses accions a un goku, només es considerarà la primera. Per altra banda, les accions il·legals seran ignorades. Així doncs, pot ser que no totes les accions demanades siguin finalment concedides.

- Per tal que un goku agafi una bola de drac, només cal que passi per una casella amb bola de drac. Si un goku amb bola de drac passa per una casella amb una altra bola de drac, no passa res: el goku continua amb la seva bola i l'altra bola queda a la casella.

- Per tal que un goku amb una bola de drac la dipositi, només cal que passi per una de les *nb_capsules* () caselles amb càpsula Hoi Poi. Si un goku sense bola de drac passa per una casella de càpsula Hoi Poi, no passa res. El nombre de boles de drac al tauler és sempre *nb_balls* (), comptant tant les que duu algun goku com les que encara estan lliures. Quan una bola de drac és dipositada, immediatament n'apareix una altra en una casella aleatòria del tauler.

- Repartides pel tauler hi ha *nb_beans* () mongetes màgiques. Quan un goku passa per una casella que conté una mongeta màgica, la consumeix i així restitueix la seva força al valor màxim inicial.

Una vegada consumida, la mongeta màgica torna a aparèixer a la mateixa casella a partir d'un mínim de *bean_regen_time* () rondes, quan no hi ha cap goku a la casella.

- Repartits pel tauler hi ha *nb_kintons* () núvols Kínton. Quan un goku passa per una casella que conté un núvol Kínton aleshores s'hi puja. Passades *kinton_life_time* () rondes, el núvol Kínton desapareix i el goku torna a ser normal.

Si en passar per una casella de núvol Kínton el goku ja estava muntant un núvol Kínton, aleshores la vida del núvol del goku s'incrementa en *kinton_life_time* () rondes, i el núvol de la casella desapareix.

Una vegada consumit, el núvol Kínton torna a aparèixer a la mateixa casella a partir de *kinton_regen_time* () rondes, quan no hi ha cap goku a la casella.

- En una casella no pot haver-hi dos gokus alhora. Quan un goku es mou a una casella on ja hi ha un altre goku, ja sigui perquè aquest segon encara no s'ha mogut o bé perquè s'hi acaba de moure, té lloc un combat. En un combat entre els gokus *A* i *B*, la probabilitat que *A* en sigui el vencedor és

$$\frac{1 + \text{força}(A)}{2 + \text{força}(A) + \text{força}(B)}$$

. El vencedor del combat es queda a la casella i la seva força després de lluitar és

$$\max(0, \text{força} - \text{combat_penalty}())$$

. El perdedor queda eliminat (i la seva força passa a ser 0; això és important a efectes de puntuació, vegeu més endavant).

Si el goku vençut o la casella on s'ha produït el combat tenen bola de drac, aquesta passa a mans del vencedor. Les boles de drac sobrants (per exemple, si el vencedor ja té bola de drac) reapareixen en una posició aleatòria del tauler.

En cas que el goku vençut estigui muntat sobre un núvol Kínton, aquest passa a mans del vencedor. Si el vencedor ja està muntat sobre un núvol Kínton, aleshores a aquest se li afegeix la vida del núvol del vençut, que desapareix.

- Un raig Kame Hame elimina tot goku que hi hagi en línia recta des de la posició on hi ha el goku llançador fins a la propera casella de roca, en la direcció indicada. La força dels gokus eliminats queda a 0. Les seves boles de drac reapareixen aleatòriament en una altra casella del tauler. Els seus núvols Kínton desapareixen.
- Les accions que realitzen els gokus poden consumir força. Aquest desgast depèn del goku i de l'acció:
 - Llançar un raig Kame Hame consumeix *kamehame_penalty()* unitats de força. Si un goku no disposa almenys d'aquesta quantitat de força, no pot llançar un raig Kame Hame.
 - Si un goku està muntat sobre un núvol Kínton, no perd força quan es mou. Si no té núvol Kínton:
 - * quan la ronda és congruent amb 2 mòdul 4, el goku només es pot moure si té almenys *moving_penalty()* unitats de força, i perd aquesta quantitat de força després del moviment;
 - * quan la ronda no és congruent amb 2 mòdul 4, no hi ha penalització per moviment. Això implica que un goku sense força i sense núvol Kínton només es pot moure en una de cada quatre rondes!

El desgast de força és independent de si el goku porta bola de drac o no.

- Les accions demanades pels jugadors a cada ronda s'executen de la forma següent. Inicialment es determina un ordre aleatori dels jugadors. Aleshores, seguint aquest ordre, s'executen primer els rajos Kame Hame, i després es realitzen els moviments (i els combats que en resultin).

Més concretament, el llançament de Kame Hames funciona de la forma següent. A cada ronda només es pot llançar un sol Kame Hame. Seguint l'ordre dels jugadors, suposem que el primer jugador que ha demanat un Kame Hame és A . Si $força(A) < kamehame_penalty()$, o és ronda senar i A no té núvol Kínton, aleshores s'ignora la petició de A , tal com s'ha

explicat anteriorment. Altrament A té una probabilitat

$$\frac{1 + \text{força}(A) - \text{kamehame_penalty}()}{1 + \text{max_strength}() - \text{kamehame_penalty}()}$$

que se li concedeixi el llançament. Es repeteix el procés per cada jugador seguint l'ordre, fins que es concedeixi un Kame Hame o no hi hagi més jugadors. Una vegada s'ha concedit un Kame Hame, les peticions de Kame Hame dels jugadors següents són ignorades.

- Al final de cada ronda, s'actualitzen els comptadors de regeneració i de vida. Això permet que els núvols Kinton i les mongetes màgiques tornin a aparèixer a les seves caselles, i que els núvols Kinton que han esgotat la seva vida desapareguin.

Els gokus eliminats també poden ressuscitar. Després de com a mínim $\text{goku_regen_time}()$ rondes, un goku eliminat torna a aparèixer en estat normal en una casella aleatòria amb el nivell $\text{res_strength}()$ de força. Els gokus sempre ressusciten en posicions buides i sense altres gokus al voltant.

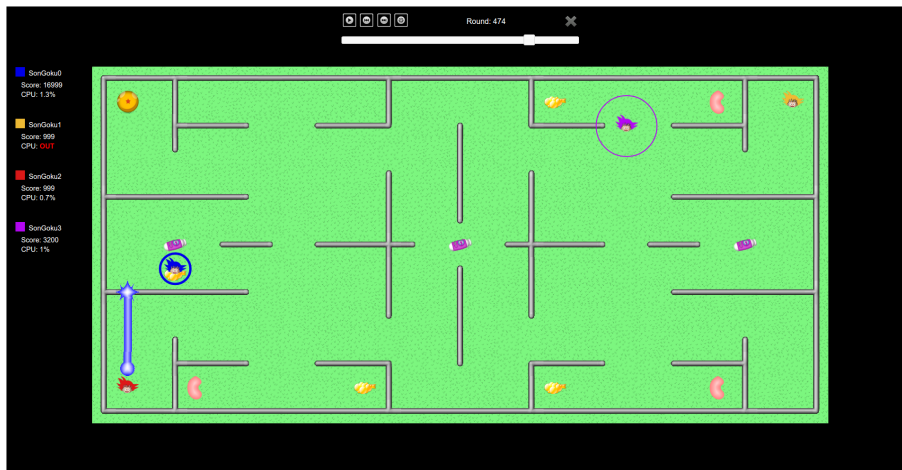
- Si en una partida en el Jutge un jugador comet un error d'execució o esgota el seu temps de càlcul (2 segons per tota la partida), el seu goku queda *congelat*, no juga més.
- En acabar totes les rondes de la partida, el rànking dels jugadors es calcula segons la seva puntuació, que segueix la fórmula:

$$\text{puntuació} = \text{boles} \cdot (\text{max_strength}() + 1) + \text{força}.$$





D'aquesta manera, guanya qui més boles de drac hagi dipositat, i en cas d'empat, el que tingui més força. Està permès que el guanyador pugui tenir el goku congelat.

2 El Visor

A continuació es mostra una captura de pantalla del visor on apareixen pràcticament tots els elements del joc:



- A la part superior apareixen botons que permeten pausar/reproduir, anar al principi i al final de la partida, desactivar el mode animació, o tancar el visor. Una barra horitzontal indica visualment en quin punt de la partida es troba la ronda actual. Pitjant 'h' s'obre una finestra d'ajuda que explica com els botons del teclat també permeten controlar la reproducció de la partida.
- A la part esquerra hi ha el marcador. Cada jugador hi apareix amb el seu nom i color corresponent. El marcador indica la seva puntuació i, en l'execució en el Judge, el percentatge de CPU consumit (quan el jugador ha quedat congelat, s'indica amb un 'OUT' en color vermell).
- Els gokus envoltats per un cercle petit i gruixut porten una bola de drac. A la captura de pantalla, és el cas del goku blau (que a més està muntat sobre un núvol Kínton).
- Els gokus envoltats per un cercle gran i fi acaben de ressuscitar. A la captura de pantalla, és el cas del goku lila.
- A la captura de pantalla, el goku vermell està llançant un raig Kame Hamme.
- Els altres elements que apareixen a la captura de pantalla són els següents:

-  Bola de drac
-  Núvol Kínton
-  Mongeta màgica
-  Càpsula Hoi Poi

3 Programació

Al problema de Bola de Drac 2015 del Jutge, juntament a aquest enunciat, hi trobareu el material necessari per programar el joc en C++. En concret, hi teniu tots els fitxers de suport, exemples de jugadors, i un visor de partides amb els quals podreu desenvolupar i provar els vostres jugadors.

Necessitareu g++, make i un navegador recent (Firefox, Chrome, Chromium, preferiblement un d'aquests dos últims). El codi del joc és portable a qualsevol sistema Linux, Mac o Windows, suposant que hi instal·leu les eines adequades. Els ordinadors de les aules de laboratori ja tenen aquest software instal·lat. Si voleu treballar amb altres màquines:

- Amb Debian o Ubuntu, amb una instrucció del tipus

```
sudo apt-get install build-essential chromium-browser
```

tindreu tot el que us cal.
- Amb Mac, en tindreu prou amb instal·lar XCode des de l'App Store.
- Amb Windows, serà suficient instal·lar MinGW (<http://mingw.org>).

3.1 Com fer un jugador

Per fer un jugador, copieu primer el fitxer `AINull.cc` a un fitxer `AIXXX.cc`, on `XXX` és un identificador de la vostra elecció. Trieu un identificador no ofensiu, que no hagi estat triat ja per un altre estudiant, i compost per, com a molt, 12 lletres, dígit i caràcters de subratllat; per exemple, `SonGoku`.

A continuació, al fitxer `AISonGoku.cc` (o com l'hagueu anomenat) que acabeu de crear, heu de canviar la línia 11 per posar-hi el nom del vostre jugador (a l'exemple, `#define PLAYER_NAME SonGoku`).

Finalment, heu d'implementar el vostre jugador tot completant la classe `PLAYER_NAME` que hereta les operacions de consulta del tauler (classe `Board`) i de creació d'accions (classe `Action`) a través de la classe base `Player`. El mètode `play()` es crida a cada ronda per transmetre-us l'estat actual del tauler i recollir les accions del vostre jugador. A la vostra classe hi podeu afegir camps (variables) per recordar l'estat d'una ronda a l'altra, mètodes (funcions) per descompondre el vostre programa, etc.

Fixeu-vos que la vostra classe ha de contenir una funció anomenada `factory()` que no heu de modificar, i que després de la classe també hi ha una crida per registrar el vostre jugador que tampoc heu de modificar.

Podeu prendre com a referència el jugador `AIDemo.cc` que s'adjunta amb el material de la pràctica.

3.2 Com executar i veure partides localment

1. Editant el Makefile tal com allà s'indica, podeu incorporar el fitxer `AIDummy.o` a la compilació per a poder usar el jugador Dummy a les vostres partides.

Podeu fer el mateix amb els fitxers .o d'altres jugadors.

2. Executeu `make` per compilar tots els fitxers que calguin i crear l'executable `Game`.
3. Per disputar una partida amb els paràmetres de joc `default.cnf`, executeu una comanda com ara

```
./Game Null SonGoku Demo Demo < default.cnf > default.res
```

Aquí, el primer jugador serà `Null`, el segon `SonGoku`, i els altres dos `Demo`. El resultat de la partida quedarà a `default.res`.

4. Visualitzeu la partida obrint el visor (`viewer.html`) amb el vostre navegador i carregueu el fitxer `default.res`.

Podeu obtenir la llista completa de paràmetres del programa `Game` fent `./Game --help`. En particular, `./Game --list` us llistarà els noms dels jugadors inclosos.

Si us cal, podeu netejar el vostre directori de fitxers executables i fitxers objectes amb la comanda `make clean`.

3.3 Restriccions

Els codis dels vostres jugadors han de complir certes condicions:

- Tot el codi s'ha de trobar en un sol fitxer i ha de seguir les indicacions donades.
- No podeu fer servir variables globals (utilitzeu camps de la vostra classe `PLAYER_NAME`).
- Només podeu usar llibreries estàndard de C++: `vector`, `map`, etc.
- No podeu obrir fitxers, ni fer altres crides al sistema operatiu (`systems`, `forks`, etc.).
- Si us cal, podeu utilitzar `cerr`, però no `cin` ni `cout`. Compte, escriure missatges consumeix temps!
- En execució local, no es controla que els jugadors avortin, que triguin massa, o que interfereixin amb els contraris. Al Jutge, sí.

3.4 Estructures de dades

Per saber com consultar el tauler, feu un cop d'ull al fitxer `Board.hh` (en particular, pareu atenció a les operacions públiques de la classe `Board`). Per saber com sol·licitar les accions, mireu el fitxer `Action.hh` (en particular, fixeu-vos en les operacions públiques de la classe `Action`). Tingueu en compte que només podeu utilitzar les operacions públiques d'aquestes classes. També us pot resultar útil mirar-vos els fitxers `Player.hh`, `PosDir.hh` i `AIDemo.cc`.

4 Consells

Us recomanem seguir els següents consells:

- Estudieu només la part pública de les classes que us proporcionem. No us preocupeu per les parts privades ni per la implementació.
- Comenceu amb estratègies molt senzilles, que siguin fàcils de programar i de depurar, que és exactament allò que necessiteu al principi. Programmeu procediments bàsics i útils, i *assegureu-vos que funcionin correctament*.
- No us arrisqueu a ser eliminats abans que comenci la competició. Aconseguiu que el Jutge us accepti un jugador tan ràpidament com sigui possible. Un jugador acceptat representa tenir assegurada part de la nota!
- Compileu sovint, testegeu sovint. És *molt* més fàcil trobar i corregir els errors quan s'han canviat unes poques línies de codi que quan s'acumulen molts canvis de cop.
- Feu servir `cerr's` per posar xivatos, i `assert's` per comprovar que el codi fa el que toca. Però millor que comenteu el xivatos abans d'enviar el jugador al Jutge, ja que alenteixen els programes.
- Activeu les opcions de *debug* al Makefile per ajudar-vos a trobar possibles errors. Les eines `valgrind` i `gdb` (o `ddd`) us poden resultar útils per depurar els vostres programes.
- Conserveu diverses versions antigues (però que funcionin correctament) dels vostres jugadors, i no les toqueu per res.
- Per testejar un jugador nou, enfronteu-lo contra uns altres jugadors que no treguin cap missatge. Així només apareixeran per pantalla els missatges del nou jugador.
- Feu competir els vostres jugadors contra rivals diferents, i estudieu les partides. Encara que no podreu veure els codis dels altres estudiants, si sou capaços de deduir les seves tàctiques i us semblen útils, podeu mirar d'imitar-les, defensar-vos-en o millorar-les.
- No deixeu el vostre codi a ningú, ni tan sols d'una versió antiga. **Un cop feta la competició s'analitzaran els programes de cada ronda amb programes detectors de plagi.** També es compararan els jugadors d'aquest curs amb els jugadors de cursos anteriors.
- Tingueu en compte que podeu lliurar nous jugadors en qualsevol moment (excepte durant la fase final).
- No espereu fins al darrer moment per enviar els vostres programes. Si tothom ho fa, la cua del Jutge no dona l'abast.
- Finalment, insistim: Feu codis senzills. Compileu sovint, testegeu sovint. O assumiu-ne les conseqüències.