Pandemic

Martí Oller

23 de novembre de 2021

1 Regles del joc

El joc es situa a l'any 2025, amb l'aparició d'un nou virus, que representa el cop gràcia a una civilització ja extremadament fràgil. Com a conseqüència, la societat col·lapsa, la borsa cau estrepitosament i milions de persones moren. D'entre els pocs supervivents, quatre faccions rivals sorgeixen i competeixen entre elles pel control d'una societat en ruïnes. Qui guanyarà?

En aquest joc, quatre jugadors competeixen per dominar les ruïnes de la societat. El guanyador del joc és el jugador que, al final de la partida, té la màxima puntuació.

El mapa on es desenvolupa la partida es representa en un tauler quadrat generat de forma aleatòria. Les cel·les del tauler poden ser de quatre tipus diferents: `WALL` (paret), `GRASS` (herba), `CITY` (ciutat) o `PATH` (camí). Les cel·les de tipus ciutat estan agrupades en rectangles representant ciutats. De forma similar, les de tipus camí s'agrupen en seqüències que formen camins disjunts que connecten ciutats. Finalment, hi ha cel·les de tipus paret que envolten tot el tauler, així com algunes més a l'interior que representen ciutats abandonades i camins que són inaccessibles.

Cada jugador controla un nombre d'unitats, que es poden moure a cada ronda. Qualsevol jugador que intenti donar més de 1000 ordres a la mateix ronda s'avortarà. Les unitats poden romandre quietes o moure's en qualsevol de les quatre direccions naturals: amunt, avall, dreta i esquerra. Si una unitat rep més d'una instrucció, s'ignoraran totes excepte la primera.

Les cel·les poden estar ocupades per com a molt una unitat. Per tant, una unitat A no pot moure's tranquil·lament a una cel·la on hi hagi una altra unitat B. Si A vol moure's a una cel·la ocupada per B, i ambdues unitats pertanyen al mateix jugador, no passarà res (A no es mourà). En canvi, si pertanyen a jugadors diferents, aleshores A atacarà a B, i B perdrà entre 25 i 40 punts de vida (de forma aleatòria amb probabilitat uniforme). Si B acaba amb una quantitat estrictament negativa de punts de vida per culpa de l'atac, aleshores B mor i es regenera sota el control del jugador que domina A, i A es mou a la posició de B. En canvi, si després de l'atac els punts de vida de B no són negatius, aleshores B sobreviu i A no es mou.

Cada cinc rondes, una màscara apareix en una cel·la de tipus herba. Les màscares es representen al mapa com a petits punts negres. Si una unitat que no porta màscara es mou a una cel·la on n'hi ha una, l'agafa automàticament. Si la unitat ja porta màscara, l'ignorarà. Si una unitat que porta màscara mor, la màscara desapareix.

Els punts s'obtenen conquerint ciutats i camins. Inicialment, estan totes buides i no tenen propietari. A mesura que el joc progressa, les unitats poden moure's cap a ciutats i camins i conquerir-los. Al final de cada ronda, es calcula el nombre d'unitats de cada jugador a cada ciutat i camí. Si un jugador té *estrictament*

més unitats que cap altre jugador en una ciutat o camí, la conquereix. En cas d'empat, el propietari de la ciutat o camí no canvia respecte la ronda prèvia.

A cada ronda, cada jugador guanya un cert nombre de punts que depèn de les ciutats i camins que té en propietat. Per cada ciutat sota el seu control, un jugador guanya bonus_per_city_cell()$\times$ mida de la ciutat. Similarment, cada camí proporciona al seu propietari un total de bonus_per_path_cell()$\times$ mida del camí. A més, per cada jugador es considera el seu *graf de conquestes*. En aquest graf, els vèrtexs són les ciutats del jugador, i les arestes són camins conquerits que connecten ciutats sota el seu control. Per cada component connexa de mida i, s'obtindrà un nombre addicional de $2^i \times$ factor_connected_component() punts.

Per exemple, considerem l'estat donat per la captura de pantalla de la Figura 1. El jugador vermell obtindrà el següent nombre de punts:

- La mida total de les ciutats controlades és $16 + 20 + 24 + 10 + 12 + 25 + 30 = 137$. Per tant, el jugador vermell obtindrà 137 punts si considerem que bonus_per_city_cell() $= 1$.
- La mida total dels camins sota control és $3 + 7 + 6 + 4 + 9 + 12 + 21 = 62$. Per tant, el jugador vermell rebrà 62 punts si bonus_per_path_cell() $= 1$.
- Finalment, el graf de conquestes té una component connexa de mida 3, una de mida 2 i dues de mida 1, atorgant $2 \times (2^3 + 2^2 + 2 \times 2^1) = 32$ punts si factor_connected_component() $= 2$.

En total, el jugador vermell rebrà 231 punts en aquesta ronda.

Algunes de les unitats s'infecten amb un virus potencialment mortal, que poden transmetre al seu entorn.

Una unitat infectada pel virus perdrà una quantitat fixa de punts de vida a cada ronda, que pot variar entre 2 i 5 (però és fixa per cada unitat infectada). Si els punts de vida de la unitat passen a ser negatius degut al virus, la unitat mor i es regenera sota el control d'un jugador aleatori (decidit aleatòriament amb probabilitat uniforme $1/4$). No obstant, les unitats es poden recuperar del virus. Si una unitat ha estat infectada pel virus durant t rondes, la probabilitat p de curar-se del virus es calcula de la manera següent:

$$p = \min\left(1,\ 0.001\left(\frac{t^2}{16} + 1\right)\right)$$

Intuïtivament, aquesta fórmula ens diu que una unitat que ha tingut el virus per més temps té més possibilitats de recuperar-se de la infecció. A la pràctica, la fórmula implica que aproximadament el 50% de les infeccions desapareixeran durant les primeres 31 rondes, i aproximadament el 90% d'elles hauran desaparegut a la 47ena ronda (assumint que la unitat encara viu).

Una unitat que es recupera del virus es torna immune a la infecció i no pot tornar a infectar-se.

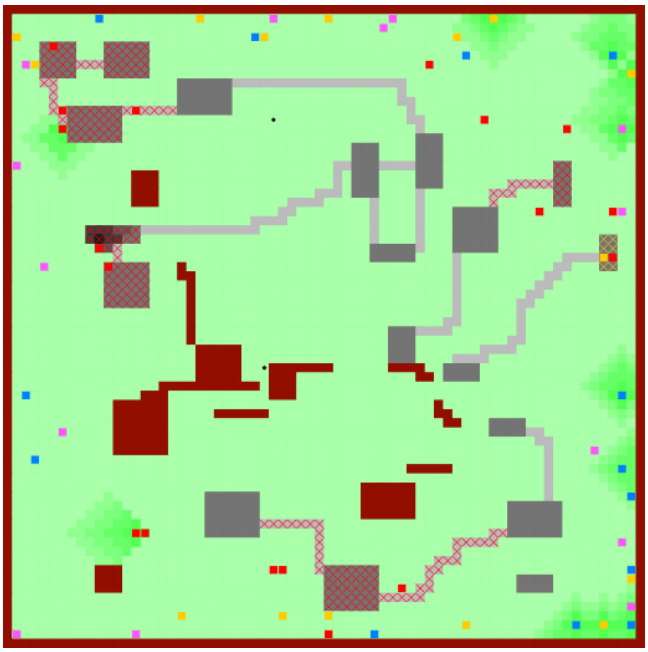

Figura 1: Captura de pantalla del joc

Les unitats infectades poden propagar el virus al seu voltant. Cada cel·la c té una certa quantitat de virus, que es representa per un enter `c.virus` que és inicialment zero. Al mapa, les cel·les amb altes concentracions de virus es representen amb un color més fosc. Cada ronda, `c.virus` s'actualitza amb el procediment següent:

1. A tota cel·la c que conté una unitat infectada que no porta màscara, la quantitat de virus s'incrementa en 3. Això no passa si la unitat infectada porta màscara.
2. En totes les cel·les del tauler que no siguin paret, la seva quantitat de virus s'actualitza prenent el màxim dels següents (com a molt) 5 nombres:
   - L'actual quantitat de virus a la cel·la menys 1.
   - L'actual quantitat de virus en qualsevol de les cel·les veïnes menys 1, però no totes les cel·les veïnes es consideren per a aquest càlcul: per les cel·les d'herba, només les cel·les veïnes que són herba es consideren. Per les cel·les de tipus ciutat, només les cel·les veïnes que són ciutat o camí es consideren. El mateix passa per les cel·les de camí: només les cel·les veïnes que són ciutat o camí es consideren. En altres paraules, el virus no travessa parets i no es propaga de l'herba a ciutats/camins i viceversa.

3. La quantitat de virus és com a molt 4 en les cel·les d'herba, com a molt 10 en ciutats i camins, i és sempre una quantitat no negativa.

Les unitats es poden infectar si estan en cel·les amb una quantitat no nul·la de virus. La probabilitat p_1 que una unitat no-immune i sense màscara s'infecti en una cel·la que té una quantitat v de virus és $p_1 = v/\text{factor\_infection}()$. Si la unitat porta màscara, aleshores la probabilitat és $p_2 = p_1/\text{mask\_protection}()$. Amb els paràmetres per defecte del joc, aquests nombres són $p_1 = v/50$ i $p_2 = v/1000$. Les unitats immunes no es poden reinfectar i les unitats que estan infectades tampoc es poden reinfectar.

Cada vegada que una unitat s'infecta, la quantitat de punts de vida que perdrà a cada ronda es fixa a un nombre entre 2 i 5 amb probabilitat uniforme. Aquesta quantitat s'anomena *damage*, i no canvia per a una unitat infectada.

Quan una unitat mor, ja sigui per un atac o pel virus, es regenera amb el màxim nombre de punts de vida en una cel·la frontera amb la paret exterior del tauler. Totes les unitats regenerades tenen un 20% de probabilitat de tenir el virus, i el seu *damage* es fixa a un nombre entre 2 i 4 (mai 5 en aquest cas) amb probabilitat uniforme.

Inicialment, a la ronda 0, exactament 3 unitats de cada jugador tenen el virus, amb *damage* fixat a 2, 3 i 4, respectivament.

En general, l'execució d'una ronda es fa seguint els següents passos en aquest ordre:

1. Totes les instruccions de tots els jugadors es registren d'acord amb les regles ja explicades.
2. Les instruccions s'ordenen de forma aleatòria i s'executen (sempre que siguin vàlides).
3. El virus es propaga per les unitats infectades a les cel·les veïnes com hem explicat anteriorment.
4. Les unitats es poden infectar o curar del virus, i les unitats infectades perden punts de vida depenent del seu *damage*.
5. Les unitats mortes es regeneren.
6. Si la ronda és múltiple de 5, una màscara apareixerà en una cel·la d'herba.
7. Per cada jugador, es calculen els punts i s'acumulen a la puntuació.

El joc es defineix per un tauler i el següent conjunt de paràmetres, amb el seu valor per defecte entre parèntesis:

- *nb_players()*: nombre de jugadors (4).
- *rows()*: nombre de files (70).
- *cols()*: nombre de columnes (70).

- *nb_rounds()*: nombre de rondes (200).
- *initial_health()*: quantitat inicial de punts de vida per cada unitat (100).
- *nb_units()*: nombre d'unitats que cada jugador controla inicialment (15).
- *bonus_per_city_cell()*: punts per cada cel·la en una ciutat conquerida (1).
- *bonus_per_path_cell()*: punts per cada cel·la en un camí conquerit (1).
- *factor_connected_component()*: factor que multiplica la mida de les components connexes (2).
- *infection_factor_()*: factor que divideix la probabilitat d'infecció (50).
- *mask_protection_()*: factor que divideix la probabilitat d'infecció si es porta màscara (20).

Excepte per causes de força major, aquests paràmetres per defecte seran els que utilitzarem en totes les partides.

2 Programar el joc

El primer que heu de fer és descarregar el codi font. Inclou un programa de C++ que executa les partides i un visor HTML per veure-les en un format animat raonable. A més, també es proporciona un jugador "Null" i un jugador "Demo" per facilitar la implementació del vostre propi jugador.

2.1 Executar la primera partida

Aquí explicarem com executar el joc sota Linux, però el mateix hauria de funcionar sota Windows, Mac, FreeBSD, OpenSolaris, ...Només cal una versió recent de `g++`, `make`, i un navegador modern com Mozilla Firefox o Google Chrome.

Per executar la vostra primera partida, seguiu els passos següents:

1. Obriu la consola i feu `cd` al directori on heu extret el codi font.
2. Executeu:

   ```
   cp AIDummy.o.Linux64 AIDummy.o
   ```

   per copiar l'arxiu objecte del jugador "Dummy" (copieu `AIDummy.o.MacOS` si utilitzeu Mac).
3. Executeu

   ```
   make all
   ```

   per construir el joc i tots els jugadors. Noteu que `Makefile` identifica com a jugador qualsevol fitxer `AI*.cc`.

4. Això crea un fitxer executable anomenat `Game`. Aquest executable us permet executar el joc usant una comanda com:

   ```
   ./Game Demo Demo Demo Demo -s 30 -i default.cnf -o default.out
   ```

   En aquest cas, això comença una partida amb llavor aleatòria 30 de quatre clons del jugador "Demo", sobre el tauler definit a `default.cnf` (els paràmetres per defecte). La sortida d'aquesta partida es redirigeix al fitxer `default.out`.

5. Per veure la partida, obriu el fitxer `viewer.html` del directori `Viewer` amb el navegador i carregueu el fitxer `default.out`.

   Nota: Per aconseguir una visió més gran de la partida, poseu el navegador en mode pantalla completa (pitgeu la tecla F11).

Useu

```
./Game --help
```

per veure la llista de paràmetres que podeu usar. És particularment útil

```
./Game --list
```

per veure tots els noms de jugadors registrats.

Si cal, recordeu que podeu executar

```
make clean
```

per esborrar tots els fitxers executables i objectes i començar de nou.

2.2 Afegir el vostre jugador

Per crear un nou jugador amb, per exemple, nom `Manolito`, copieu `AINull.cc` (un jugador buit que es proporciona com a plantilla) a un nou fitxer `AIManolito.cc`. Aleshores, editeu el nou fitxer i canvieu el

#define PLAYER_NAME Null

per

#define PLAYER_NAME Manolito

El nom escollit per al vostre jugador ha de ser únic, no ofensiu i de com a molt 12 caràcters de longitud. Aquest nom s'usarà per definir una nova classe *PLAYER_NAME*, a la que ens referirem com a la vostra classe jugador d'ara en endavant. El nom serà mostrat a la web i durant les partides.

Ara ja podeu començar a implementar el vostre mètode *play*(). Aquest mètode serà cridat a cada ronda i és on el vostre jugador ha de decidir què fer, i ferho. Naturalment, podeu definir mètodes i variables auxiliars dins de la vostra

classe jugador, però el punt d'entrada del vostre codi serà sempre el mètode *play* () .

Des de la vostra classe jugador també podeu cridar funcions per accedir a l'estat del joc, tal com es defineix a la classe *State* a `State.hh`, i per ordenar a les vostres unitats, tal com es defineix a la classe *Action* a `Action.hh`. Aquestes funcions estan disponibles al vostre codi usant herència múltiple. La documentació sobre les funcions disponibles es pot trobar en els fitxers `.hh` abans esmentats. També podeu examinar el codi del jugador "Demo" a `AIDemo.cc` com a exemple de com usar aquestes funcions. Finalment, també pot ser profitós fer un cop d'ull als fitxers `Structs.hh` per estructures de dades útils, `Random.hh` per utilitats per generar nombres aleatoris, `Settings.hh` per consultar els paràmetres del joc i `Player.hh` pel mètode *me*().

Noteu que no heu d'editar el mètode *factory* () de la vostra classe jugador, ni tampoc la darrera línia que afegeix el vostre jugador a la llista de jugadors registrats.

2.3 Jugar contra el jugador "Dummy"

Per a provar la vostra estratègia contra el jugador Dummy, proporcionem el seu arxiu objecte. D'aquesta manera, no teniu accés al seu codi font però podreu afegir-lo com a jugador i competir contra ell en local.

Com ja hem comentat, per afegir el jugador Dummy a la llista de jugadors registrats, heu de copiar l'arxiu corresponent a la vostra arquitectura cap a `AIDummy.o`. Per exemple:

```
cp AIDummy.o.Linux64 AIDummy.o
```

Consell de pro: demaneu als vostres amics els seus arxius **objecte** (mai codi font!!!) i afegiu-los al vostre `Makefile`!

2.4 Restriccions en l'enviament d'un jugador

Quan creieu que el vostre jugador és prou fort com per entrar a la competició, podeu enviar-lo a Jutge.org (`https://www.jutge.org`). Com que s'executarà en un entorn segur per evitar trampes, cal aplicar algunes restriccions al vostre codi:

- Tot el vostre codi font ha d'estar en un sol fitxer (com `AIManolito.cc`).
- No podeu usar variables globals (en lloc d'això, useu atributs de la vostra classe).
- Només se us permet usar llibreries estàndard com ara `iostream`, `vector`, `map`, `set`, `queue`, `algorithm`, `cmath`, ... En molts casos, no cal ni tan sols que incloeu la corresponent llibreria.

- No podeu obrir fitxers ni fer altres crides a sistema (threads, forks, ...).
- El vostre temps de CPU i la memòria que utilitzeu seran limitats, mentre que no ho són al vostre entorn local quan executeu `./Game`. El temps límit és d'un segon per l'execució de tota la partida. Si exhauriu el temps límit (o si l'execució del vostre codi avorta), el vostre jugador es congelarà i no admetrà més instruccions.
- El vostre programa no hauria d'escriure a **cout** ni llegir de **cin**. Podeu escriure informació de depuració a **cerr**, però recordeu que fer això en el codi que pugeu al servidor pot malgastar part del vostre temps limitat de CPU.
- Qualsevol enviament a Jutge.org ha de ser un intent honest de jugar el joc. Qualsevol intent de fer trampa de qualsevol manera serà severament penalitzat.
- Un cop hagueu enviat un jugador al Jutge que hagi derrotat al Dummy, podeu fer més enviaments però haureu de canviar el nom del jugador. És a dir, un cop un jugador ha vençut al Dummy, el seu nom queda bloquejat i no es pot reutilitzar.

3 Consells

- **NO DONEU O DEMANEU EL VOSTRE CODI A NINGÚ**. Ni tan sols una versió antiga. Ni fins i tot al vostre millor amic. Ni tans sols d'estudiants d'anys anteriors. Utilitzem detectors de plagi per comparar els vostres programes, també contra enviaments de jocs d'anys anteriors. No obstant, podeu compartir arxius objecte.

  Qualsevol plagi implicarà **una nota de 0 en l'assignatura** (no només del Joc) de tots els estudiants involucrats. Es podran també prendre mesures disciplinàries addicionals. Si els estudiants A i B es veuen implicats en un plagi, les mesures s'aplicaran als dos, independentment de qui va crear el codi original. No es farà cap excepció sota cap circumstància.
- Abans de competir amb els companys, concentreu-vos en derrotar al Dummy.
- Llegiu les capçaleres de les classes que aneu a utilitzar. No cal que mireu les parts privades o la implementació.
- Comenceu amb estratègies simples, fàcils d'implementar i depurar, ja que és exactament el que necessitareu al principi.
- Definiu mètodes auxiliars senzills (però útils) i *assegureu-vos que funcionin correctament*.
- Intenteu mantenir el vostre codi net. Això farà més fàcil canviar-lo i afegir noves estratègies.

- Com sempre, compileu i proveu el vostre codi sovint. És *molt* més fàcil rastrejar un error quan només heu canviat poques línies de codi.
- Utilitzeu **cerr** per produir informació de depuració i afegiu `asserts` per assegurar-vos que el vostre codi fa el que hauria de fer. No oblideu eliminar-los abans de pujar el codi. En cas de no fer-ho, el vostre jugador morirà.
- Quan depureu un jugador, elimineu els **cerr**s que tingueu en el codi d'altres jugadors, per tal de veure només els missatges que desitgeu.
- Podeu utilitzar comandes com el `grep` de Linux per tal de filtrar la sortida produïda per `Game`.
- Activeu l'opció `DEBUG` al Makefile, que us permetrà obtenir traces útils quan el vostre programa avorta. També hi ha una opció `PROFILE` que podeu utilitzar per optimitzar codi.
- Si l'ús de **cerr** no és suficient per depurar el vostre codi, apreneu com utilitzar `valgrind`, `gdb` o qualsevol altra eina de depuració.
- Podeu analitzar els arxius produïts per `Game`, que descriuen com evoluciona el tauler a cada ronda.
- Conserveu una còpia de les versions antigues del vostre jugador. Feu-lo lluitar contra les seves versions anteriors per quantificar les millores.
- Quan un jugador s'envia al servidor Jutge.org o durant la competició, les partides es duen a terme amb diferents llavors aleatòries. De forma que quan us entreneu localment, executeu partides també amb diferents llavors aleatòries (amb l'opció `-s` de `Game`).
- Assegureu-vos que el vostre programa sigui prou ràpid. El temps de CPU que es permet utilitzar és bastant curt.
- Intenteu esbrinar les estratègies dels altres jugadors observant diverses partides. D'aquesta manera, podeu intentar reaccionar als seus moviments, o fins i tot imiteu-los o milloreu-los amb el vostre propi codi.
- No espereu fins al darrer minut per enviar el jugador. Quan hi ha molts enviaments al mateix temps, el servidor triga més en executar les partides i podria ser ja massa tard!
- Podeu enviar noves versions del vostre programa en qualsevol moment.
- Recordeu: mantingueu el codi senzill, compileu-lo sovint i proveu-lo sovint, o us en penedireu.